

The L^AT_EX 2_ε Sources

Johannes Braams
David Carlisle
Alan Jeffrey
Leslie Lamport
Frank Mittelbach
Chris Rowley
Rainer Schöpf

2021-06-01

This file is maintained by the L^AT_EX Project team.
Bug reports can be opened (category `latex`) at
<https://latex-project.org/bugs.html>.

Contents

a	ltdirchk.dtx	1
1	L^AT_EX System Dependent Initializations	1
2	Initialization	2
2.1	INITEX	2
2.2	Some bits of 2e	4
3	texsys.cfg	5
3.1	texsys.cfg	5
3.2	UNIX (web2c)	6
3.3	UNIX (other)	7
3.4	MSDOS (emtex)	7
3.5	MSDOS (other)	7
3.6	VMS (DECUS T _E X, PD VMS 3.6)	7
3.7	VMS (???)	7
3.8	MACINTOSH (OzTeX 1.6)	8
3.9	MACINTOSH (other)	8
3.10	FAKE EXAMPLE	8
4	Setting \@currdir	9
5	Setting \input@path	10

6	Filename Parsing	11
7	T_EX Versions	13
8	ltxcheck.tex	13
b	lplain.dtx	14
1	Plain T_EX	14
c	ltvers.dtx	36
1	Version Identification	36
1.1	Declaring an all-new module	39
d	lualatex.dtx	41
1	Overview	41
2	Core T_EX functionality	41
3	Plain T_EX interface	42
4	Lua functionality	42
4.1	Allocators in Lua	42
4.2	Lua access to T _E X register numbers	43
4.3	Module utilities	44
4.4	Callback management	44
5	Implementation	45
5.1	Minimum LuaT _E X version	45
5.2	Older L ^A T _E X/Plain T _E X setup	45
5.2.1	Fixes to <code>etex.src/etex.sty</code>	46
5.2.2	luatex specific settings	46
5.3	Attributes	47
5.4	Category code tables	47
5.5	Named Lua functions	49
5.6	Custom whatsits	50
5.7	Lua bytecode registers	50
5.8	Lua chunk registers	50
5.9	Lua loader	50
5.10	Lua module preliminaries	52
5.11	Lua module utilities	52
5.11.1	Module tracking	52
5.11.2	Module messages	53
5.12	Accessing register numbers from Lua	54
5.13	Attribute allocation	55
5.14	Custom whatsit allocation	56

5.15	Bytecode register allocation	56
5.16	Lua chunk name allocation	57
5.17	Lua function allocation	57
5.18	Lua callback management	57
5.18.1	Housekeeping	57
5.18.2	Handlers	60
5.18.3	Public functions for callback management	62
e	ltxexpl.dtx	68
1	expl3-dependent code	68
1.1	Loader	68
1.2	Using expl3 code	71
f	ltxdefns.dtx	73
1	Definitions	73
1.1	Initex initializations	73
1.2	Saved versions of TeX primitives	73
1.3	Command definitions	74
1.4	Robust commands and protect	83
1.5	Acting on robust commands	89
1.5.1	Copying robust commands	91
1.5.2	Showing robust commands	92
1.5.3	Commands defined with <code>\DeclareRobustCommand</code>	93
1.5.4	Commands defined with <code>\newcommand</code> (with optional argument)	95
1.6	Internal defining commands	96
2	Discretionary Hyphenation	100
g	ltxcmd.dtx	103
1	Creating document commands	103
1.1	Variables and constants	103
1.2	Declaring commands and environments	106
1.3	Structure of xparse commands	110
1.4	Normalizing the argument specifications	115
1.5	Preparing the signature: general mechanism	123
1.6	Setting up a standard signature	124
1.7	Setting up expandable types	127
1.8	Grabbing arguments	131
1.9	Grabbing arguments expandably	142
1.10	Argument processors	148
1.11	Access to the argument specification	150
1.12	Utilities	152
1.13	Messages	156
1.14	User functions	161

h	lthooks.dtx	165
1	Introduction	165
2	Package writer interface	165
2.1	L ^A T _E X 2 _ε interfaces	165
2.1.1	Declaring hooks	165
2.1.2	Special declarations for hooks	166
2.1.3	Using hooks in code	166
2.1.4	Updating code for hooks	167
2.1.5	Hook names and default labels	168
2.1.6	The <code>top-level</code> label	170
2.1.7	Defining relations between hook code	171
2.1.8	Querying hooks	172
2.1.9	Displaying hook code	173
2.1.10	Debugging hook code	174
2.2	L3 programming layer (<code>expl3</code>) interfaces	175
2.3	On the order of hook code execution	177
2.4	The use of “reversed” hooks	178
2.5	Difference between “normal” and “one-time” hooks	179
2.6	Private L ^A T _E X kernel hooks	180
2.7	Legacy L ^A T _E X 2 _ε interfaces	181
2.8	L ^A T _E X 2 _ε commands and environments augmented by hooks	181
2.8.1	Generic hooks for all environments	181
2.8.2	Generic hooks for commands	183
2.8.3	Generic hooks provided by file loading operations	183
2.8.4	Hooks provided by <code>\begin{document}</code>	183
2.8.5	Hooks provided by <code>\end{document}</code>	184
2.8.6	Hooks provided by <code>\shipout</code> operations	185
2.8.7	Hooks provided in NFSS commands	185
3	The Implementation	186
3.1	Debugging	186
3.2	Borrowing from internals of other kernel modules	186
3.3	Declarations	187
3.4	Providing new hooks	189
3.4.1	The data structures of a hook	189
3.4.2	On the existence of hooks	190
3.4.3	Setting hooks up	191
3.4.4	Disabling and providing hooks	194
3.5	Parsing a label	195
3.6	Adding or removing hook code	198
3.7	Setting rules for hooks code	205
3.8	Specifying code for next invocation	219
3.9	Using the hook	220
3.10	Querying a hook	222
3.11	Messages	223
3.12	L ^A T _E X 2 _ε package interface commands	225
3.13	Internal commands needed elsewhere	230

i	ltcmdhooks.dtx	231
1	Introduction	231
2	Restrictions and Operational details	232
2.1	Patching	232
2.1.1	Timing	232
2.2	Commands that look ahead	232
3	Package Author Interface	233
4	The Implementation	234
4.1	Execution plan	234
4.2	Variables	235
4.3	Variants	235
4.4	Patching or delaying	236
4.5	Patching commands	237
4.5.1	Patching by expansion and redefinition	238
4.5.2	Patching by retokenization	241
4.6	Messages	245
j	ltalloc.dtx	246
1	Counters	246
k	ltcntrl.dtx	248
1	Program control structure	248
l	lterror.dtx	252
1	Error handling and tracing	252
1.1	General commands	252
1.2	Specific errors	257
1.3	Tracing	261
m	ltpar.dtx	262
1	Paragraphs	262
1.1	Implementation	262
n	ltpara.dtx	264
1	Introduction	264
1.1	The default processing done by the engine	264

2	The new mechanism implemented for L^AT_EX	266
2.1	The provided hooks	267
2.2	Altered and newly provided commands	268
2.3	Examples	269
2.3.1	Testing the mechanism	269
2.3.2	Mark the first paragraph of each <code>itemize</code>	271
2.4	Some technical notes	271
2.4.1	Glue items between paragraphs (found with <code>fancypar</code>)	271
3	The Implementation	272
3.1	Providing hooks for paragraphs	272
3.2	The error messages	277
o	ltspace.dtx	279
1	Spacing	279
1.1	User Commands	279
1.2	Chris' comments	279
1.3	Some immediate actions	281
1.4	The code	282
1.5	Vertical spacing	289
1.6	Horizontal space (and breaks)	294
p	ltlogos.dtx	298
1	Logos	298
q	ltfiles.dtx	299
1	File Handling	299
1.1	Safe Input Macros	311
1.2	Listing files	318
r	ltoutenc.dtx	320
1	Font encodings	320
1.1	Removing encoding-specific commands	322
1.2	The order of declarations	323
1.3	Docstrip modules	323
1.4	Definitions for the kernel	324
1.4.1	Declaration commands	324
1.4.2	Hyphenation	332
1.4.3	Miscellania	332
1.4.4	Default encodings	332
1.4.5	Math material	335
1.5	Definitions for the OT1 encoding	336
1.6	Definitions for the T1 encoding	338

1.7	Definitions for the OMS encoding	344
1.8	Definitions for the OML encoding	345
1.9	Definitions for the OT4 encoding	345
1.10	Definitions for the TS1 encoding	347
1.11	Definitions for the TU encoding	352
2	Package files	362
2.1	The fontenc package	363
s	ltcounts.dtx	366
1	Counters and Lengths	366
1.1	Environment Counter Macros	366
t	ltlength.dtx	374
1	Lengths	374
u	ltfssbas.dtx	376
1	Preliminary macros	376
2	Macros for setting up the tables	377
3	Selecting a new font	384
3.1	Macros for the user	384
3.2	Macros for loading fonts	388
4	Assigning math fonts to <i>versions</i>	396
v	ltfssaxes.dtx	403
1	Changing the font series	403
1.1	The series lookup table	403
1.2	Mapping rules for series changes	404
1.3	Changing to a new series	412
2	Changing the shape	417
2.1	Mapping rules for shape combinations	418
2.2	Changing to a new shape	420
3	Make sure we win ...	422
w	lfsstrc.dtx	424
1	Introduction	424

2	A driver for this document	424
3	The Implementation	425
4	Handling Options	425
5	Macros common to <code>fam.tex</code> and <code>tracefmt.sty</code>	427
5.1	General font loading	427
5.2	Math fonts setup	433
5.2.1	Outline of algorithm for math font sizes	433
5.2.2	Code for math font size setting	434
5.2.3	Other code for math	435
6	Scaled font extraction	437
6.1	Sizefunctions	445
x	<code>ltfsscmp.dtx</code>	449
y	<code>ltfssdcl.dtx</code>	454
1	Interface Commands	454
z	<code>ltfssini.dtx</code>	481
1	NFSS Initialization	481
1.1	Providing math <i>versions</i>	481
2	Custom series settings for main document families	482
3	Supporting nested emphasis	496
3.1	Legacy	500
3.2	Miscellaneous	500
A	<code>fontdef.dtx</code>	506
1	Introduction	506
2	Customization	506
3	The docstrip modules	507
4	A driver for this document	507
5	The <code>fonttext.ltx</code> file	507
5.1	Encodings	508
5.2	Defaults	510

6	The fontmath.ltx file	512
6.1	The font encodings used	512
6.1.1	Symbolfont and Alphabet declarations	512
6.2	Math font sizes	513
6.3	The math symbol assignments	514
6.3.1	The letters	514
6.3.2	The digits	515
6.3.3	Punctuation, brace, etc. keys	515
6.3.4	Delimitercodes for characters	515
6.4	Symbols accessed via control sequences	516
6.4.1	Greek letters	516
6.4.2	Ordinary symbols	517
6.4.3	Large Operators	517
6.4.4	Binary symbols	518
6.4.5	Relations	519
6.4.6	Arrows	520
6.4.7	Punctuation symbols	521
6.4.8	Math accents	522
6.4.9	Radicals	522
6.4.10	Over and under something, etc	522
6.4.11	Delimiters	523
6.5	Math versions of text commands	524
6.6	Other special functions and parameters	524
6.6.1	Biggggg	524
6.6.2	The log-like functions	525
6.6.3	Parameters	525
7	Default cfg files	525
B	preload.dtx	527
1	Overview	527
2	Customization	527
3	Module switches for the DOCSTRIP program	527
4	A driver for this document	528
5	The code	528
C	lftncmd.dtx	530
1	Introduction	530
2	The implementation	532
3	Initialization	538

D	ltxcomp.dtx	539
1	Sub-encodings	543
1.1	Sub-encoding 1 (drop symbols not working in Latin Modern)	544
1.2	Sub-encoding 2 (majority of new OTF fonts via autoinst)	544
1.3	Sub-encoding 3	548
1.4	Sub-encoding 4	548
1.5	Sub-encoding 5 (most older PS fonts)	548
1.6	Sub-encoding 6	549
1.7	Sub-encoding 7	549
1.8	Sub-encoding 8	549
1.9	Sub-encoding 9 (most missing)	549
2	Unicode engine specials	549
3	Font family sub-encodings setup	550
4	Legacy symbol support for lists and footnote symbols	554
5	The textcomp package	559
5.1	The old textcomp package code	560
5.1.1	Supporting oldstyle digits	568
5.1.2	Subset encoding defaults	569
E	ltxpago.dtx	571
1	Page Numbering	571
F	ltxref.dtx	572
1	Cross Referencing	572
1.1	Cross Referencing	572
G	ltxmisc.dtx	577
1	Miscellaneous Environments	577
1.1	Environments	577
1.2	Center, Flushright, Flushleft	589
1.3	Verbatim	592
H	ltxmath.dtx	598

1	Math setup	598
1.1	Math commands based on plain \TeX	598
1.1.1	The log-like functions	598
1.1.2	Biggggg	599
1.1.3	The UNSORTED Rest	599
1.2	Math Environments	605
1.3	External options to the standard document classes	610
1.3.1	Left equation numbering	610
1.3.2	Flush left equations	610
I	ltlists.dtx	613
1	List, and related environments	613
1.1	List and Trivlist	614
1.2	Vertical Spacing (skips)	615
1.3	Penalties	615
1.4	Horizontal Spacing (dimens)	615
1.5	Default Values	615
1.6	Itemize and Enumerate	626
J	ltboxes.dtx	629
1	\LaTeX Box commands	629
1.1	Some low-level constructs	644
K	lttab.dtx	645
1	Tabbing, Tabular and Array Environments	645
1.1	tabbing	645
1.2	array and tabular environments	654
L	ltpictur.dtx	670
1	Picture Mode	670
1.1	Curves	697
M	ltthm.dtx	702
1	Theorem Environments	702
N	ltsect.dtx	706

1	Sectioning Commands	706
1.1	The Title	706
1.2	Sectioning	707
1.2.1	Initializations	714
1.3	Table of Contents etc.	714
1.3.1	Convention	714
1.3.2	Commands	714
O	lfloat.dtx	719
1	Floats	719
1.1	Floating Environments	719
1.2	Footnotes	733
P	ltxglo.dtx	742
1	Index and Glossary Generation	742
Q	ltbibl.dtx	745
1	Bibliography Generation	745
1.1	Default definitions	749
R	ltpage.dtx	750
1	Page styles and related commands	750
1.1	Page Style Commands	750
1.2	How a page style makes running heads and feet	750
1.3	marking conventions	750
S	ltxclass.dtx	754
1	Introduction	754
2	User interface	754
2.1	Option processing	755
3	Class and Package interface	756
3.1	Class name and version	756
3.2	Package name and version	756
3.3	Requiring other packages	756
3.4	Declaring new options	757
3.5	Safe Input Macros	758

4	Implementation	758
4.1	Hooks	784
4.2	Providing shipment	786
5	Package/class rollback mechanism	794
6	After Preamble	802
T	ltfilehook.dtx	803
1	Introduction	803
1.1	Provided hooks	803
1.2	General hooks for file reading	803
1.3	Hooks for package and class files	804
1.4	Hooks for <code>\include</code> files	805
1.5	High-level interfaces for <code>L^AT_EX</code>	806
1.6	Internal interfaces for <code>L^AT_EX</code>	806
1.7	A sample package for structuring the log output	807
2	The Implementation	808
2.1	Document and package-level commands	808
2.2	<code>expl3</code> helpers	808
2.3	Declaring the file-related hooks	811
2.4	Patching <code>L^AT_EX</code> 's <code>\InputIfFileExists</code> command	811
2.5	Declaring a file substitution	814
2.6	Selecting a file (<code>\set@curr@file</code>)	815
2.7	Replacing a file and detecting loops	818
2.7.1	The Tortoise and Hare algorithm	819
2.8	Preventing a package from loading	821
2.9	High-level interfaces for <code>L^AT_EX</code>	821
2.10	Internal commands needed elsewhere	822
3	A sample package for structuring the log output	823
4	Package emulations	824
4.1	Package <code>atveryend</code> emulation	824
U	ltshipout.dtx	825
1	Introduction	825
1.1	Overloading the <code>\shipout</code> primitive	825
1.2	Provided hooks	826
1.3	Legacy <code>L^AT_EX</code> commands	828
1.4	Special commands for use inside the hooks	828
1.5	Provided <code>LuaT_EX</code> callbacks	829
1.6	Information counters	829
1.7	Debugging shipout code	830

2	Emulating commands from other packages	830
2.1	Emulating atbegshi	830
2.2	Emulating everyshi	831
2.3	Emulating atenddvi	831
2.4	Emulating everypage	832
3	The Implementation	832
3.1	Debugging	832
3.2	Handling the end of job hook	845
4	Legacy L^AT_EX 2_ε interfaces	847
5	Internal commands needed elsewhere	848
6	Package emulation for compatibility	849
6.1	Package atenddvi emulation	849
6.2	Package atbegshi emulation	850
6.3	Package everyshi emulation	851
V	ltoutput.dtx	852
1	Output Routine	852
1.1	Floats	852
1.1.1	Kludgeins	907
1.1.2	Float control	909
1.1.3	Float placement parameters	922
W	lthyphen.dtx	925
X	ltfinal.dtx	927
1	Final settings	927
1.1	Debugging	927
1.2	Typesetting parameters	927
1.3	Lccodes for hyphenation	930
1.4	Hyphenation	933
1.5	Font loading	934
1.6	Input encoding	935
1.7	Lccodes and uccodes	939
1.8	Applying Patch files	941
1.9	Freeing Memory	942
1.10	Initialise file list	943
1.11	Do some temporary work for pre-release	943
1.12	Some last minute initializations	943
1.13	Dumping the format	943
	Change History	944

File a

ltdirchk.dtx

1 L^AT_EX System Dependent Initializations

This file implements the semi-automatic determination of various system dependent parts of the initialization. The actual definitions may be placed in a file `texsys.cfg`. Thus for operating systems for which the tests here do not result in acceptable settings, a ‘hand written’ `texsys.cfg` may be produced.

The macros that must be defined are:

`\@currdir` `\@currdir{filename}{space}` should expand to a form of the filename that uniquely refers to the ‘current directory’ if this is possible. (The expansion should also end with a space.) on UNIX, this is `\def\@currdir{./}`. For more exotic operating systems you may want to make `\@currdir` a macro with arguments delimited by `.` and/or `{space}`. If the operating system has no concept of directory structure, this macro should be defined to be empty.

`\input@path` If the primitive `\openin` searches the same directories as the primitive `\input`, then it is possible to tell (using `\ifeof`) whether a file exists before trying to input it. For systems like this, `\input@path` should be left undefined.

If `\openin` does not ‘follow’ `\input` then `\input@path` must be defined to be a list of directories to search for input files. The format for each directory is as for `\@currdir`, normally just a prefix is required, but it may be a macro with space-delimited argument. That is, if `<dir>` is an entry in the input path, T_EX will try to load the expansion of `<dir>{filename}{space}`

So either `<dir>` should be defined as a macro with argument delimited by space, or it should just expand to a directory name, including the final directory separator, so that it may be concatenated with the `<filename>`. This means that for UNIX-like syntax, each `<dir>` should end with a slash, `/`.

`\input@path` should expand to a list of such directories, each in a `{}` group.

`\filename@parse` After a call of the form: `\filename@parse{<filename>}`, the three macros `\filename@area`, `\filename@base` and `\filename@ext` should be defined to be the ‘area’ (or directory), basename and extension respectively. If there was no extension specified in `<filename>`, `\filename@ext` should be `\let` to `\relax` (so this case may be tested with `\@ifundefined{filename@ext}` and, perhaps a default extension substituted).

Normally one would not need to define this macro in `texsys.cfg` as the automatic tests can supply parsers that work with UNIX and VMS and Macintosh syntax, as well as a basic parser that will cover many other cases. However some operating systems may need a ‘hand produced’ parser in which case it should be defined in this file.

The UNIX parser also works for most MSDOS T_EX versions. Currently if the UNIX, VMS or Macintosh parser is not used, `\filename@parse` is defined to always return an empty area, and to split the argument into basename and extension at the first ‘.’ that occurs in the name. Parsers for other formats may be defined in `texsys.cfg`, in which case they will be used in preference to the default definitions.

`\@TeXversion` `\@TeXversion` is now set automatically by the initialization tests in this file. You should not need to set it in `texsys.cfg`, however the following documentation is left for information. L^AT_EX does not set this variable exactly, the automatic tests set it to:

2 for any version, v , $v < 3.0$

3 for any version, v , $3.0 \leq v \leq 3.14$

<undefined> otherwise.

However these values are accurate enough for L^AT_EX to take appropriate action for these old T_EXs.

If your T_EX is older than version 3.141, then you should define `\@TeXversion` (using `\def`) to be the version number. If you do not do this¹, L^AT_EX will not work around a bug in old T_EX versions, and so error messages will appear in a very strange format, with `^^J` appearing instead of line breaks:

```
LaTeX Error: \rubbish undefined.^^J^^JSee the LaTeX manual or LaTeX=
Companion
for explanation.^^JType H <return> for immediate help.
...

.3 \renewcommand{\rubbish}
      {}
```

However if you put `\def\@TeXversion{3.14}` in `texsys.cfg` the following format will be used:

```
LaTeX Error: \rubbish undefined.

ee the LaTeX manual or LaTeX Companion for explanation.
ype H <return> for immediate help.
.
...

.3 \renewcommand{\rubbish}
      {}
```

Note that this has an extra line ! . which does not appear in error messages that use the default settings with a current version of T_EX, but this should not cause any confusion we hope.

2 Initialization

As this file is read at a very early stage, some definitions that are normally considered to be part of the format must be made here.

2.1 INITEX

```
1 <*dircheck>
2 <*initex>
3 <initex>\ifnum\catcode'\{=1
4 <initex> \errmessage
5 <initex> {LaTeX must be made using an initex with no format preloaded}
6 <initex>\fi
7 \catcode'\{=1
```

¹Actually if your T_EX is really old, version 2, L^AT_EX can detect this, and sets `\@TeXversion` to 2 if it is not set in the `cfg` file.

```
8 \catcode'\}=2
```

If LuaTeX is in use the extensions and other new primitives have to be activated: this is done as early as possible. Older versions of LuaTeX do not hide the primitives: a version check is not needed as the version itself will be missing in the case where action is needed!

```
9 \ifx\directlua\undefined
10 \else
11 \ifx\luatexversion\undefined
```

Enable e-TeX/pdfTeX/Umath primitives with their natural names

```
12 \directlua{tex.enableprimitives("",%
13 tex.extraprimitives('etex', 'pdftex', 'umath'))}
```

In current formats enable primitives with unprefix names. the latexrelease guards allow the primitives to be defined with a \luatex prefix if older formats are specified.

```
14 \</initex>
15 \</dircheck>
16 \*initex, latexrelease>
17 \<latexrelease>\ifx\directlua\undefined\else
18 \<latexrelease>\IncludeInRelease{2015/10/01}{\luatexluafunction}
19 \<latexrelease>{LuaTeX (prefixed names)}%
20 \directlua{tex.enableprimitives("",%
21 tex.extraprimitives("omega", "aleph", "luatex"))}
22 \<latexrelease>\EndIncludeInRelease
23 \<latexrelease>\IncludeInRelease{0000/00/00}{\luatexluafunction}
24 \<latexrelease>{LuaTeX (prefixed names)}%
25 \<latexrelease>\directlua{
26 \<latexrelease> tex.enableprimitives(
27 \<latexrelease> "luatex",
28 \<latexrelease> tex.extraprimitives("core", "omega", "aleph", "luatex")
29 \<latexrelease> )
30 \<latexrelease> local i
31 \<latexrelease> local t = { }
32 \<latexrelease> for _,i in pairs(tex.extraprimitives("luatex")) do
33 \<latexrelease> if not string.match(i, "U") then
34 \<latexrelease> if not string.match(i, "~luatex") then
35 \<latexrelease> table.insert(t,i)
36 \<latexrelease> end
37 \<latexrelease> else
38 \<latexrelease> if string.match(i, "~Uchar$") then
39 \<latexrelease> table.insert(t,i)
40 \<latexrelease> end
41 \<latexrelease> end
42 \<latexrelease> end
43 \<latexrelease> for _,i in pairs(t) do
44 \<latexrelease> tex.print(
45 \<latexrelease> "\noexpand\let\noexpand\" .. i
46 \<latexrelease> .. "\noexpand\undefined"
47 \<latexrelease> )
48 \<latexrelease> end
49 \<latexrelease>}
50 \<latexrelease>\EndIncludeInRelease
51 \<latexrelease>\fi
52 \</initex, latexrelease>
53 \*dircheck>
54 \*initex>
```

```

55 \fi
56 \fi

A test can now be made for eTeX.
57 \initex\ifx\TeXversion\undefined
58 \initex \errmessage
59 \initex {LaTeX requires e-TeX}
60 \initex \expandafter\endinput
61 \initex\fi

That distraction over, back to the basics of a format.
62 \catcode'\#=6
63 \catcode'\^=7
64 \chardef\active=13
65 \catcode'\@=11
66 \countdef\count@=255
67 \let\bgroup={ \let\egroup=}
68 \ifx\@@input\@undefined\let\@@input\input\fi
69 \ifx\@@end\@undefined\let\@@end\end\fi
70 \chardef\@inputcheck0
71 \chardef\sixt@n=16
72 \newlinechar'\^^J
73 \def\typeout{\immediate\write17}
74 \def\dospecials{\do\ \do\\\do\{\do\}\do\$\do\&%
75 \do\#\do\^{\do\_ \do\% \do\~}
76 \def\@makeother#1{\catcode'#1=12\relax}
77 \def\space{ }
78 \def\@tempswafalse{\let\if@tempswa\iffalse}
79 \def\@tempswatrue{\let\if@tempswa\iftrue}
80 \let\if@tempswa\iffalse
81 \def\loop#1\repeat{\def\iterate{#1\relax\expandafter\iterate\fi}%
82 \iterate \let\iterate\relax}
83 \let\repeat\fi
84 \endinitex

```

2.2 Some bits of 2e

```

85 \*2kernel)
86 \def\two@digits#1{\ifnum#1<10 0\fi\number#1}
87 \long\def\@firstoftwo#1#2{#1}
88 \long\def\@secondoftwo#1#2{#2}

```

This is a special version of \ProvidesFile for initex use.

```

89 \def\ProvidesFile#1{%
90 \begingroup
91 \catcode'\ 10 %
92 \ifnum \endlinechar<256 %
93 \ifnum \endlinechar>\m@ne
94 \catcode\endlinechar 10 %
95 \fi
96 \fi
97 \@makeother\%
98 \@ifnextchar[{\@providesfile{#1}}{\@providesfile{#1}[]}]
99 \def\@providesfile#1[#2]{%
100 \wlog{File: #1 #2}%
101 \@addtofilelist{ #2}%
102 \endgroup}

```

```

103 \long\def\@addtofilelist#1{}
104 \def\@empty{}
105 \catcode'\%=12
106 \def\@percentchar{%}
107 \catcode'\%=14
108 \let\@currdir\@undefined
109 \let\input@path\@undefined
110 \let\filename@parse\@undefined

\strip@prefix

111 \def\strip@prefix#1>{}
112 </2ekernel>

(End definition for \strip@prefix.)

```

3 texsys.cfg

As mentioned above, any site specific definitions required to describe the filename handling must be entered into a file `texsys.cfg`. If `texsys.cfg` can not be located by `\openin`, we write a default version out. The default version only contains comments, so we do not actually input the file in that case. The automatic tests later will, hopefully, correctly define the required macros.

The tricky code below checks to see if `texsys.cfg` exists. If it does not, all the text in this file between `START` and `END` is copied verbatim to a new file `texsys.cfg`. If `texsys.cfg` is found, then it is simply input. This is only done when this file is being used unstripped.

```

113 <*docstrip>
114 \openin15=texsys.cfg
115 \ifeof15
116 \typeout{** Writing a default texsys.cfg}
117 \immediate\openout15=texsys.cfg
118 \begingroup
119 \catcode'\^M\active%
120 \let^M\par%
121 \def\reserved@a#1^M{%
122   \def\reserved@b{#1}%
123   \ifx\reserved@b\reserved@c\endgroup\else%
124     \immediate\write15{#1}%
125     \expandafter\reserved@a\fi}%
126 \def\reserved@d#1START^M{\let\do\@makeother\dospecials\reserved@a}%
127 \catcode'\%=12
128 \def\reserved@c{END}
129 \reserved@d

```

START

3.1 texsys.cfg

This file contains the site specific definitions of the four macros `\@currdir`, `\input@path`, `\filename@parse` and `\@TeXversion`.

As distributed it only contains comments, however this 'empty' file will work on many systems because of the automatic tests built into `ltdirchk.dtx`. You *are* allowed to edit this file to add definitions of these macros appropriate to your system.

The macros that must be defined are:

`\@currdir` `\@currdir{filename}<space>` should expand to a form of the filename that uniquely refers to the ‘current directory’ if this is possible. (The expansion should also end with a space.) on UNIX, this is `\def\@currdir{./}`. For more exotic operating systems you may want to make `\@currdir` a macro with arguments delimited by `.` and/or `<space>`. If the operating system has no concept of directory structure, this macro should be defined to be empty.

`\input@path` If the primitive `\openin` searches the same directories as the primitive `\input`, then it is possible to tell (using `\ifeof`) whether a file exists before trying to input it. For systems like this, `\input@path` should be left undefined.

If `\openin` does not ‘follow’ `\input` then `\input@path` must be defined to be a list of directories to search for input files. The format for each directory is as for `\@currdir`, normally just a prefix is required, but it may be a macro with space-delimited argument. That is, if `<dir>` is an entry in the input path, `TeX` will try to load the expansion of

`<dir><filename><space>`

So either `<dir>` should be defined as a macro with argument delimited by space, or it should just expand to a directory name, including the final directory separator, so that it may be concatenated with the `<filename>`. This means that for UNIX-like syntax, each `<dir>` should end with a slash, `/`. One exception to this rule is that the input path should *always* contain the empty directory `{}` as this will allow ‘full pathnames’ to be used, and the ‘current directory’ to be searched.

`\input@path` should expand to a list of such directories, each in a `{}` group.

`\filename@parse` After a call of the form: `\filename@parse{<filename>}`, the three macros `\filename@area`, `\filename@base`, `\filename@ext` should be defined to be the ‘area’ (or directory), basename and extension respectively. If there was no extension specified in `<filename>`, `\filename@ext` should be `\let` to `\relax` (so this case may be tested with `\@ifundefined{filename@ext}` and, perhaps a default extension substituted).

Normally one would not need to define this macro in `texsys.cfg` as the automatic tests can supply parsers that work with UNIX and VMS syntax, as well as a basic parser that will cover many other cases. However some operating systems may need a ‘hand produced’ parser in which case it should be defined in this file.

The UNIX parser also works for most MSDOS `TeX` versions. Currently if the UNIX or VMS parser is not used, `\filename@parse` is defined to always return an empty area, and to split the argument into basename and extension at the first ‘.’ that occurs in the name. Parsers for other formats may be defined in `texsys.cfg`, in which case they will be used in preference to the default definitions.

`\@TeXversion` You should not need to set this macro in `texsys.cfg`. `LaTeX` tests to set this automatically. See the comments in the opening section of `ltdirchk.dtx`.

The following sections give examples of definitions which might work on various systems. These are currently mainly untested as I only have access to a few systems, all of which do not need this file as the automatic tests work. All the code is commented out.

3.2 UNIX (web2c)

This implementation does make `\openin` and `\input` look in the same places. Acceptable settings are made by `ltdirchk.dtx`, and so this file may be empty. The definitions below are therefore just for information.

130 `%\def\@currdir{./}`

131 `%\let\input@path\@undefined`

3.3 UNIX (other)

Apparently some commercial UNIX implementations have different paths for `\openin` and `\input`. For these one could use definitions like the following (with whatever directories are used at your site): note that the directory names should end with `/`.

```
132 % \def\currdir{./}
133 % \def\input@path{%
134 %   {/usr/local/lib/tex/inputs/distrib/}%
135 %   {/usr/local/lib/tex/inputs/contrib/}%
136 %   {/usr/local/lib/tex/inputs/local/}%
137 % }
```

3.4 MSDOS (emtex)

This implementation does make `\openin` and `\input` look in the same places. Acceptable settings are made by `ltdirchk.dtx`, and so this file may be empty. The definitions below are therefore just for information.

```
138 % \def\currdir{./}
139 % \let\input@path\undefined
```

3.5 MSDOS (other)

Some PC implementations have different paths for `\openin` and `\input`. For these one could use definitions like the following (with whatever directories are used at your site): note that the directory names should end with `/`. This assumes the implementation uses UNIX style `/` as the directory separator.

```
140 % \def\currdir{./}
141 % \def\input@path{%
142 %   {c:/tex/inputs/distrib/}%
143 %   {c:/tex/inputs/contrib/}%
144 %   {c:/tex/inputs/local/}%
145 % }
```

3.6 VMS (DECUS T_EX, PD VMS 3.6)

This implementation does make `\openin` and `\input` look in the same places. Acceptable settings are made by `ltdirchk.dtx`, and so this file may be empty. The definitions below are therefore just for information.

```
146 % \def\currdir{[] }
147 % \let\input@path\undefined
```

3.7 VMS (???)

Some VMS implementations have different paths for `\openin` and `\input`. For these one could use definitions like the following:

```
148 % \def\currdir{[] }
149 % \def\input@path{%
150 %   {tex_inputs:}%
151 %   {SOMEDISK:[SOME.TEXT.DIRECTORY]}%
152 % }
```

3.8 MACINTOSH (OzTeX 1.6)

This implementation does make `\openin` and `\input` look in the same places. Acceptable settings are made by `ltdirchk.dtx`, and so this file may be empty. The definitions below are therefore just for information.

```
153 % \def\@currdir{:}
154 % \let\input@path\@undefined
```

3.9 MACINTOSH (other)

Some Macintosh implementations have different paths for `\openin` and `\input`. For these one could use definitions like the following (with whatever folders are used on your machine): note that the directory names should end with `:`, and they should contain *no* spaces.

```
155 % \def\@currdir{:}
156 % \def\input@path{%
157 %   {Hard-Disk:Applications:TeX:TeX-inputs:}%
158 %   {Hard-Disk:Applications:TeX:My-inputs:}%
159 % }
```

3.10 FAKE EXAMPLE

This example is for an operating system that has filenames of the form `<area>name`. For maximum compatibility with macro sets, you want `name.ext` to be mapped to `<ext>name`, and `<area>name.ext` to be mapped to `<area.ext>name`. `\input` does this mapping automatically, but `\openin` does not, and does not look in the same places as `\input`. `<>name` is the desired ‘current directory’ syntax.

the following code would possibly work:

```
160 % \def\@dir#1#2 {%
161 %   \@d@r{#1}#2..\@nil}
162 % \def\@d@r#1#2.#3.#4\@nil{%
163 %   <\ifx\@dir#1\@dir\else#1\ifx\@dir#3\@dir\else.\fi\fi#3>#2 }
164 %
165 % \def\@currdir{\@dir{}}
166 % \def\input@path{%
167 %   {\@dir{area.one}}}%
168 %   {\@dir{area.two}}}%
169 % }
```

END

```
170 \immediate\closeout15
```

If `texsys.cfg` did exist, then input it.

```
171 \else
172 \typeout{** Using the existing texsys.cfg}
173 \closein15
174 \input texsys.cfg
175 \fi
176 </docstrip>
```

If the stripped version of this file is being used (in `latex2e.ltx`) then `texsys.cfg` should be there, so just input it.

```
177 <dircheck>\input texsys.cfg
```

4 Setting \@currdir

`\@currdir` This is a local definition of `\IfFileExists`. It tries to relocate `texsys.aux`. If it succeeds, then the `\@currdir` syntax has been determined. If all the tests fail then `\@currdir` will be set to `\@empty`, and `ltxcheck` will warn of this when it checks the format.

```
178 \begingroup
179 \count@ \time
180 \divide \count@ 60
181 \count2 = - \count@
182 \multiply \count2 60
183 \advance \count2 \time
```

The current date and time stamp.

```
184 \edef \today {%
\today 185 \the \year / \two@digits { \the \month } / \two@digits { \the \day } : %
186 \two@digits { \the \count@ } : \two@digits { \the \count2 }}
```

Create a file `texsys.aux` (hopefully in the current directory), then try to locate it again.

```
187 \immediate \openout15 = texsys.aux
188 \immediate \write15 { \today ^^J }
189 \immediate \closeout15 %
```

`#1` is the file to try, `#2` is what to do on success, `#3` on failure. Note that this definition is overwritten later on again!

```
190 \def \IfFileExists #1#2#3 { %
191 \openin \@inputcheck #1 %
192 \ifeof \@inputcheck
193 #3 \relax
194 \else
195 \read \@inputcheck to \reserved@a
196 \ifx \reserved@a \today
197 \typeout { #1 found } #2 \relax
198 \else
199 \typeout { BAD: old file \reserved@a (should be \today) } %
200 #3 \relax
201 \fi
202 \fi
203 \closein \@inputcheck }
204 \endlinechar = -1
```

If `\@currdir` has not been pre-defined in `texsys.cfg` then test for UNIX, VMS and Oz-TeX-Mac. syntax.

```
205 \ifx \@currdir \@undefined
206 \IfFileExists { ./texsys.aux } { \gdef \@currdir { ./ } } %
207 { \IfFileExists { [] texsys.aux } { \gdef \@currdir { [] } } %
208 { \IfFileExists { : texsys.aux } { \gdef \@currdir { : } } { } }
```

If it is still undefined at this point, all the above tests failed. Earlier versions interactively prompted for a definition at this point, but it seems impossible to reliably obtain information from users at this point in the installation. This version of the file produces

a format with no user-interaction. Later if the format is not suitable for the system, `texsys.cfg` may be edited and the format re-made.

```

209 \ifx\@currdir\undefined
210 \global\let\@currdir\empty
211 \typeout{^^J^^J%
212     !! No syntax for the current directory could be found^^J%
213 }%
214 \fi

```

Otherwise `\@currdir` was defined in `texsys.cfg`. In this case check that the syntax specified works on this system. (In case a complete \LaTeX system has been copied from one system to another.) If the test fails, give up. The installer should remove or correct the offending `texsys.cfg` and try again.

```

215 \else
216 \IfFileExists{\@currdir texsys.aux}{\{%
217 \edef\reserved@a{\errhelp{%
218 texsys.cfg specifies the current directory syntax to be^^J%
219 \meaning\@currdir^^J%
220 but this does not work on this system.^^J%
221 Remove texsys.cfg and restart.}}\reserved@a
222 \errmessage{Bad texsys.cfg file: \noexpand\@currdir}\@@end}

```

The version of `\@currdir` in `texsys.cfg` looks OK.

```

223 \fi

224 \immediate\closeout15 %
225 \endgroup

226 \typeout{^^J^^J%
227     \noexpand\@currdir set to:
228     \expandafter\strip@prefix\meaning\@currdir.^^J%
229 }

```

(End definition for \@currdir, \IfFileExists, and \today.)

Stop here if the file is being used unstripped.

```

230 \<docstrip>
231 \relax\endinput
232 \</docstrip>

```

5 Setting `\input@path`

Earlier versions of this file attempted to automatically test whether `\input@path` was required, and interactively prompt for a path if necessary. This was not found to be very reliable. The first-time installer of $\text{\LaTeX} 2_{\epsilon}$ can not be expected to have enough information to supply the correct information to the prompts. Now the interaction is omitted. After the format is made the installer can attempt to run the test document `ltxcheck.tex` through $\text{\LaTeX} 2_{\epsilon}$. This will check, among other things, whether `texsys.cfg` will need to be edited and the format remade.

`\input@path` Now set up the `\input@path`.

`\input@path` should either be undefined, or a list of directories as described in the introduction.

```

233 \typeout{^^J%

```

```

234     Assuming \noexpand\openin and \noexpand\input^^J%
235     \ifx\input@path\@undefined
\input@path has not been pre-defined.
236         have the same search path.^^J%
237     \else
\input@path has been defined in texsys.cfg.
238         have different search paths.^^J%
239         LaTeX will use the path specified by \noexpand\input@path:^^J%
240     \fi
241 }

(End definition for \input@path.)

```

6 Filename Parsing

`\filename@parse` Split a filename into its components.

```

242 \ifx\filename@parse\@undefined
243     \def\reserved@a{.}\ifx\@currdir\reserved@a
\filename@parse was not specified in texsys.cfg, but \@currdir looks like UNIX...
244     \typeout{^^JDefining UNIX/DOS style filename parser.^^J}
245     \def\filename@parse#1{%
246         \let\filename@area\@empty
247         \expandafter\filename@path#1\\}

Search for the last /.

248     \def\filename@path#1/#2\\{%
249         \ifx\\#2\\%
250             \def\reserved@a{\filename@simple#1.\\}%
251         \else
252             \edef\filename@area{\filename@area#1/}%
253             \def\reserved@a{\filename@path#2\\}%
254         \fi
255         \reserved@a}

256 \else\def\reserved@a{[]}\ifx\@currdir\reserved@a
\filename@parse was not specified in texsys.cfg, but \@currdir looks like VMS...
257     \typeout{^^JDefining VMS style filename parser.^^J}
258     \def\filename@parse#1{%
259         \let\filename@area\@empty
260         \expandafter\filename@path#1\\}

Search for the last ].

261     \def\filename@path#1]#2\\{%
262         \ifx\\#2\\%
263             \def\reserved@a{\filename@simple#1.\\}%
264         \else
265             \edef\filename@area{\filename@area#1]}%
266             \def\reserved@a{\filename@path#2\\}%
267         \fi
268         \reserved@a}

269 \else\def\reserved@a{:}\ifx\@currdir\reserved@a

```

\filename@parse was not specified in texsys.cfg, but \@currdir looks like Macintosh...

```

270 \typeout{^^JDefining Mac style filename parser.^^J}
271 \def\filename@parse#1{%
272   \let\filename@area\@empty
273   \expandafter\filename@path#1:\}

```

Search for the last :.

```

274 \def\filename@path#1:#2\{%
275   \ifx\#2\%
276     \def\reserved@a{\filename@simple#1.\}%
277   \else
278     \edef\filename@area{\filename@area#1:}%
279     \def\reserved@a{\filename@path#2\}%
280   \fi
281   \reserved@a}
282 \else

```

\filename@parse was not specified in texsys.cfg. So just make a simple parser that always sets \filename@area to empty.

```

283 \typeout{^^JDefining generic filename parser.^^J}
284 \def\filename@parse#1{%
285   \let\filename@area\@empty
286   \expandafter\filename@simple#1.\}
287 \fi\fi\fi

```

\filename@simple is used by all three versions. Finally we can split off the extension.

```

288 </dircheck>
289 < *dircheck, latexrelease>
290 < latexrelease> \IncludeInRelease{2019/10/01}{\filename@simple}
291 < latexrelease> {Final dot for extension}%
292 \def\filename@simple#1.#2\{%
293   \ifx\#2\%
294     \let\filename@ext\relax
295     \edef\filename@base{#1}%
296   \else
297     \filename@dots{#1}#2\%
298   \fi}
299 \def\filename@dots#1#2.#3\{%
300   \ifx\#3\%
301     \def\filename@ext{#2}%
302     \edef\filename@base{#1}%
303   \else
304     \filename@dots{#1.#2}#3\%
305   \fi}
306 < latexrelease> \EndIncludeInRelease
307 < latexrelease> \IncludeInRelease{0000/00/00}{\filename@simple}
308 < latexrelease> {Final dot for extension}%
309 < latexrelease> \def\filename@simple#1.#2\{%
310 < latexrelease>   \ifx\#2\%
311 < latexrelease>     \let\filename@ext\relax
312 < latexrelease>   \else
313 < latexrelease>     \edef\filename@ext{\filename@dot#2\}%

```

```

314 <latexrelease>      \fi
315 <latexrelease>      \edef\filename@base{#1}}
316 <latexrelease>\EndIncludeInRelease
317 </dircheck, latexrelease>
318 <*dircheck>

      Remove a final dot, added earlier.
319   \def\filename@dot#1.\{#1}

320 \else

Otherwise, \filename@parse was specified in texsys.cfg.
321   \typeout{^^J^^J%
322     \noexpand\filename@parse was defined in texsys.cfg:^^J%
323     \expandafter\strip@prefix\meaning\filename@parse.^^J%
324   }
325 \fi

(End definition for \filename@parse.)

```

7 T_EX Versions

`\@TeXversion` T_EX versions older than 3.141 require `\@TeXversion` to be set. This can be determined automatically due to a trick suggested by Bernd Raichle. (Actually this will not always get the correct version number, eg T_EX3.14 would be detected as T_EX3, but L^AT_EX only needs to take account of T_EX's older than 3, or between 3 and 3.14.

```

326 \ifx\@TeXversion\undefined
327   \ifx\undefined\inputlineno
328     \def\@TeXversion{2}
329   \else
330     {\catcode'\^^J=\active
331      \def\reserved@a#1#2\@@{\if#1\string^3\fi}
332      \edef\reserved@a{\expandafter\reserved@a\string^^J\@@}
333      \ifx\reserved@a\empty\else\gdef\@TeXversion{3}\fi}
334   \fi
335 \fi

(End definition for \@TeXversion.)

336 </dircheck>

```

8 ltxcheck.tex

After the format has been made, and `article.cls` moved with the other files to the ‘standard input directory’ as specified in `install.txt`, the format may be checked by running the file `ltxcheck.tex`.

File b

lplain.dtx

1 Plain T_EX

L^AT_EX includes almost all of the functionality of Knuth's original 'Basic Macros' That is, the plain T_EX format described in Appendix B of the T_EXBook. However, some of the user commands are not much use so, in order to save memory, we may remove them from the kernel into a package. Here is a list of the commands that may be removed (PROBABLY NOT COMPLETE).

```
\magstep      \magstephalf
\mathhexbox
\vglue        \vgl@
\hglue        \hgl@
```

This file is by now very small as most of it has been moved to more appropriate kernel files: it may disappear completely one day.

L^AT_EX font definitions are done using NFSS2 so none of PLAIN's font definitions are in L^AT_EX.

L^AT_EX has its own tabbing environment, so PLAIN's is disabled.

L^AT_EX uses its own output routine, so most of the plain one was removed.

```
1 (*2kernel)
2 \catcode'\{=1 % left brace is begin-group character
3 \catcode'\}=2 % right brace is end-group character
4 \catcode'\$=3 % dollar sign is math shift
5 \catcode'\&=4 % ampersand is alignment tab
6 \catcode'\#=6 % hash mark is macro parameter character
7 \catcode'\^=7 % circumflex and uparrow are for superscripts
8 \catcode'\_ =8 % underline and downarrow are for subscripts
9 \catcode'\^^I=10 % ascii tab is a blank space
10 \chardef\active=13 \catcode'\~=\active % tilde is active
11 \catcode'\^^L=\active \def^^L{\par}% ascii form-feed is \par
12 \message{catcodes,}
```

We had to define the \catcodes right away, before the message line, since \message uses the { and } characters. When INITEX (the T_EX initializer) starts up, it has defined the following \catcode values:

```
\catcode'\^^@=9 %  ascii null is ignored
\catcode'\^^M=5 %  ascii return is end-line
\catcode'\ =0 %    backslash is TeX escape character
\catcode'\%=14 %   percent sign is comment character
\catcode'\ =10 %   ascii space is blank space
\catcode'\^^?=15 %  ascii delete is invalid
\catcode'\A=11 ... \catcode'\Z=11 % uppercase letters
\catcode'\a=11 ... \catcode'\z=11 % lowercase letters
all others are type 12 (other)
```

Here is a list of the characters that have been specially catcoded:

```
13 \def\dospecials{\do\ \do\\\do\{\do\}\do\$ \do\&%
14 \do\# \do\^ \do\_ \do\% \do\~}
```

(not counting ascii null, tab, linefeed, formfeed, return, delete) Each symbol in the list is preceded by , which can be defined if you want to do something to every item in the list.

We make @ signs act like letters, temporarily, to avoid conflict between user names and internal control sequences of plain format.

```
15 \catcode'\@=11
```

To make the plain macros more efficient in time and space, several constant values are declared here as control sequences. If they were changed, anything could happen; so they are private symbols.

`\@ne` Small constants are defined using `\chardef`.

```
\tw@ 16 \chardef\@ne=1
\thr@@ 17 \chardef\tw@=2
\sixt@@n 18 \chardef\thr@@=3
\@cclv 19 \chardef\sixt@@n=16
        20 \chardef\@cclv=255
```

(End definition for \@ne and others.)

`\@ccclvi` Constants above 255 defined using `\mathchardef`.

```
\@m 21 \mathchardef\@ccclvi=256
\@M 22 \mathchardef\@m=1000
\@MM 23 \mathchardef\@M=10000
      24 \mathchardef\@MM=20000
```

(End definition for \@ccclvi and others.)

Allocation of registers

Here are macros for the automatic allocation of `\count`, `\box`, `\dimen`, `\skip`, `\muskip`, and `\toks` registers, as well as `\read` and `\write` stream numbers, `\fam` codes, `\language` codes, and `\insert` numbers.

```
25 \message{registers,}
```

When a register is used only temporarily, it need not be allocated; grouping can be used, making the value previously in the register return after the close of the group. The main use of these macros is for registers that are defined by one macro and used by others, possibly at different nesting levels. All such registers should be defined through these macros; otherwise conflicts may occur, especially when two or more macro packages are being used at the same time.

Historical L^AT_EX 2.09 comments (not necessarily accurate any more):

The following counters are reserved:

- 0 to 9 page numbering
- 10 count allocation
- 11 dimen allocation
- 12 skip allocation
- 13 muskip allocation
- 14 box allocation
- 15 toks allocation
- 16 read file allocation
- 17 write file allocation
- 18 math family allocation
- 19 language allocation
- 20 insert allocation
- 21 the most recently allocated number

22 constant -1

End of historical L^AT_EX 2.09 comments.

New counters are allocated starting with 23, 24, etc. Other registers are allocated starting with 10. This leaves 0 through 9 for the user to play with safely, except that counts 0 to 9 are considered to be the page and subpage numbers (since they are displayed during output). In this scheme, `\count 10` always contains the number of the highest-numbered counter that has been allocated, `\count 14` the highest-numbered box, etc. Inserts are given numbers 254, 253, etc., since they require a `\count`, `\dimen`, `\skip`, and `\box` all with the same number; `\count 20` contains the lowest-numbered insert that has been allocated. Of course, `\box255` is reserved for `\output`; `\count255`, `\dimen255`, and `\skip255` can be used freely.

It is recommended that macro designers always use `\global` assignments with respect to registers numbered

1, 3, 5, 7, 9,

and always non-`\global` assignments with respect to registers

0, 2, 4, 6, 8, 255.

This will prevent “save stack buildup” that might otherwise occur.

```
26 \count10=22 % allocates \count registers 23, 24, ...
27 \count11=9 % allocates \dimen registers 10, 11, ...
28 \count12=9 % allocates \skip registers 10, 11, ...
29 \count13=9 % allocates \muskip registers 10, 11, ...
30 \count14=9 % allocates \box registers 10, 11, ...
31 \count15=9 % allocates \toks registers 10, 11, ...
32 \count16=-1 % allocates input streams 0, 1, ...
33 \count17=-1 % allocates output streams 0, 1, ...
34 \count18=3 % allocates math families 4, 5, ...
35 \count19=0 % allocates \language codes 1, 2, ...
36 \count20=255 % allocates insertions 254, 253, ...
```

```
\insc@unt The insertion counter and most recent allocation.
\allocationnumber
37 \countdef\insc@unt=20
38 \countdef\allocationnumber=21
```

(End definition for \insc@unt and \allocationnumber.)

```
\m@ne The constant -1.
39 \countdef\m@ne=22 \m@ne=-1
```

(End definition for \m@ne.)

```
\wlog Write on log file (only)
40 \def\wlog{\immediate\write\m@ne}
```

(End definition for \wlog.)

```
\count@ Here are abbreviations for the names of scratch registers that don't need to be allocated.
\dimen@ 41 \countdef\count@=255
\dimen@i 42 \dimendef\dimen@=0
\dimen@ii 43 \dimendef\dimen@i=1 % global only
\skip@ 44 \dimendef\dimen@ii=2
\toks@ 45 \skipdef\skip@=0
46 \toksdef\toks@=0
```

(End definition for \count@ and others.)

```

\newcount Now, we define \newcount, \newbox, etc. so that you can say \newcount\foo and \foo
\newdimen will be defined (with \countdef) to be the next counter.
\newskip To find out which counter \foo is, you can look at \allocationnumber.
\newmuskip Since there's no \boxdef command, \chardef is used to define a \newbox,
\newbox \newinsert, \newfam, and so on.
\newtoks LATEX change: remove \outer from \newcount and \newdimen (FMi) This is nec-
\newread essary to use \newcount inside \if... later on. Also remove from \newskip, \newbox
\newwrite \newwrite and \newfam (DPC) to save later redefinition.
\newfam
\newlanguage
47 </2ekernel>
48 <*2ekernel|latexrelease>
49 <latexrelease>\IncludeInRelease{2015/01/01}%
50 <latexrelease> {\newcount}{Extended Allocation}%

51 \def\newcount {\e@alloc\count \countdef {\count10}\insecunt\float@count}
52 \def\newdimen {\e@alloc\dimen \dimendef {\count11}\insecunt\float@count}
53 \def\newskip {\e@alloc\skip \skipdef {\count12}\insecunt\float@count}
54 \def\newmuskip
55 {\e@alloc\muskip\muskipdef{\count13}\m@ne\e@alloc@top}

For compatibility use \chardef in the classical range.

56 \def\newbox {\e@alloc\box
57 {\ifnum\allocationnumber<\cclvi
58 \expandafter\chardef
59 \else
60 \expandafter\e@alloc@chardef
61 \fi}
62 {\count14}\insecunt\float@count}
63 \def\newtoks {\e@alloc\toks \toksdef{\count15}\m@ne\e@alloc@top}
64 \def\newread {\e@alloc\read \chardef{\count16}\m@ne\sixt@n}

Skip \write18 due to its traditional use as a shell-escape.

65 \ifx\directlua\@undefined
66 \def\newwrite {\e@alloc\write \chardef{\count17}\m@ne\sixt@n}
67 \else
68 \def\newwrite {\e@alloc\write
69 {\ifnum\allocationnumber=18
70 \advance\count17\@ne
71 \allocationnumber\count17 %
72 \fi
73 \global\chardef}%
74 {\count17}%
75 \m@ne
76 {128}}
77 \fi

78 \def\new@mathgroup
79 {\e@alloc\mathgroup\chardef{\count18}\m@ne\e@mathgroup@top}
80 \let\newfam\new@mathgroup

81 \ifx\directlua\@undefined
82 \def\newlanguage {\e@alloc\language \chardef{\count19}\m@ne\cclvi}
83 \else
84 \def\newlanguage {\e@alloc\language \chardef{\count19}\m@ne{16384}}

```



```

85 \fi
86 </2ekernel | latexrelease>
87 <latexrelease>\EndIncludeInRelease
88 <latexrelease>\IncludeInRelease{0000/00/00}%
89 <latexrelease>          {\newcount}{Extended Allocation}%
90 <latexrelease>\def\newcount{\alloc@0\count\countdef\insc@unt}
91 <latexrelease>\def\newdimen{\alloc@1\dimen\dimendef\insc@unt}
92 <latexrelease>\def\newskip{\alloc@2\skip\skipdef\insc@unt}
93 <latexrelease>\def\newmuskip{\alloc@3\muskip\muskipdef\@ccclvi}
94 <latexrelease>\def\newbox{\alloc@4\box\chardef\insc@unt}
95 <latexrelease>\def\newtoks{\alloc@5\toks\toksdef\@ccclvi}
96 <latexrelease>\def\newread{\alloc@6\read\chardef\sixt@@n}
97 <latexrelease>\def\newwrite{\alloc@7\write\chardef\sixt@@n}
98 <latexrelease>\def\new@mathgroup{\alloc@8\fam\chardef\sixt@@n}
99 <latexrelease>\def\newlanguage{\alloc@9\language\chardef\@ccclvi}
100 <latexrelease>\let\newfam\new@mathgroup
101 <latexrelease>\EndIncludeInRelease

```

(End definition for \newcount and others.)

\e@alloc@chardef The upper limit of extended registers, which leaves this number (eg \dimen32767) always
 \e@alloc@top unallocated by these macros. cf traditional \dimen255.

```

102 <*2ekernel | latexrelease>
103 <latexrelease>\IncludeInRelease{2015/01/01}%
104 <latexrelease>          {\e@alloc@chardef}{Extended Allocation}%
105 \ifx\directlua\@undefined
106   \ifx\widowpenalties\@undefined
107     \mathchardef\e@alloc@top=255
108     \let\e@alloc@chardef\chardef
109   \else
110     \mathchardef\e@alloc@top=32767
111     \let\e@alloc@chardef\mathchardef
112   \fi
113 \else

```

classic tex has 2⁸ registers.

etex and xetex have 2¹⁵ registers.

luatex has 2¹⁶ registers.

```

114   \chardef\e@alloc@top=65535
115   \let\e@alloc@chardef\chardef
116 \fi
117 </2ekernel | latexrelease>
118 <latexrelease>\EndIncludeInRelease
119 <latexrelease>\IncludeInRelease{0000/00/00}%
120 <latexrelease>          {\e@alloc@chardef}{Extended Allocation}%
121 <latexrelease>\let\e@alloc@top\@undefined
122 <latexrelease>\let\e@alloc@chardef\@undefined
123 <latexrelease>\EndIncludeInRelease

```

(End definition for \e@alloc@chardef and \e@alloc@top.)

`\e@mathgroup@top` The upper limit of extended math groups (`\fam`) 16 in classic T_EX and e-T_EX, but 256 in Unicode TeX variants.

```

124 <*2ekernel | latexrelease>
125 <latexrelease>\IncludeInRelease{2015/01/01}%
126 <latexrelease>                {\e@mathgroup@top}{Extended Allocation}%
127 \ifx\Umathcode\@undefined
classic and e tex have 16 fam (0–15).
128   \chardef\e@mathgroup@top=16
129   \else
xetex and luatex have 256 fam (0–255).
130   \chardef\e@mathgroup@top=256
131   \fi
132 </2ekernel | latexrelease>
133 <latexrelease>\EndIncludeInRelease
134 <latexrelease>\IncludeInRelease{0000/00/00}%
135 <latexrelease>                {\e@mathgroup@top}{Extended Allocation}%
136 <latexrelease>\let\e@mathgroup@top\@undefined
137 <latexrelease>\EndIncludeInRelease

```

(End definition for \e@mathgroup@top.)

`\e@alloc` A modified version of `\alloc@` that takes the count register rather than just the final digit of its number (assuming `\count1x`). It also has an extra argument to give the top of the extended range.

```

#1 #2 #3 #4 #5 #6
\e@alloc type defcmd current top extended-top newname
Note that if just a single allocation range is required (not omitting a range up to
255 for inserts) then −1 should be used for the first upper bound argument, #4.
138 <*2ekernel | latexrelease>
139 <latexrelease>\IncludeInRelease{2015/01/01}{\e@alloc}{Extended Allocation}%
140 \def\e@alloc#1#2#3#4#5#6{%
141   \global\advance#3\@ne
142   \e@ch@ck{#3}{#4}{#5}#1%
143   \allocationnumber#3\relax
144   \global#2#6\allocationnumber
145   \wlog{\string#6=\string#1\the\allocationnumber}}%
146 </2ekernel | latexrelease>
147 <latexrelease>\EndIncludeInRelease
148 <latexrelease>\IncludeInRelease{0000/00/00}{\e@alloc}{Extended Allocation}%
149 <latexrelease>\let\e@alloc\@undefined
150 <latexrelease>\EndIncludeInRelease
151 <*2ekernel>

```

(End definition for \e@alloc.)

`\e@ch@ck` Extended check command. If the first range is exceeded, bump to 256 (or 266 for counts) and try again, testing the extended range.

Allocate matching registers from the top of the extended range and add to \@freelist.

```
\extrafloats 152 </2kernel>
153 <*2kernel | latexrelease>
154 <latexrelease> \IncludeInRelease{2015/10/01}
155 <latexrelease> { \e@ch@ck } { Extended Allocation (checking) } %
156 \gdef \e@ch@ck#1#2#3#4{%
157   \ifnum#1<#2\else
```

If we've reached the classical top limit, bump to 256 or 266 for counts (count 256–265 are reserved by the allocation system).

```
158   \ifnum#1=#2\relax
159     \global#1\@cclvi
160     \ifx\count#4\global\advance#1 10 \fi
161   \fi
```

Check we are below the extended limit.

```
162   \ifnum#1<#3\relax
163   \else
164     \errmessage{No room for a new \string#4}%
165   \fi
166 \fi}%
167 <latexrelease> \EndIncludeInRelease
168 <latexrelease> \IncludeInRelease{2015/01/01}%
169 <latexrelease> { \e@ch@ck } { Extended Allocation (checking) } %
170 <latexrelease> \gdef \e@ch@ck#1#2#3#4{%
171 <latexrelease>   \ifnum#1<#2\else
172 <latexrelease>     \ifnum#1=#2\relax
173 <latexrelease>       #1\@cclvi
174 <latexrelease>       \ifx\count#4\advance#1 10 \fi
175 <latexrelease>       \fi
176 <latexrelease>       \ifnum#1<#3\relax
177 <latexrelease>       \else
178 <latexrelease>         \errmessage{No room for a new #4}%
179 <latexrelease>         \fi
180 <latexrelease>       \fi}%
181 <latexrelease> \EndIncludeInRelease
182 <latexrelease> \IncludeInRelease{0000/00/00}%
183 <latexrelease> { \e@ch@ck } { Extended Allocation (checking) } %
184 <latexrelease> \let \e@ch@ck \@undefined
185 <latexrelease> \EndIncludeInRelease
186 <latexrelease> \IncludeInRelease{2015/01/01}%
187 <latexrelease> { \extrafloats } { Extra floats } %
188 \let \float@count \e@alloc@top
```

```
\extrafloats 189 \ifx\numexpr\@undefined
```

In classic TeX use \newinsert to allocate float boxes.

```
190 \def\extrafloats#1{%
191   \count@#1\relax
192   \ifnum\count@>\z@
193     \newinsert\reserved@a
194   \global\expandafter\chardef
```

```

195         \csname bx@\the\allocationnumber\endcsname\allocationnumber
196 \@cons\@freelist{\csname bx@\the\allocationnumber\endcsname}%
197 \advance\count@m@ne
198 \expandafter\extrafloats
199 \expandafter\count@
200 \fi
201 }%
202 \else

```

In e-tex take float boxes from the top of the extended range.

```

203 \def\extrafloats#1{%
204 \ifnum#1>\z@
205 \count@\numexpr\float@count-1\relax
206 \ch@ck0\count@\count
207 \ch@ck1\count@\dimen
208 \ch@ck2\count@\skip
209 \ch@ck4\count@\box
210 \global\@alloc@chardef\float@count\count@
211 \global\expandafter\@alloc@chardef
212         \csname bx@\the\float@count\endcsname\float@count
213 \@cons\@freelist{\csname bx@\the\float@count\endcsname}%
214 \expandafter
215 \extrafloats\expandafter{\numexpr#1-1\relax}%
216 \fi}%
217 \fi
218 </2ekernel | latexrelease>
219 <latexrelease>\EndIncludeInRelease
220 <latexrelease>\IncludeInRelease{0000/00/00}%
221 <latexrelease>         {\extrafloats}{Extra floats}%
222 <latexrelease>\let\float@count\@undefined
223 <latexrelease>\let\extrafloats\@undefined
224 <latexrelease>\EndIncludeInRelease
225 <*2ekernel>

```

(End definition for \e@ch@ck, \extrafloats, and \extrafloats.)

\alloc@ Since \e@alloc was added in 2015, \@alloc has not been used, but was left as some legacy code calls it. However the original definition gives spurious errors once the “classic” registers run out, so it is now defined to call \e@alloc internally.

```

226 </2ekernel>
227 <*2ekernel | latexrelease>
228 <latexrelease>\IncludeInRelease{2020/10/01}
229 <latexrelease>         {\alloc@}{emulate alloc@}%
230 \def\alloc@#1#2#3#4{\e@alloc#2#3{\count1#1}#4\float@count}
231 </2ekernel | latexrelease>
232 <latexrelease>\EndIncludeInRelease
233 <latexrelease>\IncludeInRelease{0000/00/00}%
234 <latexrelease>         {\alloc@}{emulate alloc@}%
235 <latexrelease>\def\alloc@#1#2#3#4#5{\global\advance\count1#1\@ne
236 <latexrelease> \ch@ck#1#4#2%
237 <latexrelease> \allocationnumber\count1#1%
238 <latexrelease> \global#3#5\allocationnumber
239 <latexrelease> \wlog{\string#5=\string#2\the\allocationnumber}}

```

```

240 <latexrelease>\EndIncludeInRelease
241 <*2ekernel>

```

(End definition for \alloc@.)

\newinsert

```

242 </2ekernel>
243 <*2ekernel | latexrelease>
244 <latexrelease>\IncludeInRelease{2015/10/01}
245 <latexrelease>{\newinsert}{Extended \newinsert}%
246 \ifx\numexpr\@undefined

```

If e-TeX is not available use the original plain TeX definition of \newinsert.

```

247 \def\newinsert#1{\global\advance\insc@unt \m@ne
248 \ch@ck0\insc@unt\count
249 \ch@ck1\insc@unt\dimen
250 \ch@ck2\insc@unt\skip
251 \ch@ck4\insc@unt\box
252 \allocationnumber\insc@unt
253 \global\chardef#1\allocationnumber
254 \wlog{\string#1=\string\insert\the\allocationnumber}}
255 \else

```

The highest register allowed with \insert.

```

256 \ifx\directlua\@undefined
257 \chardef\@insert@top255
258 \else
259 \chardef\@insert@top\@alloc@top
260 \fi

```

If the classic registers are exhausted, take an insert from the free float list and use \extrafloats to add a new float to that list.

```

261 \def\newinsert#1{%
262 \@tempswafalse
263 \global\advance\insc@unt\m@ne
264 \ifnum\count10<\insc@unt
265 \ifnum\count11<\insc@unt
266 \ifnum\count12<\insc@unt
267 \ifnum\count14<\insc@unt
268 \@tempswatruetrue
269 \fi\fi\fi\fi
270 \if@tempswa
271 \allocationnumber\insc@unt
272 \else
273 \global\advance\insc@unt\@ne
274 \extrafloats\@ne
275 \@next\@currbox\@freelist
276 {\ifnum\@currbox<\@insert@top
277 \allocationnumber\@currbox
278 \else
279 \ch@ck0\m@ne\insert
280 \fi}%
281 {\ch@ck0\m@ne\insert}%
282 \fi

```

```

283 \global\chardef#1\allocationnumber
284 \wlog{\string#1=\string\insert\the\allocationnumber}%
285 }

286 \fi
287 </2ekernel | latexrelease>

288 <latexrelease>\EndIncludeInRelease
289 <latexrelease>\IncludeInRelease{0000/00/00}%
290 <latexrelease>{\newinsert}{Extended \newinsert}%
291 <latexrelease>\let\@insert@top\@undefined
292 <latexrelease>\def\newinsert#1{\global\advance\insc@unt \m@ne
293 <latexrelease> \ch@ck0\insc@unt\count
294 <latexrelease> \ch@ck1\insc@unt\dimen
295 <latexrelease> \ch@ck2\insc@unt\skip
296 <latexrelease> \ch@ck4\insc@unt\box
297 <latexrelease> \allocationnumber\insc@unt
298 <latexrelease> \global\chardef#1\allocationnumber
299 <latexrelease> \wlog{\string#1=\string\insert\the\allocationnumber}}
300 <latexrelease>\EndIncludeInRelease
301 <*2ekernel>

```

(End definition for \newinsert.)

\ch@ck

```

302 \gdef\ch@ck#1#2#3{%
303   \ifnum\count1#1<#2\else
304     \errmessage{No room for a new #3}%
305   \fi}

```

(End definition for \ch@ck.)

\newhelp

```

306 \def\newhelp#1#2{\newtoks#1#1\expandafter{\csname#2\endcsname}}

```

(End definition for \newhelp.)

\@inputcheck Allocate read stream for testing and output stream that is never open and thus writes to the terminal.

\@unused

```

307 \newread\@inputcheck
308 \newwrite\@unused

```

(End definition for \@inputcheck and \@unused.)

\maxdimen Here are some examples of allocation.

\hideskip

```

309 \newdimen\maxdimen \maxdimen=16383.999999pt % the largest legal <dimen>
310 \newskip\hideskip \hideskip=-1000pt plus 1fill % negative but can grow

```

(End definition for \maxdimen and \hideskip.)

\p@

\z@

\z@skip

\voidb@x

```

311 \newdimen\p@ \p@=1pt % this saves macro space and time
312 \newdimen\z@ \z@=0pt % can be used both for 0pt and 0
313 \newskip\z@skip \z@skip=0pt plus0pt minus0pt
314 \newbox\voidb@x % permanently void box register

```

(End definition for \p@ and others.)

Assign initial values to T_EX's parameters

```
315 \message{parameters,}
```

All of T_EX's numeric parameters are listed here, but the code is commented out if no special value needs to be set. INITEX makes all parameters zero except where noted. *Historical L^AT_EX 2.09 comments (not necessarily accurate any more):*

```
316 \pretolerance=100
317 \tolerance=200 % INITEX sets this to 10000
318 \hbadness=1000
319 \vbadness=1000
320 \linepenalty=10
321 \hyphenpenalty=50
322 \exhyphenpenalty=50
323 \binoppenalty=700
324 \relpenalty=500
325 \clubpenalty=150
326 \widowpenalty=150
327 \displaywidowpenalty=50
328 \brokenpenalty=100
329 \predisplaypenalty=10000

\postdisplaypenalty=0
\interlinepenalty=0
\floatingpenalty=0, set during \insert
\outputpenalty=0, set before TeX enters \output

330 \doublehyphendemerits=10000
331 \finalhyphendemerits=5000
332 \adjdemerits=10000

\looseness=0, cleared by TeX after each paragraph
\pausing=0
\holdinginserts=0
\tracingonline=0
\tracingmacros=0
\tracingstats=0
\tracingparagraphs=0
\tracingpages=0
\tracingoutput=0

333 \tracinglostchars=1

\tracingcommands=0
\tracingrestores=0
\end{macrocode}
```

```
\begin{macro}{\tracingstacklevels}
```

For Lua[®]TeX, the \cs{tracingstacklevels} functionality was implemented as a callback, so here we just define the count register to hold the value of the parameter.

```
334 \</2ekernel>
335 \<*2ekernel | latexrelease>
336 \<latexrelease>\IncludeInRelease{2021/06/01}{\tracingstacklevels}%
```

```

337 <latexrelease> {tracingstacklevels}%
338 \ifx\directlua\@undefined
339 % \tracingstacklevels=0 % added in 2021
340 \else
341 \newcount\tracingstacklevels
342 % Code for \tracingstacklevels defined in ltfinal.dtx
343 \fi
344 <latexrelease>\EndIncludeInRelease
345 <latexrelease>
346 <latexrelease>\IncludeInRelease{0000/00/00}{\tracingstacklevels}%
347 <latexrelease> {tracingstacklevels}%
348 <latexrelease>\ifx\directlua\@undefined
349 <latexrelease>\else
350 <latexrelease> \let\tracingstacklevels\@undefined
351 <latexrelease>\fi
352 <latexrelease>\EndIncludeInRelease
353 </2ekernel | latexrelease>
354 <*2ekernel>

\end{macro}

\language=0

355 \uchyph=1

\lefthyphenmin=2 \righthyphenmin=3 set below
\globaldefs=0
\maxdeadcycles=25 % INITEX does this
\hangafter=1 % INITEX does this, also TeX after each paragraph
\fam=0
\mag=1000 % INITEX does this
\escapechar='\' % INITEX does this

356 \defaultthyphenchar='\'
357 \defaultskewchar=-1

\endlinechar='^M % INITEX does this
\newlinechar=-1 \LaTeX\ sets this in ltdefs.dtx.

358 \delimiterfactor=901

\time=now % TeX does this at beginning of job
\day=now % TeX does this at beginning of job
\month=now % TeX does this at beginning of job
\year=now % TeX does this at beginning of job

End of historical LATEX 2.09 comments.

In LATEX we don't want box information in the transcript unless we do a full tracing.

359 \showboxbreadth=-1
360 \showboxdepth=-1
361 \errorcontextlines=-1

```



```

362 \hfuzz=0.1pt
363 \vfuzz=0.1pt
364 \overfullrule=5pt
365 \maxdepth=4pt
366 \splitmaxdepth=\maxdimen
367 \boxmaxdepth=\maxdimen

Historical LATEX 2.09 comments (not necessarily accurate any more):
\lineskiplimit=0pt, changed by \normalbaselines

368 \delimitershortfall=5pt
369 \nulldelimiterspace=1.2pt
370 \scriptspace=0.5pt

\mathsurround=0pt
\predisplaysize=0pt, set before TeX enters $$
\displaywidth=0pt, set before TeX enters $$
\displayindent=0pt, set before TeX enters $$

371 \parindent=20pt

\hangindent=0pt, zeroed by TeX after each paragraph
\hoffset=0pt
\voffset=0pt

```

```

\baselineskip=0pt, changed by \normalbaselines
\lineskip=0pt, changed by \normalbaselines

```

```

372 \parskip=0pt plus 1pt
373 \abovedisplayskip=12pt plus 3pt minus 9pt
374 \abovedisplayshortskip=0pt plus 3pt
375 \belowdisplayskip=12pt plus 3pt minus 9pt
376 \belowdisplayshortskip=7pt plus 3pt minus 4pt

```

```

\leftskip=0pt
\rightskip=0pt

377 \topskip=10pt
378 \splittopskip=10pt

```

```

\tabskip=0pt
\spaceskip=0pt
\xspaceskip=0pt

```

```

379 \parfillskip=0pt plus 1fil

```

End of historical L^AT_EX 2.09 comments.

```

\normalbaselineskip We also define special registers that function like parameters:
\normallineskip
\normallineskiplimit 380 \newskip\normalbaselineskip \normalbaselineskip=12pt
381 \newskip\normallineskip \normallineskip=1pt
382 \newdimen\normallineskiplimit \normallineskiplimit=0pt

```

(End definition for \normalbaselineskip, \normallineskip, and \normallineskiplimit.)

```

\interfootlinepenalty

```

```

383 \newcount\interfootnotelinepenalty \interfootnotelinepenalty=100

```

(End definition for \interfootlinepenalty.)

Definitions for preloaded fonts

\magstephalf

\magstep

```
384 \def\magstephalf{1095 }
385 \def\magstep#1{\ifcase#1 \@m\or 1200\or 1440\or 1728\or
386 2074\or 2488\fi\relax}
```

(End definition for \magstephalf and \magstep.)

Macros for setting ordinary text

\frenchspacing

\nonfrenchspacing

```
387 \def\frenchspacing{\sfcode'\.\@m \sfcode'\?\@m \sfcode'\!\@m
388 \sfcode'\:\@m \sfcode'\;\@m \sfcode'\,\@m}
389 \def\nonfrenchspacing{\sfcode'\.3000\sfcode'\?3000\sfcode'\!3000%
390 \sfcode'\:2000\sfcode'\;1500\sfcode'\,1250 }
```

(End definition for \frenchspacing and \nonfrenchspacing.)

\normalbaselines

```
391 \def\normalbaselines{\lineskip\normallineskip
392 \baselineskip\normalbaselineskip \lineskiplimit\normallineskiplimit}
```

(End definition for \normalbaselines.)

\M Save a bit of space by using \let here.

\I 393 \def\^^M{\ } % control <return> = control <space>

394 \let\^^I\^^M % same for <tab>

(End definition for \M and \I.)

\lq

\rq

```
395 \def\lq{' }
396 \def\rq{' }
```

(End definition for \lq and \rq.)

\lbrack

\rbrack

```
397 \def\lbrack{[ ]}
398 \def\rbrack{[ ]}
```

(End definition for \lbrack and \rbrack.)

\aa These are not from plain.tex but they are similar to other commands found here and

\AA nowhere else, being alternate input forms for characters.

```
399 \def \aa {\r a}
400 \def \AA {\r A}
```

(End definition for \aa and \AA.)

\endgraf

\endline

```
401 \let\endgraf=\par
402 \let\endline=\cr
```

(End definition for \endgraf and \endline. These functions are documented on page 268.)

`\space`

```
403 \def\space{ }
```

(End definition for `\space`.)

`\empty` This probably ought to go altogether, but let it to the L^AT_EX version to save space.

```
404 \let\empty\@empty
```

(End definition for `\empty`.)

`\null`

```
405 \def\null{\hbox{}}
```

(End definition for `\null`.)

`\bgroup`

`\egroup`

```
406 \let\bgroup={
```

```
407 \let\egroup=}
```

(End definition for `\bgroup` and `\egroup`.)

`\obeylines` In `\obeylines`, we say `\let^^M=\par` instead of `\def^^M{\par}` since this allows, for
`\obeyspaces` example, `\let\par=\cr \obeylines \halign{...`

```
408 {\catcode'\^^M=\active % these lines must end with %
409 \gdef\obeylines{\catcode'\^^M\active \let^^M\par}%
410 \global\let^^M\par} % this is in case ^^M appears in a \write
411 \def\obeyspaces{\catcode'\ \active}
412 {\obeyspaces\global\let \=\space}
```

(End definition for `\obeylines` and `\obeyspaces`.)

`\loop` We use Kabelschacht's method of doing loops, see TUB 8#2 (1987). (unless that breaks
`\iterate` something :-). It turned out to need an extra `\relax`: see pr/642 (`\loop` could do one
`\repeat` iteration too much in certain cases).

```
413 \long\def \loop #1\repeat{%
414 \def\iterate{#1\relax % Extra \relax
415 \expandafter\iterate\fi
416 }%
417 \iterate
418 \let\iterate\relax
419 }
```

This setting of `\repeat` is needed to make `\loop...\if...\repeat` skippable within
another `\if...`

```
420 \let\repeat=\fi
```

(End definition for `\loop`, `\iterate`, and `\repeat`.)

L^AT_EX defines `\smallskip`, etc. in `ltspace.dtx`.

`\nointerlineskip`

`\offinterlineskip`

```
421 \def\nointerlineskip{\prevdepth-\@m\p@}
```

```
422 \def\offinterlineskip{\baselineskip-\@m\p@
```

```
423 \lineskip\z@ \lineskiplimit\maxdimen}
```

(End definition for `\nointerlineskip` and `\offinterlineskip`.)

```

\vglue
\hglue 424 \def\vglue{\afterassignment\vgl@\skip@=}
425 \def\vgl@{\par \dimen@\prevdepth \hrule \@height\z@
426 \nobreak\vskip\skip@ \prevdepth\dimen@}
427 \def\hglue{\afterassignment\hgl@\skip@=}
428 \def\hgl@{\leavevmode \count@\spacefactor \vrule \@width\z@
429 \nobreak\hskip\skip@ \spacefactor\count@}

(End definition for \vglue and \hglue.)
LATEX defines ~ in ltdefs.dtx.

\slash This generates a / acting a bit like - but still allows hyphenation in the word part
preceding it (but not after).
430 \def\slash{/\penalty\exhyphenpenalty}

(End definition for \slash.)

\break
\nobreak 431 \def\break{\penalty-\@M}
\allowbreak 432 \def\nobreak{\penalty \@M}
433 \def\allowbreak{\penalty \z@}

(End definition for \break, \nobreak, and \allowbreak.)

\filbreak
\goodbreak 434 \def\filbreak{\par\vfil\penalty-200\vfilneg}
435 \def\goodbreak{\par\penalty-500 }

(End definition for \filbreak and \goodbreak.)

\eject Define \eject as in plain TEX but define \supereject only in the compatibility file.
436 \def\eject{\par\break}

(End definition for \eject.)

\removelastskip
437 \def\removelastskip{\ifdim\lastskip=\z@\else\vskip-\lastskip\fi}

(End definition for \removelastskip.)

\smallbreak
\medbreak 438 \def\smallbreak{\par\ifdim\lastskip<\smallskipamount
\bigbreak 439 \removelastskip\penalty-50\smallskip\fi}
440 \def\medbreak{\par\ifdim\lastskip<\medskipamount
\removelastskip\penalty-100\medskip\fi}
441 \def\bigbreak{\par\ifdim\lastskip<\bigskipamount
\removelastskip\penalty-200\bigskip\fi}
442
443

(End definition for \smallbreak, \medbreak, and \bigbreak.)

\m@th
444 \def\m@th{\mathsurround\z@}

(End definition for \m@th.)

```

`\underbar` Due to L^AT_EX's redefinition of `\underline` plain T_EX's `\underbar` can be done in a simpler fashion (but do we need it at all?).

```

445 \def\underbar#1{\underline{\sbox\tw@{#1}\dp\tw@z@{\box\tw@}}}

(End definition for \underbar.)

```

`\strutbox` L^AT_EX sets `\strutbox` in `\set@fontsize`.

```

\strut
446 \newbox\strutbox
447 \def\strut{\relax\ifmmode\copy\strutbox\else\unhcopy\strutbox\fi}

(End definition for \strutbox and \strut.)

```

`\hidewidth` For alignment entries that can stick out.

```

448 \def\hidewidth{\hskip\hideskip}

(End definition for \hidewidth.)

```

`\narrower`

```

449 \def\narrower{%
450   \advance\leftskip\parindent
451   \advance\rightskip\parindent}

(End definition for \narrower.)

```

L^AT_EX defines `\ae` and similar commands elsewhere.

```

452 \chardef\%= '\%
453 \chardef\&= '\&
454 \chardef\#= '\#

```

Most text commands are actually encoding specific and therefore defined later, so commented out or removed from this file.

`\leavevmode` begins a paragraph, if necessary

```

455 \def\leavevmode{\unhbox\voidb@x}

(End definition for \leavevmode.)

```

`\mathhexbox`

```

456 \def\mathhexbox#1#2#3{\mbox{$\m@th \mathchar"#1#2#3$}}

(End definition for \mathhexbox.)

```

`\ialign`

```

457 \def\ialign{\everycr{}\tabskipz@skip\halign} % initialized \halign

(End definition for \ialign.)

```

`\oalign`

```

\oalign
\o@lign
458 \def\oalign#1{\leavevmode\vtop{\baselineskipz@skip \lineskip.25ex%
\ialign{##\cr#1\cr}}
459 \oalign{##\cr#1\cr}}
460 \def\o@lign{\lineskiplimitz@ \oalign}
461 \def\oalign{\lineskiplimit-\maxdimen \oalign}

(End definition for \oalign, \o@lign, and \oalign.)

```

`\sh@ft` The definition of this macro in `plain.tex` was improved in about 1997; but as a result its usage was changed and its new definition is not appropriate for L^AT_EX.

Since the version given here has been in use by L^AT_EX for many years it does not seem prudent to remove it now. As far as we can tell it has only been used to define `\b` and `\d` but this cannot be certain.

```
462 \def\sh@ft#1{\dimen@.00#1ex\multiply\dimen@\fontdimen1\font
463 \kern-.0156\dimen@} % compensate for slant in lowered accents
```

(End definition for \sh@ft.)

`\ltx@sh@ft` This is the L^AT_EX version of the second incarnation of the plain macro `\sh@ft`, which takes a dimension as its argument. It shifts a pseudo-accent horizontally by an amount proportional to the product of its argument and the slant-per-point (`fontdimen 1`).

```
464 \def\ltx@sh@ft #1{%
465 \dimen@ #1%
466 \kern \strip@pt
467 \fontdimen1\font \dimen@
468 } % kern by #1 times the current slant
```

(End definition for \ltx@sh@ft.)

L^AT_EX change: the text commands such as `\d`, `\b`, `\c`, `\copyright`, `\TeX` are now defined elsewhere.

L^AT_EX change: Make `\t` work in a moving argument. Now defined elsewhere.

`\hrulefill` L^AT_EX change: `\kern\z@` added to end of `\hrulefill` and `\dotfill` to make them work
`\dotfill` in ‘tabular’ and ‘array’ environments. (Change made 24 July 1987). L^AT_EX change: `\leavevmode` added at beginning of `\dotfill` and `\hrulefill` so that they work as expected in vertical mode.

```
469 \def\hrulefill{\leavevmode\leaders\hrule\hfill\kern\z@}
```

The box in `\dotfill` originally contained (in `plain.tex`):

```
\mkern 1.5mu .\mkern 1.5mu;
```

the width of `.44em` differs from this by `.04pt` which is probably an acceptable difference within leaders.

```
470 \def\dotfill{%
471 \leavevmode
472 \cleaders \hb@xt@ .44em{\hss.\hss}\hfill
473 \kern\z@}
```

(End definition for \hrulefill and \dotfill.)

INIT_{EX} sets `\sfcode x=1000` for all `x`, except that `\sfcode X=999` for uppercase letters. The following changes are needed:

```
474 \sfcode'\)=0 \sfcode'\ '=0 \sfcode'\]=0
```

The `\nonfrenchspacing` macro will make further changes to `\sfcode` values.

Definitions related to output

`\magnification` doesn’t work in L^AT_EX.

```
def\magnification{\afterassignment\m@g\count@}
def\m@g{\mag\count@
\hsize6.5truein\vsize8.9truein\dimen\footins8truein}
```

`\showoverfull` The following commands are used in debugging:

```
475 \def\showoverfull{\tracingonline\@ne}
```

(End definition for \showoverfull.)

```
\showoutput
\loggingoutput 476 \gdef\loggingoutput{\tracingoutput\@ne
477 \showboxbreadth\maxdimen\showboxdepth\maxdimen\errorstopmode}
478 \gdef\showoutput{\loggingoutput\showoverfull}
479 \</2ekernel>
```

(End definition for \showoutput and \loggingoutput.)

```
\tracingall
\loggingall 480 <latexrelease>\IncludeInRelease{2021/06/01}{\loggingall}
481 <latexrelease> \tracingstacklevels and \tracinglostchars=3}%
482 <*2ekernel | latexrelease>
483 \edef\loggingall{%
484 \tracingstats\tw@
485 \tracingpages\@ne
486 \tracinglostchars\thr@@
487 \tracingparagraphs\@ne
488 \tracinggroups\@ne
489 \tracingifs\@ne
490 \tracingscantokens\@ne
491 \tracingnesting\@ne
492 \errorcontextlines\maxdimen
493 \ifdefined\tracingstacklevels \tracingstacklevels\maxdimen \fi
494 \noexpand \loggingoutput
495 \tracingmacros\tw@
496 \tracingcommands\thr@@
497 \tracingrestores\@ne
498 \tracingassigns\@ne
499 }%
500 \def\tracingall{\showoverfull\loggingall}
501 \</2ekernel | latexrelease>
502 <latexrelease>\EndIncludeInRelease
503 <latexrelease>
504 <latexrelease>\IncludeInRelease{2015/01/01}{\loggingall}{etex tracing}%
505 <latexrelease>\ifx\tracingscantokens\@undefined
506 <latexrelease>\gdef\loggingall{%
507 <latexrelease> \tracingstats\tw@
508 <latexrelease> \tracingpages\@ne
509 <latexrelease> \tracinglostchars\@ne
510 <latexrelease> \tracingparagraphs\@ne
511 <latexrelease> \errorcontextlines\maxdimen
512 <latexrelease> \loggingoutput
513 <latexrelease> \tracingmacros\tw@
514 <latexrelease> \tracingcommands\tw@
515 <latexrelease> \tracingrestores\@ne
516 <latexrelease> }%
517 <latexrelease>\else
518 <latexrelease>\gdef\loggingall{%
519 <latexrelease> \tracingstats\tw@
520 <latexrelease> \tracingpages\@ne
521 <latexrelease> \tracinglostchars\tw@
522 <latexrelease> \tracingparagraphs\@ne
```

```

523 <latexrelease> \tracinggroups\@ne
524 <latexrelease> \tracingifs\@ne
525 <latexrelease> \tracingscantokens\@ne
526 <latexrelease> \tracingnesting\@ne
527 <latexrelease> \errorcontextlines\maxdimen
528 <latexrelease> \loggingoutput
529 <latexrelease> \tracingmacros\tw@
530 <latexrelease> \tracingcommands\thr@@
531 <latexrelease> \tracingrestores\@ne
532 <latexrelease> \tracingassigns\@ne
533 <latexrelease>}%
534 <latexrelease>\fi
535 <latexrelease>\gdef\tracingall{\showoverfull\loggingall}
536 <latexrelease>\EndIncludeInRelease
537 <latexrelease>
538 <latexrelease>\IncludeInRelease{0000/00/00}{\loggingall}{etex tracing}%
539 <latexrelease>\gdef\loggingall{\tracingcommands\tw@\tracingstats\tw@
540 <latexrelease> \tracingpages\@ne\tracinglostchars\@ne
541 <latexrelease> \tracingmacros\tw@\tracingparagraphs\@ne\tracingrestores\@ne
542 <latexrelease> \errorcontextlines\maxdimen\loggingoutput}
543 <latexrelease> \gdef\tracingall{\loggingall\showoverfull}
544 <latexrelease>\EndIncludeInRelease

```

(End definition for \tracingall and \loggingall.)

\tracingnone

```

545 <latexrelease>\IncludeInRelease{2015/01/01}{\tracingnone}%
546 <latexrelease> \turn off etex tracing}%
547 </2ekernel | latexrelease>
548 \edef\tracingnone{%
549 \tracingassigns\z@
550 \tracingrestores\z@
551 \tracingonline\z@
552 \tracingcommands\z@
553 \showboxdepth\m@ne
554 \showboxbreadth\m@ne
555 \tracingoutput\z@
556 \errorcontextlines\m@ne
557 \ifdefined\tracingstacklevels \tracingstacklevels\z@ \fi
558 \tracingnesting\z@
559 \tracingscantokens\z@
560 \tracingifs\z@
561 \tracinggroups\z@
562 \tracingparagraphs\z@
563 \tracingmacros\z@
564 \tracinglostchars\@ne
565 \tracingpages\z@
566 \tracingstats\z@
567 }%
568 </2ekernel | latexrelease>
569 <latexrelease>\EndIncludeInRelease
570 <latexrelease>
571 <latexrelease>\IncludeInRelease{2015/01/01}{\tracingnone}%
572 <latexrelease> \turn off etex tracing}%

```



```

573 <latexrelease>\ifx\tracingscantokens\@undefined
574 <latexrelease>\def\tracingnone{%
575 <latexrelease> \tracingonline\z@
576 <latexrelease> \tracingcommands\z@
577 <latexrelease> \showboxdepth\m@ne
578 <latexrelease> \showboxbreadth\m@ne
579 <latexrelease> \tracingoutput\z@
580 <latexrelease> \errorcontextlines\m@ne
581 <latexrelease> \tracingrestores\z@
582 <latexrelease> \tracingparagraphs\z@
583 <latexrelease> \tracingmacros\z@
584 <latexrelease> \tracinglostchars\@ne
585 <latexrelease> \tracingpages\z@
586 <latexrelease> \tracingstats\z@
587 <latexrelease>}%
588 <latexrelease>\else
589 <latexrelease>\def\tracingnone{%
590 <latexrelease> \tracingassigns\z@
591 <latexrelease> \tracingrestores\z@
592 <latexrelease> \tracingonline\z@
593 <latexrelease> \tracingcommands\z@
594 <latexrelease> \showboxdepth\m@ne
595 <latexrelease> \showboxbreadth\m@ne
596 <latexrelease> \tracingoutput\z@
597 <latexrelease> \errorcontextlines\m@ne
598 <latexrelease> \tracingnesting\z@
599 <latexrelease> \tracingscantokens\z@
600 <latexrelease> \tracingifs\z@
601 <latexrelease> \tracinggroups\z@
602 <latexrelease> \tracingparagraphs\z@
603 <latexrelease> \tracingmacros\z@
604 <latexrelease> \tracinglostchars\@ne
605 <latexrelease> \tracingpages\z@
606 <latexrelease> \tracingstats\z@
607 <latexrelease>}%
608 <latexrelease>\fi
609 <latexrelease>\EndIncludeInRelease
610 <latexrelease>
611 <latexrelease>\IncludeInRelease{0000/00/00}{\tracingnone}%
612 <latexrelease> {turn off etex tracing}%
613 <latexrelease>\let\tracingnone\@undefined
614 <latexrelease>\EndIncludeInRelease

```

(End definition for \tracingnone.)

\hideoutput

```

615 <*2ekernel | latexrelease>
616 <latexrelease>\IncludeInRelease{2015/01/01}{\hideoutput}%
617 <latexrelease> {hide output from tracing}%
618 \def\hideoutput{%
619 \tracingoutput\z@
620 \showboxbreadth\m@ne
621 \showboxdepth\m@ne
622 \tracingonline\m@ne

```

```

623 }%
624 <latexrelease>\EndIncludeInRelease
625 <latexrelease>
626 <latexrelease>\IncludeInRelease{0000/00/00}{\hideoutput}%
627 <latexrelease>{hide output from tracing}%
628 <latexrelease>\let\hideoutput\@undefined
629 <latexrelease>\EndIncludeInRelease
630 </2ekernel|latexrelease>

(End definition for \hideoutput.)
    LATEX change: \showhyphens Defined later.
    Punctuation affects the spacing.

631 <*2ekernel>
632 \nonfrenchspacing
633 </2ekernel>

```

File c

ltvers.dtx

1 Version Identification

First we identify the date and version number of this release of L^AT_EX, and set `\everyjob` so that it is printed at the start of every L^AT_EX run.

`\fmtname` A `\patch@level` of 0 or higher denotes an official public release. A negative value indicates a candidate release that is not distributed.

`\fmtversion`

`\latexreleaseversion` If we put code updates into the kernel that are supposed to go into the next release we set the `\patch@level` to -1 and the `\fmtversion` / `\latexreleaseversion` to the dated of the next release (guessed, the real value is not so important and will get corrected when we make the release official).

`\patch@level`

If the `\patch@level` is already at -1 we do nothing here and use the `\fmtversion` date for any new `\IncludeInRelease` line when we add further code.

Finally, if we do make a public release we either just set the `\patch@level` to zero (if our initial guess was good) or we also change the date and then have to additionally change to that date on all the `\IncludeInRelease` statements that used the “guessed” date.

```
1 <*2ekernel>
2 \def\fmtname{LaTeX2e}
3 \edef\fmtversion
4 </2ekernel>
5 <latexrelease>\edef\latexreleaseversion
6 <*2ekernel | latexrelease>
7 {2021-06-01}
8 </2ekernel | latexrelease>
9 <*2ekernel>
10 \def\patch@level{0}
```

For more fine grain control there is the possibility to name the current development branch. This is only used when the `\patch@level` is negative (i.e., a pre-release format) and is intended to help us internally when we locally install a format out of some development branch.

```
\development@branch@name 11 \edef\development@branch@name{develop \the\year-\the\month-\the\day}
```

(End definition for `\fmtname` and others.)

Check that the format being made is not too old. The error message complains about ‘more than 5 years’ but in fact the error is not triggered until 65 months.

This code is currently not activated as we don’t know if we already got to the last official 2e version (due to staff shortage or due to a successor (think positive:-)).

```
12 \iffalse
13 \def\reserved@a#1/#2/#3\@nil{%
14   \count@ \year
15   \advance\count@-#1\relax
16   \multiply\count@ by 12\relax
17   \advance\count@ \month
18   \advance\count@-#2\relax}
19 \expandafter\reserved@a\fmtversion\@nil
```



```

69 <*2ekernel | latexrelease>
70 <latexrelease>\newif\if@includeinrelease
71 <latexrelease>\@includeinreleasefalse
72 \def\IncludeInRelease#1{%
73   \if@includeinrelease
74     \PackageError{latexrelease}{mis-matched IncludeInRelease}%
75       {There is an \string\EndIncludeRelease\space missing}%
76     \@includeinreleasefalse
77   \fi
78   \ifnum0%
79     \ifx\new@moduledate\empty\else 1\fi
80     \ifnum \expandafter\@parse@version#1//00\@nil=0 1\fi
81     =11
82     \expandafter\@firstoftwo
83   \else
84     \expandafter\@secondoftwo
85   \fi
86   {\finish@module@release{#1}}%
87   {\kernel@ifnextchar[%
88     {\@IncludeInRelease{#1}}
89     {\@IncludeInRelease{#1}[#1]}}}%
90 \def\finish@module@release#1#2#3{%
91   \toks@{[#1] #3}%
92   \ifnum\expandafter\@parse@version\new@moduledate//00\@nil
93     >\expandafter\@parse@version\fmtversion//00\@nil
94     \GenericInfo{}\{Applying: \the\toks@\}%
95   \else
96     \GenericInfo{}\{Skipping: \the\toks@\}%
97     \expandafter\gobble@finish@module@release
98   \fi}
99 \long\def\gobble@finish@module@release#1\EndModuleRelease{%
100   \EndModuleRelease}

```

If a specific date has not been specified in latexrelease use ‘#1’.

```

101 \def\@IncludeInRelease#1[#2]{\@IncludeInRelease{#2}}
102 \def\@IncludeInRelease#1#2#3{%
103   \toks@{[#1] #3}%
104   \expandafter\ifx\csname\string#2+\@currname+IIR\endcsname\relax

```

If we roll back and the first patch already match then applying that is actually reapplying what is already in the format, i.e., it is useless and possibly allocating new registers. However, it makes the logic simpler so this is the way it is for now. In theory we could always jump over the first patch because that is only really needed for rolling forward. So maybe one day ...

```

105     \ifnum\expandafter\@parse@version#1//00\@nil
106       >\expandafter\@parse@version\fmtversion//00\@nil
107       \GenericInfo{}\{Skipping: \the\toks@\}%
108       \expandafter\expandafter\expandafter\gobble@IncludeInRelease
109     \else
110       \GenericInfo{}\{Applying: \the\toks@\}%
111       \@includeinreleasetrue
112       \expandafter\let\csname\string#2+\@currname+IIR\endcsname\empty
113     \fi

```

```

114 \else
115 \GenericInfo{}{Already applied: \the\toks@}%
116 \expandafter\@gobble\IncludeInRelease
117 \fi
118 }

119 \def\EndIncludeInRelease{%
120 \if@includeinrelease
121 \@includeinreleasefalse
122 \else
123 \PackageError{latexrelease}{mis-matched EndIncludeInRelease}{}%
124 \fi
125 \if@skipping@module
126 \expandafter\new@module@skip
127 \fi}

128 \long\def\@gobble\IncludeInRelease#1\EndIncludeInRelease{%
129 \@includeinreleasefalse
130 \@check\IncludeInRelease#1\IncludeInRelease\@check\IncludeInRelease
131 \@end\check\IncludeInRelease}

132 \long\def\@check\IncludeInRelease#1\IncludeInRelease
133 #2#3\@end\check\IncludeInRelease{%
134 \ifx\@check\IncludeInRelease#2\else
135 \PackageError{latexrelease}{skipped IncludeInRelease for tag \string#2}{}%
136 \fi
137 \if@skipping@module
138 \expandafter\new@module@skip
139 \fi}

```

(End definition for \IncludeInRelease and others.)

1.1 Declaring an all-new module

When we have a whole new module, we can't roll back to a date where such module exists, otherwise hundreds of "command already defined" errors will pop up. But we can't skip it altogether either, because the module might have changes we still want applied, so a more detailed cherry-picking of code chunks have to be done.

```

\if@skipping@module
\NewModuleRelease
\EndModuleRelease
\new@module@skip
\new@modulename
\new@moduledate
140 \let\if@skipping@module\iffalse
141 \def\@skipping@moduletrue{\let\if@skipping@module\iftrue}
142 \def\@skipping@modulefalse{\let\if@skipping@module\iffalse}
143 \let\new@modulename\@empty
144 \let\new@moduledate\@empty
145 \def\NewModuleRelease#1#2#3{%
146 \ifx\new@modulename\@empty \else
147 \latexerror{Nested \noexpand\NewModuleRelease forbidden.}\@ehd \fi
148 \edef\new@moduledate{#1}%
149 \edef\new@modulename{#2}%
150 \GenericInfo{}{BEGIN module: \new@modulename\space (\new@moduledate)}%
151 \GenericInfo{}{ \spaces\@spaces\space#3\@gobble}%
152 \ifnum\sourceLaTeXdate<%
153 \expandafter\@parse@version\new@moduledate//00\@nil\relax
154 \ifnum\expandafter\@parse@version\fmtversion//00\@nil<%
155 \expandafter\@parse@version\new@moduledate//00\@nil\relax
156 \GenericInfo{}{Skipping module \new@modulename}%

```

```

157     \expandafter\expandafter
158     \expandafter\gobble@finish@module@release
159     \else
160     \GenericInfo{}{Applying module \new@modulename}
161     \@skipping@modulefalse
162     \fi
163 \else
164     \GenericInfo{}{Skipping module \new@modulename}
165     \@skipping@moduletrue
166     \expandafter\new@module@skip
167     \fi}
168 \long\def\new@module@skip#1\IncludeInRelease{\IncludeInRelease}
169 \def\EndModuleRelease{%
170     \ifx\new@modulename\@empty
171     \@latex@error{Extra \string\EndModuleRelease.}\@eha
172     \else
173     \GenericInfo{}{END module: \new@modulename\space (\new@moduledate)}%
174     \let\new@modulename\@empty
175     \let\new@moduledate\@empty
176     \@skipping@modulefalse
177     \fi}

(End definition for \if@skipping@module and others.)

178 </2ekernel | latexrelease>

```

File d

lualatex.dtx

1 Overview

LuaTeX adds a number of engine-specific functions to TeX. Several of these require set up that is best done in the kernel or need related support functions. This file provides *basic* support for LuaTeX at the L^AT_ΕX 2_ε kernel level plus as a loadable file which can be used with plain TeX and L^AT_ΕX.

This file contains code for both TeX (to be stored as part of the format) and Lua (to be loaded at the start of each job). In the Lua code, the kernel uses the namespace `luatexbase`.

The following `\count` registers are used here for register allocation:

```
\e@alloc@attribute@count Attributes (default 258)
\e@alloc@ccodetable@count Category code tables (default 259)
\e@alloc@luafunction@count Lua functions (default 260)
  \e@alloc@whatsit@count User whatsits (default 261)
  \e@alloc@bytecode@count Lua bytecodes (default 262)
  \e@alloc@luachunk@count Lua chunks (default 263)
```

(`\count 256` is used for `\newmarks` allocation and `\count 257` is used for `\newXeTeXintercharclass` with XeTeX, with code defined in `ltfinal.dtx`). With any L^AT_ΕX 2_ε kernel from 2015 onward these registers are part of the block in the extended area reserved by the kernel (prior to 2015 the L^AT_ΕX 2_ε kernel did not provide any functionality for the extended allocation area).

2 Core TeX functionality

The commands defined here are defined for possible inclusion in a future L^AT_ΕX format, however also extracted to the file `lualatex.tex` which may be used with older L^AT_ΕX formats, and with plain TeX.

<code>\newattribute</code>	<code>\newattribute{⟨attribute⟩}</code> Defines a named <code>\attribute</code> , indexed from 1 (<i>i.e.</i> <code>\attribute0</code> is never defined). Attributes initially have the marker value <code>-7FFFFFFF</code> ('unset') set by the engine.
<code>\newcatcodetable</code>	<code>\newcatcodetable{⟨catcodetable⟩}</code> Defines a named <code>\catcodetable</code> , indexed from 1 (<code>\catcodetable0</code> is never assigned). A new catcode table will be populated with exactly those values assigned by IniTeX (as described in the LuaTeX manual).
<code>\newluafunction</code>	<code>\newluafunction{⟨function⟩}</code> Defines a named <code>\luafunction</code> , indexed from 1. (Lua indexes tables from 1 so <code>\luafunction0</code> is not available).
<code>\newwhatsit</code>	<code>\newwhatsit{⟨whatsit⟩}</code> Defines a custom <code>\whatsit</code> , indexed from 1.
<code>\newluabytecode</code>	<code>\newluabytecode{⟨bytecode⟩}</code>

	Allocates a number for Lua bytecode register, indexed from 1.
<code>\newluachunkname</code>	<code>newluachunkname{⟨chunkname⟩}</code> Allocates a number for Lua chunk register, indexed from 1. Also enters the name of the register (without backslash) into the <code>lua.name</code> table to be used in stack traces.
<code>\catcodetable@initex</code>	Predefined category code tables with the obvious assignments. Note that the <code>latex</code> and
<code>\catcodetable@string</code>	<code>atletter</code> tables set the full Unicode range to the codes predefined by the kernel.
<code>\catcodetable@latex</code>	<code>\setattribute{⟨attribute⟩}{⟨value⟩}</code>
<code>\catcodetable@atletter</code>	<code>\unsetattribute{⟨attribute⟩}</code>
<code>\setattribute</code>	Set and unset attributes in a manner analogous to <code>\setlength</code> . Note that attributes
<code>\unsetattribute</code>	take a marker value when unset so this operation is distinct from setting the value to zero.

3 Plain T_EX interface

The `luatex` interface may be used with plain T_EX using `\input{luatex}`. This inputs `luatex.tex` which inputs `etex.src` (or `etex.sty` if used with L^AT_EX) if it is not already input, and then defines some internal commands to allow the `luatex` interface to be defined.

The `luatexbase` package interface may also be used in plain T_EX, as before, by inputting the package `\input luatexbase.sty`. The new version of `luatexbase` is based on this `luatex` code but implements a compatibility layer providing the interface of the original package.

4 Lua functionality

4.1 Allocators in Lua

<code>new_attribute</code>	<code>luatexbase.new_attribute(⟨attribute⟩)</code> Returns an allocation number for the <code>⟨attribute⟩</code> , indexed from 1. The attribute will be initialised with the marker value <code>-"7FFFFFFF</code> ('unset'). The attribute allocation sequence is shared with the T _E X code but this function does <i>not</i> define a token using <code>\attributedef</code> . The attribute name is recorded in the <code>attributes</code> table. A metatable is provided so that the table syntax can be used consistently for attributes declared in T _E X or Lua.
<code>new_whatsit</code>	<code>luatexbase.new_whatsit(⟨whatsit⟩)</code> Returns an allocation number for the custom <code>⟨whatsit⟩</code> , indexed from 1.
<code>new_bytecode</code>	<code>luatexbase.new_bytecode(⟨bytecode⟩)</code> Returns an allocation number for a bytecode register, indexed from 1. The optional <code>⟨name⟩</code> argument is just used for logging.
<code>new_chunkname</code>	<code>luatexbase.new_chunkname(⟨chunkname⟩)</code> Returns an allocation number for a Lua chunk name for use with <code>\directlua</code> and <code>\latelua</code> , indexed from 1. The number is returned and also <code>⟨name⟩</code> argument is added to the <code>lua.name</code> array at that index.
<code>new_luafunction</code>	<code>luatexbase.new_luafunction(⟨functionname⟩)</code> Returns an allocation number for a lua function for use with <code>\luafunction</code> , <code>\lateluafunction</code> , and <code>\luaodef</code> , indexed from 1. The optional <code>⟨functionname⟩</code> argument is just used for logging.

These functions all require access to a named T_EX count register to manage their allocations. The standard names are those defined above for access from T_EX,

e.g. `\e@alloc@attribute@count`, but these can be adjusted by defining the variable `\type_count_name` before loading `ltnlualatex.lua`, for example

```
local attribute_count_name = "attributetracker"
require("ltnlualatex")
```

would use a \TeX `\count` (`\countdef`'d token) called `attributetracker` in place of `\e@alloc@attribute@count`.

4.2 Lua access to \TeX register numbers

`registernumber` `luatexbase.registernumber(<name>)`
 Sometimes (notably in the case of Lua attributes) it is necessary to access a register *by number* that has been allocated by \TeX . This package provides a function to look up the relevant number using Lua \TeX 's internal tables. After for example `\newattribute\myattrib`, `\myattrib` would be defined by (say) `\myattrib=\attribute15`. `luatexbase.registernumber("myattrib")` would then return the register number, 15 in this case. If the string passed as argument does not correspond to a token defined by `\attributedef`, `\countdef` or similar commands, the Lua value `false` is returned.

As an example, consider the input:

```
\newcommand\test[1]{%
\typeout{#1: \expandafter\meaning\csname#1\endcsname^^J
\space\space\space\space
\directlua{tex.write(luatexbase.registernumber("#1") or "bad input")}}%
}

\test{undefinedrubbish}

\test{space}

\test{hbox}

\test{@MM}

\test{@tempdima}
\test{@tempdimb}

\test{strutbox}

\test{sixt@@n}

\attributedef\myattr=12
\myattr=200
\test{myattr}
```

If the demonstration code is processed with Lua \TeX then the following would be produced in the log and terminal output.

```
undefinedrubbish: \relax
      bad input
```

```

space: macro:->
    bad input
hbox: \hbox
    bad input
@MM: \mathchar"4E20
    20000
@tempdima: \dimen14
    14
@tempdimb: \dimen15
    15
strutbox: \char"B
    11
sist@@n: \char"10
    16
myattr: \attribute12
    12

```

Notice how undefined commands, or commands unrelated to registers do not produce an error, just return `false` and so print `bad input` here. Note also that commands defined by `\newbox` work and return the number of the box register even though the actual command holding this number is a `\chardef` defined token (there is no `\boxdef`).

4.3 Module utilities

`provides_module` `luatexbase.provides_module(<info>)`

This function is used by modules to identify themselves; the `info` should be a table containing information about the module. The required field `name` must contain the name of the module. It is recommended to provide a field `date` in the usual L^AT_EX format `yyyy/mm/dd`. Optional fields `version` (a string) and `description` may be used if present. This information will be recorded in the log. Other fields are ignored.

`module_info` `luatexbase.module_info(<module>, <text>)`

`module_warning` `luatexbase.module_warning(<module>, <text>)`

`module_error` `luatexbase.module_error(<module>, <text>)`

These functions are similar to L^AT_EX's `\PackageError`, `\PackageWarning` and `\PackageInfo` in the way they format the output. No automatic line breaking is done, you may still use `\n` as usual for that, and the name of the package will be prepended to each output line.

Note that `luatexbase.module_error` raises an actual Lua error with `error()`, which currently means a call stack will be dumped. While this may not look pretty, at least it provides useful information for tracking the error down.

4.4 Callback management

`add_to_callback` `luatexbase.add_to_callback(<callback>, <function>, <description>)` Registers the *<function>* into the *<callback>* with a textual *<description>* of the function. Functions are inserted into the callback in the order loaded.

`remove_from_callback` `luatexbase.remove_from_callback(<callback>, <description>)` Removes the callback function with *<description>* from the *<callback>*. The removed function and its description are returned as the results of this function.

`in_callback` `luatexbase.in_callback(<callback>, <description>)` Checks if the *<description>* matches one of the functions added to the list for the *<callback>*, returning a boolean value.

<code>disable_callback</code>	<code>luatexbase.disable_callback(<callback>)</code> Sets the <code><callback></code> to <code>false</code> as described in the LuaTeX manual for the underlying <code>callback.register</code> built-in. Callbacks will only be set to false (and thus be skipped entirely) if there are no functions registered using the callback.
<code>callback_descriptions</code>	A list of the descriptions of functions registered to the specified callback is returned. <code>{}</code> is returned if there are no functions registered.
<code>create_callback</code>	<code>luatexbase.create_callback(<name>,metatype,<default>)</code> Defines a user defined callback. The last argument is a default function or <code>false</code> .
<code>call_callback</code>	<code>luatexbase.call_callback(<name>,...)</code> Calls a user defined callback with the supplied arguments.

5 Implementation

```

1 <*2ekernel | tex | latexrelease>
2 <2ekernel | latexrelease>\ifx\directlua\@undefined\else

```

5.1 Minimum LuaTeX version

LuaTeX has changed a lot over time. In the kernel support for ancient versions is not provided: trying to build a format with a very old binary therefore gives some information in the log and loading stops. The cut-off selected here relates to the tree-searching behaviour of `require()`: from version 0.60, LuaTeX will correctly find Lua files in the `texmf` tree without ‘help’.

```

3 <latexrelease>\IncludeInRelease{2015/10/01}
4 <latexrelease>                {\newluafunction}{LuaTeX}%
5 \ifnum\luatexversion<60 %
6   \wlog{*****}
7   \wlog{* LuaTeX version too old for ltuatex support *}
8   \wlog{*****}
9   \expandafter\endinput
10 \fi

```

Two simple L^AT_EX macros from `ltdfn.s.dtx` have to be defined here because `ltdfn.s.dtx` is not loaded yet when `ltluatex.dtx` is executed.

```

11 \long\def\@gobble#1{}
12 \long\def\@firstofone#1{#1}

```

5.2 Older L^AT_EX/Plain T_EX setup

```

13 <*tex>

```

Older L^AT_EX formats don’t have the primitives with ‘native’ names: sort that out. If they already exist this will still be safe.

```

14 \directlua{tex.enableprimitives("",tex.extraprimitives("luatex"))}
15 \ifx\@alloc\@undefined

```

In pre-2014 L^AT_EX, or plain T_EX, load `etex.{sty,src}`.

```

16 \ifx\documentclass\@undefined
17   \ifx\loccount\@undefined
18     \input{etex.src}%
19   \fi
20   \catcode'\@=11 %
21   \outer\expandafter\def\csname newfam\endcsname

```

```

22                                     {\alloc@8\fam\chardef\et@xmaxfam}
23 \else
24   \RequirePackage{etex}
25   \expandafter\def\csname newfam\endcsname
26                                     {\alloc@8\fam\chardef\et@xmaxfam}
27   \expandafter\let\expandafter\new@mathgroup\csname newfam\endcsname
28 \fi

```

5.2.1 Fixes to etex.src/etex.sty

These could and probably should be made directly in an update to `etex.src` which already has some LuaTeX-specific code, but does not define the correct range for LuaTeX. 2015-07-13 higher range in luatex.

```

29 \edef \et@xmaxregs {\ifx\directlua\@undefined 32768\else 65536\fi}

```

luatex/xetex also allow more math fam.

```

30 \edef \et@xmaxfam {\ifx\Umathcode\@undefined\sixt@@n\else\ccclvi\fi}
31 \count 270=\et@xmaxregs % locally allocates \count registers
32 \count 271=\et@xmaxregs % ditto for \dimen registers
33 \count 272=\et@xmaxregs % ditto for \skip registers
34 \count 273=\et@xmaxregs % ditto for \muskip registers
35 \count 274=\et@xmaxregs % ditto for \box registers
36 \count 275=\et@xmaxregs % ditto for \toks registers
37 \count 276=\et@xmaxregs % ditto for \marks classes

```

and 256 or 16 fam. (Done above due to plain/LaTeX differences in ltuatex.)

```

38 % \outer\def\newfam{\alloc@8\fam\chardef\et@xmaxfam}

```

End of proposed changes to `etex.src`

5.2.2 luatex specific settings

Switch to global cf `luatex.sty` to leave room for inserts not really needed for luatex but possibly most compatible with existing use.

```

39 \expandafter\let\csname newcount\expandafter\expandafter\endcsname
40   \csname globcount\endcsname
41 \expandafter\let\csname newdimen\expandafter\expandafter\endcsname
42   \csname globdimen\endcsname
43 \expandafter\let\csname newskip\expandafter\expandafter\endcsname
44   \csname globskip\endcsname
45 \expandafter\let\csname newbox\expandafter\expandafter\endcsname
46   \csname globbox\endcsname

```

Define `\e@alloc` as in latex (the existing macros in `etex.src` hard to extend to further register types as they assume specific 26x and 27x count range. For compatibility the existing register allocation is not changed.

```

47 \chardef\e@alloc@top=65535
48 \let\e@alloc\chardef\chardef
49 \def\e@alloc#1#2#3#4#5#6{%
50   \global\advance#3\@ne
51   \e@ch@ck{#3}{#4}{#5}#1%
52   \allocationnumber#3\relax
53   \global#2#6\allocationnumber
54   \wlog{\string#6=\string#1\the\allocationnumber}}%

```

```

55 \gdef\ech@ck#1#2#3#4{%
56   \ifnum#1<#2\else
57     \ifnum#1=#2\relax
58       #1\ccclvi
59       \ifx\count#4\advance#1 10 \fi
60     \fi
61     \ifnum#1<#3\relax
62     \else
63       \errmessage{No room for a new \string#4}%
64     \fi
65   \fi}%

```

Fix up allocations not to clash with etex.src.

```

66 \expandafter\csname newcount\endcsname\@alloc@attribute@count
67 \expandafter\csname newcount\endcsname\@alloc@ccodetable@count
68 \expandafter\csname newcount\endcsname\@alloc@luafunction@count
69 \expandafter\csname newcount\endcsname\@alloc@whatsit@count
70 \expandafter\csname newcount\endcsname\@alloc@bytecode@count
71 \expandafter\csname newcount\endcsname\@alloc@luachunk@count

```

End of conditional setup for plain T_EX / old L^AT_EX.

```

72 \fi
73 \</tex>

```

5.3 Attributes

\newattribute As is generally the case for the LuaT_EX registers we start here from 1. Notably, some code assumes that `\attribute0` is never used so this is important in this case.

```

74 \ifx\@alloc@attribute@count\@undefined
75   \countdef\@alloc@attribute@count=258
76   \@alloc@attribute@count=\z@
77 \fi
78 \def\newattribute#1{%
79   \@alloc\attribute\attributedef
80   \@alloc@attribute@count\m@ne\@alloc@top#1%
81 }

```

(End definition for `\newattribute`.)

\setattribute Handy utilities.

```

\unsetattribute 82 \def\setattribute#1#2{#1=\numexpr#2\relax}
83 \def\unsetattribute#1{#1=-"7FFFFFFF\relax}

```

(End definition for `\setattribute` and `\unsetattribute`.)

5.4 Category code tables

\newcatcodetable Category code tables are allocated with a limit half of that used by LuaT_EX for everything else. At the end of allocation there needs to be an initialization step. Table 0 is already taken (it's the global one for current use) so the allocation starts at 1.

```

84 \ifx\@alloc@ccodetable@count\@undefined
85   \countdef\@alloc@ccodetable@count=259
86   \@alloc@ccodetable@count=\z@
87 \fi

```

```

88 \def\newcatcodetable#1{%
89   \e@alloc\catcodetable\chardef
90   \e@alloc\ccodetable@count\m@ne{"8000}#1%
91   \initcatcodetable\allocationnumber
92 }

```

(End definition for \newcatcodetable.)

```

\catcodetable@initex Save a small set of standard tables. The Unicode data is read here in using a parser sim-
\catcodetable@string plified from that in load-unicode-data: only the nature of letters needs to be detected.
\catcodetable@latex
\catcodetable@atletter
93 \newcatcodetable\catcodetable@initex
94 \newcatcodetable\catcodetable@string
95 \begingroup
96   \def\setrangecatcode#1#2#3{%
97     \ifnum#1>#2 %
98       \expandafter\@gobble
99     \else
100       \expandafter\@firstofone
101     \fi
102     {%
103       \catcode#1=#3 %
104       \expandafter\setrangecatcode\expandafter
105       {\number\numexpr#1 + 1\relax}{#2}{#3}
106     }%
107   }
108   \@firstofone{%
109     \catcodetable\catcodetable@initex
110     \catcode0=12 %
111     \catcode13=12 %
112     \catcode37=12 %
113     \setrangecatcode{65}{90}{12}%
114     \setrangecatcode{97}{122}{12}%
115     \catcode92=12 %
116     \catcode127=12 %
117     \savecatcodetable\catcodetable@string
118   \endgroup
119   }%
120 \newcatcodetable\catcodetable@latex
121 \newcatcodetable\catcodetable@atletter
122 \begingroup
123   \def\parseunicodedataI#1;#2;#3;#4\relax{%
124     \parseunicodedataII#1;#3;#2 First>\relax
125   }%
126   \def\parseunicodedataII#1;#2;#3 First>#4\relax{%
127     \ifx\relax#4\relax
128       \expandafter\parseunicodedataIII
129     \else
130       \expandafter\parseunicodedataIV
131     \fi
132     {#1}#2\relax%
133   }%
134   \def\parseunicodedataIII#1#2#3\relax{%
135     \ifnum 0%
136     \if L#21\fi

```

```

137     \if M#21\fi
138     >0 %
139     \catcode"#1=11 %
140     \fi
141 }%
142 \def\parseunicodedataIV#1#2#3\relax{%
143     \read\unicoderead to \unicodedataline
144     \if L#2%
145         \count0="#1 %
146         \expandafter\parseunicodedataV\unicodedataline\relax
147     \fi
148 }%
149 \def\parseunicodedataV#1;#2\relax{%
150     \loop
151         \unless\ifnum\count0>"#1 %
152             \catcode\count0=11 %
153             \advance\count0 by 1 %
154     \repeat
155 }%
156 \def\storedpar{\par}%
157 \chardef\unicoderead=\numexpr\count16 + 1\relax
158 \openin\unicoderead=UnicodeData.txt %
159 \loop\unless\ifeof\unicoderead %
160     \read\unicoderead to \unicodedataline
161     \unless\ifx\unicodedataline\storedpar
162         \expandafter\parseunicodedataI\unicodedataline\relax
163     \fi
164 \repeat
165 \closein\unicoderead
166 \@firstofone{%
167     \catcode64=12 %
168     \savecatcodetable\catcodetable@latex
169     \catcode64=11 %
170     \savecatcodetable\catcodetable@atletter
171 }
172 \endgroup

```

(End definition for \catcodetable@initex and others.)

5.5 Named Lua functions

`\newluafunction` Much the same story for allocating LuaTeX functions except here they are just numbers so they are allocated in the same way as boxes. Lua indexes from 1 so once again slot 0 is skipped.

```

173 \ifx\e@alloc@luafunction@count\@undefined
174     \countdef\e@alloc@luafunction@count=260
175     \e@alloc@luafunction@count=\z@
176 \fi
177 \def\newluafunction{%
178     \e@alloc@luafunction\e@alloc@chardef
179     \e@alloc@luafunction@count\m@ne\e@alloc@top
180 }

```

(End definition for \newluafunction.)

5.6 Custom whatsits

`\newwhatsit` These are only settable from Lua but for consistency are definable here.

```
181 \ifx\e@alloc@whatsit@count\@undefined
182   \countdef\e@alloc@whatsit@count=261
183   \e@alloc@whatsit@count=\z@
184 \fi
185 \def\newwhatsit#1{%
186   \e@alloc@whatsit\e@alloc@chardef
187   \e@alloc@whatsit@count\m@ne\e@alloc@top#1%
188 }
```

(End definition for \newwhatsit.)

5.7 Lua bytecode registers

`\newluabytocode` These are only settable from Lua but for consistency are definable here.

```
189 \ifx\e@alloc@bytocode@count\@undefined
190   \countdef\e@alloc@bytocode@count=262
191   \e@alloc@bytocode@count=\z@
192 \fi
193 \def\newluabytocode#1{%
194   \e@alloc@luabytocode\e@alloc@chardef
195   \e@alloc@bytocode@count\m@ne\e@alloc@top#1%
196 }
```

(End definition for \newluabytocode.)

5.8 Lua chunk registers

`\newluachunkname` As for bytecode registers, but in addition we need to add a string to the `lua.name` table to use in stack tracing. We use the name of the command passed to the allocator, with no backslash.

```
197 \ifx\e@alloc@luachunk@count\@undefined
198   \countdef\e@alloc@luachunk@count=263
199   \e@alloc@luachunk@count=\z@
200 \fi
201 \def\newluachunkname#1{%
202   \e@alloc@luachunk\e@alloc@chardef
203   \e@alloc@luachunk@count\m@ne\e@alloc@top#1%
204   {\escapechar\m@ne
205    \directlua{lua.name[\the\allocationnumber]="\string#1"}}%
206 }
```

(End definition for \newluachunkname.)

5.9 Lua loader

Lua code loaded in the format often has to be loaded again at the beginning of every job, so we define a helper which allows us to avoid duplicated code:

```
207 \def\now@and@everyjob#1{%
208   \everyjob\expandafter{\the\everyjob
209     #1%
```

```

210 }%
211 #1%
212 }

```

Load the Lua code at the start of every job. For the conversion of T_EX into numbers at the Lua side we need some known registers: for convenience we use a set of systematic names, which means using a group around the Lua loader.

```

213 <2ekernel>\now@and@everyjob{%
214   \begingroup
215     \attributedef\attributezero=0 %
216     \chardef      \charzero      =0 %

```

Note name change required on older luatex, for hash table access.

```

217     \countdef      \CountZero    =0 %
218     \dimendef      \dimenzero    =0 %
219     \mathchardef    \mathcharzero=0 %
220     \muskipdef      \muskipzero   =0 %
221     \skipdef        \skipzero     =0 %
222     \toksdef        \tokszero     =0 %
223     \directlua{require("ltnatex")}
224   \endgroup
225 <2ekernel>}
226 <latexrelease>\EndIncludeInRelease

227 <latexrelease>\IncludeInRelease{0000/00/00}
228 <latexrelease>          {\newluafunction}{LuaTeX}%
229 <latexrelease>\let\@alloc@attribute@count\@undefined
230 <latexrelease>\let\newattribute\@undefined
231 <latexrelease>\let\setattribute\@undefined
232 <latexrelease>\let\unsetattribute\@undefined
233 <latexrelease>\let\@alloc@ccodetable@count\@undefined
234 <latexrelease>\let\newcatcodetable\@undefined
235 <latexrelease>\let\catcodetable@initex\@undefined
236 <latexrelease>\let\catcodetable@string\@undefined
237 <latexrelease>\let\catcodetable@latex\@undefined
238 <latexrelease>\let\catcodetable@atletter\@undefined
239 <latexrelease>\let\@alloc@luafunction@count\@undefined
240 <latexrelease>\let\newluafunction\@undefined
241 <latexrelease>\let\@alloc@luafunction@count\@undefined
242 <latexrelease>\let\newwhatsit\@undefined
243 <latexrelease>\let\@alloc@whatsit@count\@undefined
244 <latexrelease>\let\newluabytcode\@undefined
245 <latexrelease>\let\@alloc@bytcode@count\@undefined
246 <latexrelease>\let\newluachunkname\@undefined
247 <latexrelease>\let\@alloc@luachunk@count\@undefined
248 <latexrelease>\directlua{luatexbase.uninstall()}
249 <latexrelease>\EndIncludeInRelease

```

In `\everyjob`, if luaotfload is available, load it and switch to TU.

```

250 <latexrelease>\IncludeInRelease{2017/01/01}%
251 <latexrelease>          {\fontencoding}{TU in everyjob}%
252 <latexrelease>\fontencoding{TU}\let\encodingdefault\fontencoding
253 <latexrelease>\ifx\directlua\@undefined\else
254 <2ekernel>\everyjob\expandafter{%
255 <2ekernel>  \the\everyjob

```

```

256 <*2kernel, latexrelease>
257   \directlua{%
258   if xpcall(function ()%
259             require('luaotfload-main')%
260             end, texio.write_nl) then %
261     local _void = luaotfload.main ()%
262     else %
263     texio.write_nl('Error in luaotfload: reverting to OT1')%
264     tex.print('\string\def\string\encodingdefault{OT1}')%
265     end %
266   }%
267   \let\f@encoding\encodingdefault
268   \expandafter\let\csname ver@luaotfload.sty\endcsname\fmtversion
269 </2kernel, latexrelease>
270 <latexrelease>\fi
271 <2kernel> }
272 <latexrelease>\EndIncludeInRelease
273 <latexrelease>\IncludeInRelease{0000/00/00}%
274 <latexrelease>          {\fontencoding}{TU in everyjob}%
275 <latexrelease>\fontencoding{OT1}\let\encodingdefault\f@encoding
276 <latexrelease>\EndIncludeInRelease
277 <2kernel | latexrelease>\fi
278 </2kernel | tex | latexrelease>

```

5.10 Lua module preliminaries

```

279 <*lua>

```

Some set up for the Lua module which is needed for all of the Lua functionality added here.

luatexbase Set up the table for the returned functions. This is used to expose all of the public functions.

```

280 luatexbase      = luatexbase or { }
281 local luatexbase = luatexbase

```

(End definition for luatexbase.)

Some Lua best practice: use local versions of functions where possible.

```

282 local string_gsub      = string.gsub
283 local tex_count        = tex.count
284 local tex_setattribute = tex.setattribute
285 local tex_setcount     = tex.setcount
286 local texio_write_nl   = texio.write_nl

287 local luatexbase_warning
288 local luatexbase_error

```

5.11 Lua module utilities

5.11.1 Module tracking

modules To allow tracking of module usage, a structure is provided to store information and to return it.

```

289 local modules = modules or { }

```

(End definition for modules.)

`provides_module` Local function to write to the log.

```
290 local function luatexbase_log(text)
291     texio_write_nl("log", text)
292 end
```

Modelled on `\ProvidesPackage`, we store much the same information but with a little more structure.

```
293 local function provides_module(info)
294     if not (info and info.name) then
295         luatexbase_error("Missing module name for provides_module")
296     end
297     local function spaced(text)
298         return text and (" " .. text) or ""
299     end
300     luatexbase_log(
301         "Lua module: " .. info.name
302         .. spaced(info.date)
303         .. spaced(info.version)
304         .. spaced(info.description)
305     )
306     modules[info.name] = info
307 end
308 luatexbase.provides_module = provides_module
```

(End definition for provides_module.)

5.11.2 Module messages

There are various warnings and errors that need to be given. For warnings we can get exactly the same formatting as from \TeX . For errors we have to make some changes. Here we give the text of the error in the \LaTeX format then force an error from Lua to halt the run. Splitting the message text is done using `\n` which takes the place of `\MessageBreak`.

First an auxiliary for the formatting: this measures up the message leader so we always get the correct indent.

```
309 local function msg_format(mod, msg_type, text)
310     local leader = ""
311     local cont
312     local first_head
313     if mod == "LaTeX" then
314         cont = string_gsub(leader, ".", " ")
315         first_head = leader .. "LaTeX: "
316     else
317         first_head = leader .. "Module " .. msg_type
318         cont = "(" .. mod .. ")"
319         .. string_gsub(first_head, ".", " ")
320         first_head = leader .. "Module " .. mod .. " " .. msg_type .. ":"
321     end
322     if msg_type == "Error" then
323         first_head = "\n" .. first_head
324     end
325     if string.sub(text,-1) ~= "\n" then
```

```

326     text = text .. " "
327 end
328 return first_head .. " "
329 .. string_gsub(
330     text
331 .. "on input line "
332     .. tex.inputlineno, "\n", "\n" .. cont .. " "
333 )
334 .. "\n"
335 end

module_info Write messages.
module_warning
module_error
336 local function module_info(mod, text)
337     texio_write_nl("log", msg_format(mod, "Info", text))
338 end
339 luatexbase.module_info = module_info
340 local function module_warning(mod, text)
341     texio_write_nl("term and log", msg_format(mod, "Warning", text))
342 end
343 luatexbase.module_warning = module_warning
344 local function module_error(mod, text)
345     error(msg_format(mod, "Error", text))
346 end
347 luatexbase.module_error = module_error

(End definition for module_info, module_warning, and module_error.)

Dedicated versions for the rest of the code here.

348 function luatexbase_warning(text)
349     module_warning("luatexbase", text)
350 end
351 function luatexbase_error(text)
352     module_error("luatexbase", text)
353 end

```

5.12 Accessing register numbers from Lua

Collect up the data from the T_EX level into a Lua table: from version 0.80, LuaT_EX makes that easy.

```

354 local luaregisterbasetable = { }
355 local registermap = {
356     attributezero = "assign_attr"    ,
357     charzero      = "char_given"    ,
358     CountZero     = "assign_int"    ,
359     dimenzero     = "assign_dimen"  ,
360     mathcharzero  = "math_given"    ,
361     muskipzero    = "assign_mu_skip",
362     skipzero      = "assign_skip"   ,
363     tokszero      = "assign_toks"   ,
364 }
365 local createtoken
366 if tex.luatexversion > 81 then
367     createtoken = token.create
368 elseif tex.luatexversion > 79 then

```

```

369   createtoken = newtoken.create
370 end
371 local hashtokens = tex.hashtokens()
372 local luatexversion = tex.luatexversion
373 for i,j in pairs (registermap) do
374   if luatexversion < 80 then
375     luaregisterbasetable[hashtokens[i][1]] =
376       hashtokens[i][2]
377   else
378     luaregisterbasetable[j] = createtoken(i).mode
379   end
380 end

```

registernumber Working out the correct return value can be done in two ways. For older LuaTeX releases it has to be extracted from the `hashtokens`. On the other hand, newer LuaTeX's have `newtoken`, and whilst `.mode` isn't currently documented, Hans Hagen pointed to this approach so we should be OK.

```

381 local registernumber
382 if luatexversion < 80 then
383   function registernumber(name)
384     local nt = hashtokens[name]
385     if(nt and luaregisterbasetable[nt[1]]) then
386       return nt[2] - luaregisterbasetable[nt[1]]
387     else
388       return false
389     end
390   end
391 else
392   function registernumber(name)
393     local nt = createtoken(name)
394     if(luaregisterbasetable[nt.cmdname]) then
395       return nt.mode - luaregisterbasetable[nt.cmdname]
396     else
397       return false
398     end
399   end
400 end
401 luatexbase.registernumber = registernumber

```

(End definition for registernumber.)

5.13 Attribute allocation

new_attribute As attributes are used for Lua manipulations its useful to be able to assign from this end.

```

402 local attributes=setmetatable(
403   {},
404   {
405     __index = function(t,key)
406       return registernumber(key) or nil
407     end}
408 )
409 luatexbase.attributes = attributes

```

```

410 local attribute_count_name =
411     attribute_count_name or "e@alloc@attribute@count"
412 local function new_attribute(name)
413     tex_setcount("global", attribute_count_name,
414         tex_count[attribute_count_name] + 1)
415     if tex_count[attribute_count_name] > 65534 then
416         luatexbase_error("No room for a new \\attribute")
417     end
418     attributes[name]= tex_count[attribute_count_name]
419     luatexbase_log("Lua-only attribute " .. name .. " = " ..
420         tex_count[attribute_count_name])
421     return tex_count[attribute_count_name]
422 end
423 luatexbase.new_attribute = new_attribute

```

(End definition for *new_attribute*.)

5.14 Custom whatsit allocation

new_whatsit Much the same as for attribute allocation in Lua.

```

424 local whatsit_count_name = whatsit_count_name or "e@alloc@whatsit@count"
425 local function new_whatsit(name)
426     tex_setcount("global", whatsit_count_name,
427         tex_count[whatsit_count_name] + 1)
428     if tex_count[whatsit_count_name] > 65534 then
429         luatexbase_error("No room for a new custom whatsit")
430     end
431     luatexbase_log("Custom whatsit " .. (name or "") .. " = " ..
432         tex_count[whatsit_count_name])
433     return tex_count[whatsit_count_name]
434 end
435 luatexbase.new_whatsit = new_whatsit

```

(End definition for *new_whatsit*.)

5.15 Bytecode register allocation

new_bytecode Much the same as for attribute allocation in Lua. The optional *<name>* argument is used in the log if given.

```

436 local bytecode_count_name =
437     bytecode_count_name or "e@alloc@bytecode@count"
438 local function new_bytecode(name)
439     tex_setcount("global", bytecode_count_name,
440         tex_count[bytecode_count_name] + 1)
441     if tex_count[bytecode_count_name] > 65534 then
442         luatexbase_error("No room for a new bytecode register")
443     end
444     luatexbase_log("Lua bytecode " .. (name or "") .. " = " ..
445         tex_count[bytecode_count_name])
446     return tex_count[bytecode_count_name]
447 end
448 luatexbase.new_bytecode = new_bytecode

```

(End definition for *new_bytecode*.)

5.16 Lua chunk name allocation

`new_chunkname` As for bytecode registers but also store the name in the `lua.name` table.

```
449 local chunkname_count_name =
450     chunkname_count_name or "e@alloc@luachunk@count"
451 local function new_chunkname(name)
452     tex_setcount("global", chunkname_count_name,
453         tex_count[chunkname_count_name] + 1)
454     local chunkname_count = tex_count[chunkname_count_name]
455     chunkname_count = chunkname_count + 1
456     if chunkname_count > 65534 then
457         luatexbase_error("No room for a new chunkname")
458     end
459     lua.name[chunkname_count]=name
460     luatexbase_log("Lua chunkname " .. (name or "") .. " = " ..
461         chunkname_count .. "\n")
462     return chunkname_count
463 end
464 luatexbase.new_chunkname = new_chunkname
```

(End definition for new_chunkname.)

5.17 Lua function allocation

`new_luafunction` Much the same as for attribute allocation in Lua. The optional $\langle name \rangle$ argument is used in the log if given.

```
465 local luafunction_count_name =
466     luafunction_count_name or "e@alloc@luafunction@count"
467 local function new_luafunction(name)
468     tex_setcount("global", luafunction_count_name,
469         tex_count[luafunction_count_name] + 1)
470     if tex_count[luafunction_count_name] > 65534 then
471         luatexbase_error("No room for a new luafunction register")
472     end
473     luatexbase_log("Lua function " .. (name or "") .. " = " ..
474         tex_count[luafunction_count_name])
475     return tex_count[luafunction_count_name]
476 end
477 luatexbase.new_luafunction = new_luafunction
```

(End definition for new_luafunction.)

5.18 Lua callback management

The native mechanism for callbacks in LuaTeX allows only one per function. That is extremely restrictive and so a mechanism is needed to add and remove callbacks from the appropriate hooks.

5.18.1 Housekeeping

The main table: keys are callback names, and values are the associated lists of functions. More precisely, the entries in the list are tables holding the actual function as `func` and

the identifying description as `description`. Only callbacks with a non-empty list of functions have an entry in this list.

```
478 local callbacklist = callbacklist or { }
```

Numerical codes for callback types, and name-to-value association (the table keys are strings, the values are numbers).

```
479 local list, data, exclusive, simple, reverselist = 1, 2, 3, 4, 5
480 local types = {
481   list      = list,
482   data      = data,
483   exclusive = exclusive,
484   simple    = simple,
485   reverselist = reverselist,
486 }
```

Now, list all predefined callbacks with their current type, based on the LuaTeX manual version 1.01. A full list of the currently-available callbacks can be obtained using

```
\directlua{
  for i,_ in pairs(callback.list()) do
    texio.write_nl("- " .. i)
  end
}
\bye
```

in plain LuaTeX. (Some undocumented callbacks are omitted as they are to be removed.)

```
487 local callbacktypes = callbacktypes or {
```

Section 8.2: file discovery callbacks.

```
488   find_read_file      = exclusive,
489   find_write_file     = exclusive,
490   find_font_file      = data,
491   find_output_file    = data,
492   find_format_file    = data,
493   find_vf_file        = data,
494   find_map_file       = data,
495   find_enc_file       = data,
496   find_pk_file        = data,
497   find_data_file      = data,
498   find_opentype_file  = data,
499   find_truetype_file  = data,
500   find_type1_file     = data,
501   find_image_file     = data,

502   open_read_file      = exclusive,
503   read_font_file      = exclusive,
504   read_vf_file        = exclusive,
505   read_map_file       = exclusive,
506   read_enc_file       = exclusive,
507   read_pk_file        = exclusive,
508   read_data_file      = exclusive,
509   read_truetype_file  = exclusive,
510   read_type1_file     = exclusive,
511   read_opentype_file  = exclusive,
```

Not currently used by luatex but included for completeness. may be used by a font handler.

```
512 find_cidmap_file = data,  
513 read_cidmap_file = exclusive,
```

Section 8.3: data processing callbacks.

```
514 process_input_buffer = data,  
515 process_output_buffer = data,  
516 process_jobname = data,
```

Section 8.4: node list processing callbacks.

```
517 contribute_filter = simple,  
518 buildpage_filter = simple,  
519 build_page_insert = exclusive,  
520 pre_linebreak_filter = list,  
521 linebreak_filter = exclusive,  
522 append_to_vlist_filter = exclusive,  
523 post_linebreak_filter = reverselist,  
524 hpack_filter = list,  
525 vpack_filter = list,  
526 hpack_quality = list,  
527 vpack_quality = list,  
528 pre_output_filter = list,  
529 process_rule = exclusive,  
530 hyphenate = simple,  
531 ligaturing = simple,  
532 kerning = simple,  
533 insert_local_par = simple,  
534 pre_mlist_to_hlist_filter = list,  
535 mlist_to_hlist = exclusive,  
536 post_mlist_to_hlist_filter = reverselist,  
537 new_graf = exclusive,
```

Section 8.5: information reporting callbacks.

```
538 pre_dump = simple,  
539 start_run = simple,  
540 stop_run = simple,  
541 start_page_number = simple,  
542 stop_page_number = simple,  
543 show_error_hook = simple,  
544 show_warning_message = simple,  
545 show_error_message = simple,  
546 show_lua_error_hook = simple,  
547 start_file = simple,  
548 stop_file = simple,  
549 call_edit = simple,  
550 finish_synctex = simple,  
551 wrapup_run = simple,
```

Section 8.6: PDF-related callbacks.

```
552 finish_pdffile = data,  
553 finish_pdfpage = data,  
554 page_objnum_provider = data,  
555 page_order_index = data,  
556 process_pdf_image_content = data,
```

Section 8.7: font-related callbacks.

```
557 define_font                = exclusive,
558 glyph_info                  = exclusive,
559 glyph_not_found              = exclusive,
560 glyph_stream_provider        = exclusive,
561 make_extensible              = exclusive,
562 font_descriptor_objnum_provider = exclusive,
563 input_level_string            = exclusive,
564 }
565 luatexbase.callbacktypes=callbacktypes
```

callback.register Save the original function for registering callbacks and prevent the original being used. The original is saved in a place that remains available so other more sophisticated code can override the approach taken by the kernel if desired.

```
566 local callback_register = callback_register or callback.register
567 function callback.register()
568   luatexbase_error("Attempt to use callback.register() directly\n")
569 end
```

(End definition for *callback.register*.)

5.18.2 Handlers

The handler function is registered into the callback when the first function is added to this callback's list. Then, when the callback is called, the handler takes care of running all functions in the list. When the last function is removed from the callback's list, the handler is unregistered.

More precisely, the functions below are used to generate a specialized function (closure) for a given callback, which is the actual handler.

The way the functions are combined together depends on the type of the callback. There are currently 4 types of callback, depending on the calling convention of the functions the callback can hold:

simple is for functions that don't return anything: they are called in order, all with the same argument;

data is for functions receiving a piece of data of any type except node list head (and possibly other arguments) and returning it (possibly modified): the functions are called in order, and each is passed the return value of the previous (and the other arguments untouched, if any). The return value is that of the last function;

list is a specialized variant of *data* for functions filtering node lists. Such functions may return either the head of a modified node list, or the boolean values **true** or **false**. The functions are chained the same way as for *data* except that for the following. If one function returns **false**, then **false** is immediately returned and the following functions are *not* called. If one function returns **true**, then the same head is passed to the next function. If all functions return **true**, then **true** is returned, otherwise the return value of the last function not returning **true** is used.

reverselist is a specialized variant of *list* which executes functions in inverse order.

exclusive is for functions with more complex signatures; functions in this type of callback are *not* combined: An error is raised if a second callback is registered.

Handler for data callbacks.

```
570 local function data_handler(name)
571   return function(data, ...)
572     for _,i in ipairs(callbacklist[name]) do
573       data = i.func(data,...)
574     end
575     return data
576   end
577 end
```

Default for user-defined data callbacks without explicit default.

```
578 local function data_handler_default(value)
579   return value
580 end
```

Handler for exclusive callbacks. We can assume `callbacklist[name]` is not empty: otherwise, the function wouldn't be registered in the callback any more.

```
581 local function exclusive_handler(name)
582   return function(...)
583     return callbacklist[name][1].func(...)
584   end
585 end
```

Handler for list callbacks.

```
586 local function list_handler(name)
587   return function(head, ...)
588     local ret
589     local alltrue = true
590     for _,i in ipairs(callbacklist[name]) do
591       ret = i.func(head, ...)
592       if ret == false then
593         luatexbase_warning(
594           "Function '" .. i.description .. "' returned false\n"
595           .. "in callback '" .. name .. "'")
596       end
597       return false
598     end
599     if ret ~= true then
600       alltrue = false
601       head = ret
602     end
603   end
604   return alltrue and true or head
605 end
606 end
```

Default for user-defined list and reverselist callbacks without explicit default.

```
607 local function list_handler_default()
608   return true
609 end
```

Handler for reverselist callbacks.

```
610 local function reverselist_handler(name)
611   return function(head, ...)
612     local ret
613     local alltrue = true
```

```

614     local callbacks = callbacklist[name]
615     for i = #callbacks, 1, -1 do
616         local cb = callbacks[i]
617         ret = cb.func(head, ...)
618         if ret == false then
619             luatexbase_warning(
620                 "Function '" .. cb.description .. "' returned false\n"
621                 .. "in callback '" .. name .. "'")
622             )
623             return false
624         end
625         if ret ~= true then
626             alltrue = false
627             head = ret
628         end
629     end
630     return alltrue and true or head
631 end
632 end

```

Handler for simple callbacks.

```

633 local function simple_handler(name)
634     return function(...)
635         for _,i in ipairs(callbacklist[name]) do
636             i.func(...)
637         end
638     end
639 end

```

Default for user-defined simple callbacks without explicit default.

```

640 local function simple_handler_default()
641 end

```

Keep a handlers table for indexed access and a table with the corresponding default functions.

```

642 local handlers = {
643     [data] = data_handler,
644     [exclusive] = exclusive_handler,
645     [list] = list_handler,
646     [reverselist] = reverselist_handler,
647     [simple] = simple_handler,
648 }
649 local defaults = {
650     [data] = data_handler_default,
651     [exclusive] = nil,
652     [list] = list_handler_default,
653     [reverselist] = list_handler_default,
654     [simple] = simple_handler_default,
655 }

```

5.18.3 Public functions for callback management

Defining user callbacks perhaps should be in package code, but impacts on `add_to_callback`. If a default function is not required, it may be declared as `false`. First we need a list of user callbacks.

```

656 local user_callbacks_defaults = {
657   pre_mlist_to_hlist_filter = list_handler_default,
658   mlist_to_hlist = node.mlist_to_hlist,
659   post_mlist_to_hlist_filter = list_handler_default,
660 }

```

`create_callback` The allocator itself.

```

661 local function create_callback(name, ctype, default)
662   local ctype_id = types[ctype]
663   if not name or name == ""
664   or not ctype_id
665   then
666     luatexbase_error("Unable to create callback:\n" ..
667                       "valid callback name and type required")
668   end
669   if callbacktypes[name] then
670     luatexbase_error("Unable to create callback '" .. name ..
671                       "':\ncallback is already defined")
672   end
673   default = default or defaults[ctype_id]
674   if not default then
675     luatexbase_error("Unable to create callback '" .. name ..
676                       "':\ndefault is required for '" .. ctype ..
677                       "': callbacks")
678   elseif type(default) ~= "function" then
679     luatexbase_error("Unable to create callback '" .. name ..
680                       "':\ndefault is not a function")
681   end
682   user_callbacks_defaults[name] = default
683   callbacktypes[name] = ctype_id
684 end
685 luatexbase.create_callback = create_callback

```

(End definition for create_callback.)

`call_callback` Call a user defined callback. First check arguments.

```

686 local function call_callback(name,...)
687   if not name or name == "" then
688     luatexbase_error("Unable to create callback:\n" ..
689                       "valid callback name required")
690   end
691   if user_callbacks_defaults[name] == nil then
692     luatexbase_error("Unable to call callback '" .. name
693                       .. "':\nunknown or empty")
694   end
695   local l = callbacklist[name]
696   local f
697   if not l then
698     f = user_callbacks_defaults[name]
699   else
700     f = handlers[callbacktypes[name]](name)
701   end
702   return f(...)
703 end
704 luatexbase.call_callback=call_callback

```

(End definition for `call_callback`.)

`add_to_callback` Add a function to a callback. First check arguments.

```
705 local function add_to_callback(name, func, description)
706   if not name or name == "" then
707     luatexbase_error("Unable to register callback:\n" ..
708                       "valid callback name required")
709   end
710   if not callbacktypes[name] or
711     type(func) ~= "function" or
712     not description or
713     description == "" then
714     luatexbase_error(
715       "Unable to register callback.\n\n"
716       .. "Correct usage:\n"
717       .. "add_to_callback(<callback>, <function>, <description>)"
718     )
719   end
```

Then test if this callback is already in use. If not, initialise its list and register the proper handler.

```
720   local l = callbacklist[name]
721   if l == nil then
722     l = { }
723     callbacklist[name] = l
```

If it is not a user defined callback use the primitive callback register.

```
724     if user_callbacks_defaults[name] == nil then
725       callback_register(name, handlers[callbacktypes[name]](name))
726     end
727   end
```

Actually register the function and give an error if more than one exclusive one is registered.

```
728   local f = {
729     func      = func,
730     description = description,
731   }
732   local priority = #l + 1
733   if callbacktypes[name] == exclusive then
734     if #l == 1 then
735       luatexbase_error(
736         "Cannot add second callback to exclusive function\n'" ..
737         name .. "'")
738     end
739   end
740   table.insert(l, priority, f)
```

Keep user informed.

```
741   luatexbase_log(
742     "Inserting '" .. description .. "' at position "
743     .. priority .. " in '" .. name .. "'."
744   )
745 end
746 luatexbase.add_to_callback = add_to_callback
```

(End definition for add_to_callback.)

`remove_from_callback` Remove a function from a callback. First check arguments.

```
747 local function remove_from_callback(name, description)
748   if not name or name == "" then
749     luatexbase_error("Unable to remove function from callback:\n" ..
750                       "valid callback name required")
751   end
752   if not callbacktypes[name] or
753     not description or
754     description == "" then
755     luatexbase_error(
756       "Unable to remove function from callback.\n\n"
757       .. "Correct usage:\n"
758       .. "remove_from_callback(<callback>, <description>)"
759     )
760   end
761   local l = callbacklist[name]
762   if not l then
763     luatexbase_error(
764       "No callback list for '" .. name .. "'\n")
765   end
```

Loop over the callback's function list until we find a matching entry. Remove it and check if the list is empty: if so, unregister the callback handler.

```
766   local index = false
767   for i,j in ipairs(l) do
768     if j.description == description then
769       index = i
770       break
771     end
772   end
773   if not index then
774     luatexbase_error(
775       "No callback '" .. description .. "' registered for '" ..
776       name .. "'\n")
777   end
778   local cb = l[index]
779   table.remove(l, index)
780   luatexbase_log(
781     "Removing '" .. description .. "' from '" .. name .. "'."
782   )
783   if #l == 0 then
784     callbacklist[name] = nil
785     if user_callbacks_defaults[name] == nil then
786       callback_register(name, nil)
787     end
788   end
789   return cb.func,cb.description
790 end
791 luatexbase.remove_from_callback = remove_from_callback
```

(End definition for remove_from_callback.)

in_callback Look for a function description in a callback.

```
792 local function in_callback(name, description)
793   if not name
794     or name == ""
795     or not callbacklist[name]
796     or not callbacktypes[name]
797     or not description then
798     return false
799   end
800   for _, i in pairs(callbacklist[name]) do
801     if i.description == description then
802       return true
803     end
804   end
805   return false
806 end
807 luatexbase.in_callback = in_callback
```

(End definition for in_callback.)

disable_callback As we subvert the engine interface we need to provide a way to access this functionality.

```
808 local function disable_callback(name)
809   if(callbacklist[name] == nil) then
810     callback_register(name, false)
811   else
812     luatexbase_error("Callback list for " .. name .. " not empty")
813   end
814 end
815 luatexbase.disable_callback = disable_callback
```

(End definition for disable_callback.)

callback_descriptions List the descriptions of functions registered for the given callback.

```
816 local function callback_descriptions (name)
817   local d = {}
818   if not name
819     or name == ""
820     or not callbacklist[name]
821     or not callbacktypes[name]
822   then
823     return d
824   else
825     for k, i in pairs(callbacklist[name]) do
826       d[k] = i.description
827     end
828   end
829   return d
830 end
831 luatexbase.callback_descriptions = callback_descriptions
```

(End definition for callback_descriptions.)

uninstall Unlike at the \TeX level, we have to provide a back-out mechanism here at the same time as the rest of the code. This is not meant for use by anything other than `latexrelease`: as such this is *deliberately* not documented for users!

```

832 local function uninstall()
833   module_info(
834     "luatexbase",
835     "Uninstalling kernel luatexbase code"
836   )
837   callback.register = callback_register
838   luatexbase = nil
839 end
840 luatexbase.uninstall = uninstall

```

(End definition for uninstall.)

mlist_to_hlist To emulate these callbacks, the “real” `mlist_to_hlist` is replaced by a wrapper calling the wrappers before and after.

```

841 callback_register("mlist_to_hlist", function(head, display_type, need_penalties)
842   local current = call_callback("pre_mlist_to_hlist_filter", head, display_type, need_penalties)
843   if current == false then
844     flush_list(head)
845     return nil
846   elseif current == true then
847     current = head
848   end
849   current = call_callback("mlist_to_hlist", current, display_type, need_penalties)
850   local post = call_callback("post_mlist_to_hlist_filter", current, display_type, need_penalties)
851   if post == true then
852     return current
853   elseif post == false then
854     flush_list(current)
855     return nil
856   end
857   return post
858 end)

```

(End definition for mlist_to_hlist.)

```

859 </lua>

```

Reset the catcode of @.

```

860 <tex>\catcode'\@=\etacatcode\relax

```

File e

ltxexpl.dtx

1 expl3-dependent code

1.1 Loader

`\@kernel@after@enddocument` These two kernel hooks are used by the shipout code. They are defined earlier here
`\@kernel@after@enddocument@afterlastpage` because the `lthooks` code adds material to them.

```

1 <*2ekernel | latexrelease>
2 <latexrelease> \IncludeInRelease{2020/10/01}%
3 <latexrelease> {kernel@enddocument hooks}{Define several kernel hooks}

```

We only initialize these kernel hooks if they are not already existing. Otherwise they would be set to `\@empty` on rollback which would be wrong because code that has been added to to them may still have to be executed in the rollback situation . Instead code that writes to them needs to handle the rollback as needed. It is likely that we have to change that approach in the future, but for now it should do. (It is enough to test only for the existence of one hook, as all got added at the same time.)

```

4 \ifx\@kernel@after@enddocument\undefined
5   \let \@kernel@after@enddocument \@empty
6   \let \@kernel@after@enddocument@afterlastpage \@empty

```

For the similar reasons we also define those that are used in `\document` because they too get material added to in early modules.

```

\@kernel@before@begindocument
\@kernel@after@begindocument
7   \let \@kernel@before@begindocument \@empty
8   \let \@kernel@after@begindocument \@empty
9   \fi
10  <latexrelease> \EndIncludeInRelease
11  <latexrelease> \IncludeInRelease{0000/00/00}%
12  <latexrelease> {kernel@enddocument hooks}{Define several kernel hooks}
13  <latexrelease> \let \@kernel@after@enddocument\undefined
14  <latexrelease> \let \@kernel@after@enddocument@afterlastpage\undefined
15  <latexrelease> \let \@kernel@before@begindocument\undefined
16  <latexrelease> \let \@kernel@after@begindocument\undefined
17  </2ekernel | latexrelease>
18  <latexrelease> \EndIncludeInRelease

```

(End definition for `\@kernel@after@enddocument` and others.)

First define some blank commands, so that in case something goes wrong while loading `expl3`, we won't get strange `Undefined control sequence` errors.

```

19 <*2ekernel | latexrelease>
20 <latexrelease> \IncludeInRelease{2020/10/01}%
21 <latexrelease> {\@expl@sys@load@backend@@}{Roll forward support}%
22 \def\reserved@a#1{\ifdefined#1\else\def#1{}\fi}
23 \reserved@a\@expl@sys@load@backend@@
24 \reserved@a\@expl@push@filename@@
25 \reserved@a\@expl@push@filename@aux@@
26 \reserved@a\@expl@pop@filename@@
27 <latexrelease> \EndIncludeInRelease
28 </2ekernel | latexrelease>

```

Create a hook for last-minute expl3 material.

```

29 <*2kernel>
30 \def\@expl@finalise@setup@{
31 </2kernel>

```

Now define some basics to support loading expl3. These macros can be defined here safely, because they are redefined later on by the kernel, so we define simpler versions just to suit our needs.

```

32 <*2kernel>
33 \long\def\@gobble#1{}
34 \long\def\@firstofone#1{#1}
35 \long\def\@firstoftwo#1#2{#1}
36 \long\def\@secondoftwo#1#2{#2}
37 \long\def\IfFileExists#1{%
38   \openin\@inputcheck"#1" %
39   \ifeof\@inputcheck
40     \expandafter\@secondoftwo
41   \else
42     \closein\@inputcheck
43     \expandafter\@firstoftwo
44   \fi}
45 \long\def\@ifnextchar#1#2#3{%
46   \let\reserved@d=#1%
47   \def\reserved@a{#2}%
48   \def\reserved@b{#3}%
49   \futurelet\@let@token\@ifnch}
50 \def\@ifnch{%
51   \ifx\@let@token\reserved@d
52     \expandafter\reserved@a
53   \else
54     \expandafter\reserved@b
55   \fi}
56 </2kernel>

```

If we are doing a rollback with a format containing expl3 we aren't reloading it as that creates havoc. This may need a refined version!

```

57 <*2kernel | latexrelease>
58 <latexrelease> \IncludeInRelease{2020/10/01}%
59 <latexrelease> {expl3}{Pre-load expl3}%
60 \expandafter\ifx\csname tex\string _let:D\endcsname\relax
61   \expandafter\@firstofone
62 \else
63   \GenericInfo{}{Skipping: expl3 code already part of the format}%
64 <2kernel> \expandafter\endinput
65 <latexrelease> \expandafter\@gobble
66 \fi

```

Check for the required primitive/engine support and the existence of a loader.

```

67 {%
68   \IfFileExists{expl3.ltx}
69   {%
70     \ifnum0%
71       \ifdefined\pdffilesize 1\fi
72       \ifdefined\filesize 1\fi
73       \ifdefined\luatexversion\ifnum\luatexversion>94 1\fi\fi

```

```

74         \ifdefined\kanjiskip 1\fi
75         >0 %
76         \expandafter\@firstofone
77     \else

```

In 2ekernel mode, an error is fatal and building the format is aborted. Use `\batchmode \read -1 to \tokenlist`, which errors with

! Emergency stop. (cannot \read from terminal in nonstop modes)

and aborts the T_EX run. In latexrelease mode, raise an error and do nothing. Both ways, the error message shows the minimum expl3 engine requirements.

```

78 <2ekernel>         \def~{ }\def\MessageBreak{^^J~~~~~}%
79 <2ekernel>         \errmessage{LaTeX Error:
80 <latexrelease>     \@latex@error{%
81                     LaTeX requires the e-TeX primitives and additional\MessageBreak
82                     functionality available in the engines:\MessageBreak
83                     - pdfTeX v1.40\MessageBreak
84                     - XeTeX v0.99992\MessageBreak
85                     - LuaTeX v0.95\MessageBreak
86                     - e-(u)pTeX mid-2012\MessageBreak
87                     or later}%
88 <latexrelease>     }\@ehd \expandafter\@gobble
89 <2ekernel>         }\batchmode \read -1 to \reserved@a
90     \fi
91 }
92 {%
93 <*2ekernel>
94     \errmessage{LaTeX requires expl3}%
95     \batchmode \read -1 to \reserved@a
96 </2ekernel>

```

We do not support a roll forward across 2019. You need to start with 2019 if you want to get to 2020 or beyond.

```

97 <*latexrelease>
98     \@latex@warning@no@line
99     {You need a format that already contains a recent\MessageBreak
100     expl3 as part of the kernel, e.g. at least a kernel\MessageBreak
101     from 2019 to roll forward to that date!\MessageBreak
102     --- I'm giving up!\MessageBreak\MessageBreak
103     Note that manually loading the expl3 package\MessageBreak
104     from your distribution is not enough}%
105     \batchmode \read -1 to \reserved@a
106 </latexrelease>
107 }%
108 {\input expl3.ltx}%
109 }
110 <latexrelease>\EndIncludeInRelease
111 <latexrelease>

```

To support roll-forward for the case where xparse is fully integrated into the kernel, we do not need to repeat the complex test above as we can simply look for the marker command.

```

112 <latexrelease>\IncludeInRelease{2020/02/02}%
113 <latexrelease>     {expl3}{Pre-load expl3}%

```

```

114 <latexrelease>\IfFileExists{expl3.ltx}
115 <latexrelease> {%
116 <latexrelease>     \ifnum0%
117 <latexrelease>         \ifdefined\pdffilesize 1\fi
118 <latexrelease>         \ifdefined\filesize 1\fi
119 <latexrelease>         \ifdefined\luatexversion\ifnum\luatexversion>94 1\fi\fi
120 <latexrelease>         >0 %
121 <latexrelease>     \else
122 <latexrelease>         \message{Skipping expl3-dependent extensions}
123 <latexrelease>         \expandafter\@gobbletwo
124 <latexrelease>     \fi
125 <latexrelease> }
126 <latexrelease> {%
127 <latexrelease>     \message{Skipping expl3-dependent extensions}%
128 <latexrelease>     \@gobbletwo
129 <latexrelease> }%
130 <latexrelease>\input{expl3.ltx}
131 <latexrelease>\EndIncludeInRelease

```

1.2 Using expl3 code

In order to ease the implementation of some new features in L^AT_EX 2_ε we may (temporarily) use some coding based on the `expl3`-code. Such macros will eventually vanish and may be changed unannounced. They are there for internal use in the L^AT_EX 2_ε kernel and are not meant to be used in third-party packages. These macros will always have the `@expl@` prefix in their name.

The rest of the name matches the `expl3` name but with all underscores replaced by `@`s and the `:` replaced by `@@`, e.g.,

```
\cs_new_eq:NN \@expl@tl@trim@spaces@apply@@nN \tl_trim_spaces_apply:nN
```

if that `expl3` command is needed in places that are others coded in L^AT_EX 2_ε conventions.

In this file, each release of LaTeX adds an `\IncludeInRelease` block, in which the macros copied for that release were defined. In case a rollback is requested, the entire block is changed.

Each macro copied has a `\changes` entry to explain when and why it was copied, so that further to that may spot it easily.

Here `\cs_gset_eq:NN` is used, instead of the `new` variant because if different releases use that same name for different purposes, each can copy the macro without worrying about redefinitions.

```

132 <latexrelease>\IncludeInRelease{2020/10/01}{\@expl@cs@to@str@@N}%
133 <latexrelease>     {expl3 macros added for the 2020-10-01 release}%

```

The `expl3` activation needs to be inside the release guards as otherwise rolling forward is broken in old kernels that do not have `expl3` loaded.

```

134 \ExplSyntaxOn

135 \cs_gset_eq:NN \@expl@cs@to@str@@N \cs_to_str:N
136 \cs_gset_eq:NN \@expl@str@if@eq@@nnTF \str_if_eq:nnTF

137 \cs_gset_eq:NN \@expl@cs@prefix@spec@@N \cs_prefix_spec:N
138 \cs_gset_eq:NN \@expl@cs@argument@spec@@N \cs_argument_spec:N
139 \cs_gset_eq:NN \@expl@cs@replacement@spec@@N \cs_replacement_spec:N

```

```

140 \cs_gset_eq:NN \@expl@str@map@function@@NN \str_map_function:NN
141 \cs_gset_eq:NN \@expl@char@generate@@nn \char_generate:nn
142 \ExplSyntaxOff

```

Here we can't assume that expl3 is available. It will be if we roll back but if this code is executed rolling forward it needs to be pure 2e.

```

143 <latexrelease>\EndIncludeInRelease
144 <latexrelease>\IncludeInRelease{0000/00/00}{\@expl@cs@to@str@@N}%
145 <latexrelease>      {expl3 macros added for the 2020-10-01 release}%
146 <latexrelease>\let \@expl@cs@to@str@@N \@undefined
147 <latexrelease>\let \@expl@str@if@eq@@nnTF \@undefined
148 <latexrelease>\let \@expl@cs@prefix@spec@@N \@undefined
149 <latexrelease>\let \@expl@cs@argument@spec@@N \@undefined
150 <latexrelease>\let \@expl@cs@replacement@spec@@N \@undefined
151 <latexrelease>\let \@expl@str@map@function@@NN \@undefined
152 <latexrelease>\EndIncludeInRelease
153 </2ekernel | latexrelease>

```

File f

ltdefns.dtx

1 Definitions

This section contains commands used in defining other macros.

```
1 <*2ekernel>
```

1.1 Initex initializations

`\two@digits` Prefix a number less than 10 with ‘0’.

```
2 \def\two@digits#1{\ifnum#1<10 0\fi\number#1}
```

(End definition for \two@digits.)

`\typeout` Display something on the terminal.

```
3 </2ekernel>
4 <*2ekernel | latexrelease>
5 <latexrelease> \IncludeInRelease{2020/10/01}%
6 <latexrelease>           {\typeout}{Allow "par" in \typeout}%
7 \protected\long\def\typeout#1{\begingroup
8   \set@display@protect
9   \def\par{^^J^^J}%
10  \immediate\write\@unused{#1}\endgroup}
11 </2ekernel | latexrelease>
12 <latexrelease> \EndIncludeInRelease
13 <latexrelease> \IncludeInRelease{0000/00/00}%
14 <latexrelease>           {\typeout}{Allow "par" in \typeout}%
15 <latexrelease>
16 <latexrelease> \def\typeout#1{\begingroup\set@display@protect
17 <latexrelease>   \immediate\write\@unused{#1}\endgroup}
18 <latexrelease> \EndIncludeInRelease
19 <*2ekernel>
```

(End definition for \typeout.)

`\newlinechar` A char to be used as new-line in output to files.

```
20 \newlinechar‘^^J
```

(End definition for \newlinechar.)

1.2 Saved versions of TeX primitives

The TeX primitive `\foo` is saved as `\@@foo`. The following primitives are handled in this way:

`\@@par`

```
21 \let\@@par=\par
22 %\let\@@input=\input      %%% moved earlier
23 %\let\@@end=\end          %%%
```

(End definition for \@@par.)

`\@@hyph` Save original primitive definition.

```

24 \let\@@hyph=\-

```

(End definition for `\@@hyph`.)

`\@@italiccorr` Save the original italic correction.

```

25 \let\@@italiccorr=\/

```

(End definition for `\@@italiccorr`.)

`\@height` The following definitions save token space. E.g., using `\@height` instead of `height` saves

`\@depth` 5 tokens at the cost in time of one macro expansion.

```

\@width 26 \def\@height{height} \def\@depth{depth} \def\@width{width}
\@minus 27 \def\@minus{minus}
\@plus 28 \def\@plus{plus}

```

The next one is another 100 tokens worth.

```

29 \def\hb@xt@{\hbox to}

```

(End definition for `\@height` and others.)

```

30 \message{hacks,}

```

`\hb@xt@`

1.3 Command definitions

This section defines the following commands:

`\@namedef` `{\NAME}`
Expands to `\def\{\NAME}`, except name can contain any characters.

`\@nameuse` `{\NAME}`
Expands to `\{\NAME}`.

`\@ifnextchar` `X{\YES}{\NO}`
Expands to `\YES` if next character is an ‘X’, and to `\NO` otherwise. (Uses `\reserved@a–\reserved@c`.) NOTE: GOBBLES ANY SPACE FOLLOWING IT.

`\@ifstar` `{\YES}{\NO}`
Gobbles following spaces and then tests if next the character is a ‘*’. If it is, then it gobbles the ‘*’ and expands to `\YES`, otherwise it expands to `\NO`.

`\@dblarg` `{\CMD}{\ARG}`
Expands to `\{\CMD\}[\ARG]{\ARG}`. Use `\@dblarg\CS` when `\CS` takes arguments `[ARG1]{ARG2}`, where default is `ARG1 = ARG2`.

`\@ifundefined` `{\NAME}{\YES}{\NO}`
: If `\NAME` is undefined then it executes `\YES`, otherwise it executes `\NO`. More precisely, true if `\NAME` either undefined or = `\relax`.

`\@ifdefinable` `\NAME{\YES}` Executes `\YES` if the user is allowed to define `\NAME`, otherwise it gives an error. The user can define `\NAME` if `\@ifundefined{\NAME}` is true, ‘`NAME`’ ≠ ‘`relax`’ and the first three letters of ‘`NAME`’ are not ‘`end`’, and if `\endNAME` is not defined.

`\newcommand` `*{\F00}[\i]{\TEXT}`
User command to define `\F00` to be a macro with `i` arguments (`i = 0` if missing) having the definition `\TEXT`. Produces an error if `\F00` already defined.

Normally the command is defined to be `\long` (ie it may take multiple paragraphs in its argument). In the star-form, the command is not defined as `\long` and a blank line in any argument to the command would generate an error.

`\renewcommand` `*{\F00}[\i]{\TEXT}`

Same as `\newcommand`, except it checks if `\F00` already defined.

`\newenvironment` $\star{\langle FOO\rangle}[\langle i\rangle]{\langle DEF1\rangle}{\langle DEF2\rangle}$
equivalent to:
`\newcommand{\F00}[i]{DEF1} \def{\endF00}{DEF2}`
(or the appropriate star forms).

`\renewenvironment` Obvious companion to `\newenvironment`.

`\@cons` : See description of `\output` routine.

`\@car` `\@car T1 T2 ... Tn\@nil == T1` (unexpanded)

`\@cdr` `\@cdr T1 T2 ... Tn\@nil == T2 ... Tn` (unexpanded)

`\typeout` $\{\langle message\rangle\}$
Produces a warning message on the terminal.

`\typein` $\{\langle message\rangle\}$
Types message, asks the user to type in a command, then executes it

`\typein` $[\langle\backslash CS\rangle]{\langle MSG\rangle}$
Same as above, except defines `\CS` to be the input instead of executing it.

`\typein`

```

31 \def\typein{%
32   \let\@typein\relax
33   \@testopt\@xtypein\@typein}

34 \ifx\directlua\@undefined

35 \def\@xtypein[#1]#2{%
36   \typeout{#2}%
37   \advance\endlinechar\@M
38   \read\@inputcheck to#1%
39   \advance\endlinechar-\@M
40   \@typein}%

41 \else

42 \def\@xtypein[#1]#2{%
43   \typeout{#2}%
44   \begingroup \endlinechar\m@ne
45   \read\@inputcheck to#1%
46   \expandafter\endgroup
47   \expandafter\def\expandafter#1\expandafter{#1}%
48   \@typein}%

49 \fi

```

(End definition for `\typein`.)

`\@namedef`

```

50 \def\@namedef#1{\expandafter\def\csname #1\endcsname}

```

(End definition for `\@namedef`.)

`\@nameuse`

```

51 \def\@nameuse#1{\csname #1\endcsname}

```

(End definition for `\@nameuse`.)

```

\@cons
52 \def\@cons#1#2{\begingroup\let\@elt\relax\xdef#1{#1\@elt #2}\endgroup}

(End definition for \@cons.)

\@car
\@cdr
53 \def\@car#1#2\@nil{#1}
54 \def\@cdr#1#2\@nil{#2}

(End definition for \@car and \@cdr.)

\@carcube \@carcube T1 ... Tn\@nil = T1 T2 T3 , n > 3
55 \</2ekernel>
56 \<latexrelease>\IncludeInRelease{2020/10/01}{\@carcube}{Make \@carcube long}%
57 \<*2ekernel | latexrelease>
58 \long\def\@carcube#1#2#3#4\@nil{#1#2#3}
59 \</2ekernel | latexrelease>
60 \<latexrelease>\EndIncludeInRelease
61 %
62 \<latexrelease>\IncludeInRelease{0000/00/00}{\@carcube}{Undo: Make \@carcube long}%
63 \<latexrelease>\def\@carcube#1#2#3#4\@nil{#1#2#3}
64 \<latexrelease>\EndIncludeInRelease
65 \<*2ekernel>

(End definition for \@carcube.)

\@onlypreamble This macro adds its argument to the list of commands stored in \@preamblecmds
\@preamblecmds to be disabled after \begin{document}. These commands are redefined to generate
\@notprerr at this point.
66 \def\@preamblecmds{}
67 \def\@onlypreamble#1{%
68   \expandafter\gdef\expandafter\@preamblecmds\expandafter{%
69     \@preamblecmds\do#1}}
70 \@onlypreamble\@onlypreamble
71 \@onlypreamble\@preamblecmds

(End definition for \@onlypreamble and \@preamblecmds.)

\@star@or@long Look ahead for a *. If present reset \l@ngrel@x so that the next definition, #1, will be
non-long.
72 \def\@star@or@long#1{%
73   \ifstar
74     {\let\l@ngrel@x\relax#1}%
75     {\let\l@ngrel@x\long#1}}

(End definition for \@star@or@long.)

\l@ngrel@x This is either \relax or \long depending on whether the *-form of a definition command
is being executed.
76 \let\l@ngrel@x\relax

(End definition for \l@ngrel@x.)

\newcommand User level \newcommand.
77 \def\newcommand{\@star@or@long\new@command}

```

```

\newcommand 78 \def\newcommand#1{%
79 \testopt{\@newcommand#1}0}

(End definition for \newcommand and \@newcommand.)

```

```

\@newcommand Handling arguments for \newcommand.
\@argdef 80 \def\@newcommand#1[#2]{%
\@xargdef 81 \kernel@ifnextchar [{\@xargdef#1[#2]}%
82 {\@argdef#1[#2]}}

```

Define #1 if it is definable.

Both here and in \@xargdef the replacement text is absorbed as an argument because if we are not allowed to make the definition we have to get rid of it completely.

```

83 \long\def\@argdef#1[#2]#3{%
84 \ifdefinable #1{\@yargdef#1\@ne{#2}{#3}}

```

Handle the second optional argument.

```

85 \long\def\@xargdef#1[#2] [#3]#4{%
86 \ifdefinable#1{%

```

Define the actual command to be:

```

\def\foo{\@protected@testopt\foo\\foo{default}}

```

where \\foo is a csname generated from applying \csname and \string to \foo, ie the actual name contains a backslash and therefore can't clash easily with existing command names. "Default" is the contents of the second optional argument of (re)newcommand.

```

87 \expandafter\def\expandafter#1\expandafter{%
88 \expandafter
89 \@protected@testopt
90 \expandafter
91 #1%
92 \csname\string#1\endcsname
93 {#3}}%

```

Now we define the internal macro ie \\foo which is supposed to pick up all arguments (optional and mandatory).

```

94 \expandafter\@yargdef
95 \csname\string#1\endcsname
96 \tw@
97 {#2}%
98 {#4}}

```

(End definition for \@newcommand, \@argdef, and \@xargdef.)

\@testopt This macro encapsulates the most common call to \@ifnextchar, saving several tokens each time it is used in the definition of a command with an optional argument. #1 The code to execute in the case that there is a [need not be a single token but can be any sequence of commands that 'expects' to be followed by [. If this command were only used in \newcommand definitions then #1 would be a single token and the braces could be omitted from {#1} in the definition below, saving a bit of memory.

```

99 \long\def\@testopt#1#2{%
100 \kernel@ifnextchar[{#1}{#1[{#2}]]}

```

(End definition for \@testopt.)

`\@protected@testopt` Robust version of `\@testopt`. The extra argument (`#1`) must be a single token. If protection is needed the call expands to `\protect` applied to this token, and the 2nd and 3rd arguments are discarded (by `\@x@protect`). Otherwise `\@testopt` is called on the 2nd and 3rd arguments.

This method of making commands robust avoids the need for using up two csnames per command, the price is the extra expansion time for the `\ifx` test.

```

101 \def\@protected@testopt#1{%
102   \ifx\protect\@typeset@protect
103     \expandafter\@testopt
104   \else
105     \@x@protect#1%
106   \fi}

```

(End definition for `\@protected@testopt`.)

`\@yargdef` These generate a primitive argument specification, from a L^AT_EX [*digit*] form; in fact `\@yargd@f` *digit* can be anything such that `\number <digit>` is single digit.

Reorganised slightly so that `\renewcommand{\reserved@a}[1]{foo}` works. I am not sure this is worth it, as a following `\newcommand` would over-write the definition of `\reserved@a`.

Recall that L^AT_EX 2.09 goes into an infinite loop with `\renewcommand[1]{\@tempa}{foo}` (DPC 6 October 93).

Reorganised again (DPC 1999). Rather than make a loop to construct the argument spec by counting, just extract the required argument spec by using a delimited argument (delimited by the digit). This is faster and uses less tokens. The coding is slightly odd to preserve the old interface (using `#2 = \tw@` as the flag to surround the first argument with []). But the new method did not allow for the number of arguments `#3` not being given as an explicit digit; hence (further expansion of this argument and use of) `\number` was added later in 1999.

It is not clear why these are still `\long`.

```

107 \long \def \@yargdef #1#2#3{%
108   \ifx#2\tw@
109     \def\reserved@b##11{####1}}%
110   \else
111     \let\reserved@b\@gobble
112   \fi
113   \expandafter
114     \@yargd@f \expandafter{\number #3}#1%
115 }

116 \long \def \@yargd@f#1#2{%
117   \def \reserved@a ##1#1##2##{%
118     \expandafter\def\expandafter#2\reserved@b ##1#1%
119   }%
120   \l@ngrel@x \reserved@a 0##1##2##3##4##5##6##7##8##9###1%
121 }

```

(End definition for `\@yargdef` and `\@yargd@f`.)

`\@reargdef`

```

122 \long\def\@reargdef#1[#2]{%
123   \@yargdef#1\@ne{#2}}

```

(End definition for \@reargdef.)

\renewcommand Check the command name is already used. If not give an error message. Then temporarily disable \@ifdefinable then call \newcommand. (Previous version \let#1=\relax but this does not work too well if #1 is \@tempa-e.)

```
124 \def\renewcommand{\@star@or@long\renew@command}
```

```
\renew@command 125 \def\renew@command#1{%
126   \begingroup \escapechar\m@ne\xdef\@gtempa{\string#1}}\endgroup
127   \expandafter\ifundefined\@gtempa
128     {\@latex@error{Command \string#1 undefined}\@ehc}%
129     \relax
130   \let\@ifdefinable\@rc@ifdefinable
131   \new@command#1}
```

(End definition for \renewcommand and \renew@command.)

\@ifdefinable Test if user is allowed to define a command.

```
\@ifdefinable 132 \long\def\@ifdefinable #1#2{%
\@rc@ifdefinable 133   \edef\reserved@a{\expandafter\@gobble\string #1}%
134   \@ifundefined\reserved@a
135     {\edef\reserved@b{\expandafter\@carcube \reserved@a xxx\@nil}%
136     \ifx \reserved@b\@qend \@notdefinable\else
137     \ifx \reserved@a\@qrelax \@notdefinable\else
138     #2%
139     \fi
140     \fi}%
141   \@notdefinable}
```

Saved definition of \@ifdefinable.

```
142 \let\@@ifdefinable\@ifdefinable
```

Version of \@ifdefinable for use with \renewcommand. Does not do the check this time, but restores the normal definition.

```
143 \long\def\@rc@ifdefinable#1#2{%
144   \let\@ifdefinable\@@ifdefinable
145   #2}
```

(End definition for \@ifdefinable, \@@ifdefinable, and \@rc@ifdefinable.)

\newenvironment Define a new user environment. #1 is the environment name. #2# Grabs all the tokens up to the first {. These will be any optional arguments. They are not parsed at this point, but are just passed to \@newenv which will eventually call \newcommand. Any optional arguments will then be parsed by \newcommand as it defines the command that executes the 'begin code' of the environment.

This #2# trick removed with version 1.2i as it fails if a { occurs in the optional argument. Now use \@ifnextchar directly.

```
146 \def\newenvironment{\@star@or@long\new@environment}
```

```
\new@environment 147 \def\new@environment#1{%
148   \@testopt{\@newenva#1}0}
```

```

149 \def\@newenva#1[#2]{%
\@newenva 150 \kernel@ifnextchar [{\@newenvb#1[#2]}{\@newenv{#1}{[#2]}}}]

151 \def\@newenvb#1[#2][#3]{\@newenv{#1}{[#2][#3]}}
\@newenvb (End definition for \newenvironment and others.)

\renewenvironment Redefine an environment. For \renewenvironment disable \@ifdefinable and then call
\newenvironment. It is OK to \let the argument to \relax here as there should not
be a @temp... environment.
152 \def\renewenvironment{\@star@or@long\renew@environment}

\renew@environment 153 \def\renew@environment#1{%
154 \ifundefined{#1}%
155 {\@latex@error{Environment #1 undefined}\@ehc
156 }\relax
157 \expandafter\let\csname#1\endcsname\relax
158 \expandafter\let\csname end#1\endcsname\relax
159 \new@environment{#1}}
(End definition for \renewenvironment and \renew@environment.)

\@newenv The internal version of \newenvironment.
Call \newcommand to define the <begin-code> for the environment. \def is used for
the <end-code> as it does not take arguments. (but may contain \pars)
Make sure that an attempt to define a ‘graf’ or ‘group’ environment fails.
160 \long\def\@newenv#1#2#3#4{%
161 \ifundefined{#1}%
162 {\expandafter\let\csname#1\expandafter\endcsname
163 \csname end#1\endcsname}%
164 \relax
165 \expandafter\new@command
166 \csname #1\endcsname#2{#3}%
167 \l@ngrel@x\expandafter\def\csname end#1\endcsname{#4}}
(End definition for \@newenv.)

\newif And here’s a different sort of allocation: For example, \newif\iffoo creates \footrue,
\foofalse to go with \iffoo.
168 \def\newif#1{%
169 \count@\escapechar \escapechar\m@ne
170 \let#1\iffalse
171 \@if#1\iftrue
172 \@if#1\iffalse
173 \escapechar\count@}

\@if 174 \def\@if#1#2{%
175 \expandafter\def\csname\expandafter@gobbletwo\string#1%
176 \expandafter@gobbletwo\string#2\endcsname
177 {\let#1#2}}

```

(End definition for \newif and \if.)

\providecommand \providecommand takes the same arguments as \newcommand, but discards them if #1 is already defined, Otherwise it just acts like \newcommand. This implementation currently leaves any discarded definition in \reserved@a (and possibly \reserved@a) this wastes a bit of space, but it will be reclaimed as soon as these scratch macros are redefined.

```
178 \def\providecommand{\@star@or@long\provide@command}
```

```
\provide@command 179 \def\provide@command#1{%
180   \begingroup
181     \escapechar\m@ne\xdef\@gtempa{\string#1}%
182   \endgroup
183   \expandafter\@ifundefined\@gtempa
184     {\def\reserved@a{\new@command#1}}%
185     {\def\reserved@a{\renew@command\reserved@a}}%
186   \reserved@a}%

```

(End definition for \providecommand and \provide@command.)

\CheckCommand \CheckCommand takes the same arguments as \newcommand. If the command already exists, with the same definition, then nothing happens, otherwise a warning is issued. Useful for checking the current state before a macro package starts redefining things. Currently two macros are considered to have the same definition if they are the same except for different default arguments. That is, if the old definition was: \newcommand\xxx[2][a]{(#1)(#2)} then \CheckCommand\xxx[2][b]{(#1)(#2)} would *not* generate a warning, but, for instance \CheckCommand\xxx[2]{(#1)(#2)} would.

```
187 \def\CheckCommand{\@star@or@long\check@command}
```

\CheckCommand is only available in the preamble part of the document.

```
188 \@onlypreamble\CheckCommand
```

```
\check@command 189 \def\check@command#1#2#{\@check@c#1{#2}}
190 \@onlypreamble\check@command
```

(End definition for \CheckCommand and \check@command.)

\@check@c \CheckCommand itself just grabs all the arguments we need, without actually looking for [optional argument forms. Now define \reserved@a. If \reserved@a is then defined, compare it with the “\#1’ otherwise compare \reserved@a with #1.

```
191 \long\def\@check@c#1#2#3{%
192   \expandafter\let\csname\string\reserved@a\endcsname\relax
193   \renew@command\reserved@a#2{#3}%
194   \@ifundefined{\string\reserved@a}%
195     {\@check@eq#1\reserved@a}%
196     {\expandafter\@check@eq
197       \csname\string#1\expandafter\endcsname
198       \csname\string\reserved@a\endcsname}}
199 \@onlypreamble\@check@c

```

(End definition for \@check@c.)


```

\@check@eq Complain if #1 and #2 are not \ifx equal.
200 \def\@check@eq#1#2{%
201   \ifx#1#2\else
202     \@latex@warning@no@line
203       {Command \noexpand#1 has
204         changed.\MessageBreak
205         Check if current package is valid}%
206   \fi}
207 \onlypreamble\@check@eq

(End definition for \@check@eq.)

\@gobble The \@gobble macro is used to get rid of its argument.
\@gobbletwo 208 \long\def \@gobble #1{}
\@gobblethree 209 \long\def \@gobbletwo #1#2{}
\@gobblefour 210 \long\def \@gobblethree #1#2#3{}
211 \long\def \@gobblefour #1#2#3#4{}

(End definition for \@gobble and others.)

\@firstofone Some argument-grabbers.
\@firstoftwo 212 \long\def \@firstofone#1{#1}
\@secondoftwo 213 \long\def \@firstoftwo#1#2{#1}
214 \long\def \@secondoftwo#1#2{#2}

\@iden is another name for \@firstofone for compatibility reasons.
215 \let\@iden\@firstofone

\@iden (End definition for \@firstofone and others.)

\@thirdofthree Another grabber now used in the encoding specific section.
216 \long\def \@thirdofthree#1#2#3{#3}

(End definition for \@thirdofthree.)

\@expandtwoargs A macro to totally expand two arguments to another macro
217 \def\@expandtwoargs#1#2#3{%
218   \edef\reserved@a{\noexpand#1{#2}{#3}}\reserved@a}

(End definition for \@expandtwoargs.)

\@backslashchar A category code 12 backslash.
219 \edef\@backslashchar{\expandafter\@gobble\string\\}

(End definition for \@backslashchar.)

```

1.4 Robust commands and protect

Fragile and robust commands are one of the thornier issues in L^AT_EX's commands. Whilst typesetting documents, L^AT_EX makes use of many of T_EX's features, such as arithmetic, defining macros, and setting variables. However, there are (at least) three different occasions when these commands are not safe. These are called 'moving arguments' by L^AT_EX, and consist of:

- writing information to a file, such as indexes or tables of contents.
- writing information to the screen.
- inside an `\edef`, `\message`, `\mark`, or other command which evaluates its argument fully.

The method L^AT_EX uses for making fragile commands robust is to precede them with `\protect`. This can have one of four possible values:

- `\relax`, for normal typesetting. So `\protect\foo` will execute `\foo`.
- `\string`, for writing to the screen. So `\protect\foo` will write `\foo`.
- `\noexpand`, for writing to a file. So `\protect\foo` will write `\foo` followed by a space.
- `\@unexpandable@protect`, for writing a moving argument to a file. So `\protect\foo` will write `\protect\foo` followed by a space. This value is also used inside `\edefs`, `\marks` and other commands which evaluate their arguments fully. More precisely, whenever the content of an `\edef` or `\xdef` etc. can contain arbitrary user input not under the direct control of the programmer, one should use `\protected@edef` instead of `\edef`, etc., so that `\protect` has a suitable definition and the user input will not break if it contains fragile commands.

`\@unexpandable@protect`

```
220 \def\@unexpandable@protect{\noexpand\protect\noexpand}
```

(End definition for \@unexpandable@protect.)

`\DeclareRobustCommand`
`\declare@robustcommand`

This is a package-writers command, which has the same syntax as `\newcommand`, but which declares a protected command. It does this by having

```
\DeclareRobustCommand\foo
define \foo to be \protect\foo<space>,
and then use \newcommand\foo<space>.
```

Since the internal command is `\foo<space>`, when it is written to an auxiliary file, it will appear as `\foo`.

We have to be a bit cleverer if we're defining a short command, such as `_`, in order to make sure that the auxiliary file does not include a space after the command, since `_ a` and `_a` aren't the same. In this case we define `_` to be:

```
\x@protect\_ \protect\_<space>
```

which expands to:

```

\ifx\protect\@typeset@protect\else
  \@x@protect@\_
\fi
\protect\_<space>

```

Then if `\protect` is `\@typeset@protect` (normally `\relax`) then we just perform `_<space>`, and otherwise `\@x@protect@` gobbles everything up and expands to `\protect_<space>`.

Note: setting `\protect` to any value other than `\relax` whilst in ‘typesetting’ mode will cause commands to go into an infinite loop! In particular, setting `\protect` to `\@empty` will cause `_` to loop forever. It will also break lots of other things, such as protected `\ifmmodes` inside `\haligns`. If you really have to do such a thing, then please set `\@typeset@protect` to be `\@empty` as well. (This is what the code for `\patterns` does, for example.)

More fun with `\expandafter` and `\csname`.

```

221 \def\DeclareRobustCommand{\@star@or@long\declare@robustcommand}
222 \def\declare@robustcommand#1{%
223   \ifx#1\@undefined\else\ifx#1\relax\else
224     \@latex@info{Redefining \string#1}%
225   \fi\fi
226   \edef\reserved@a{\string#1}%
227   \def\reserved@b{#1}%
228   \edef\reserved@b{\expandafter\strip@prefix\meaning\reserved@b}%
229   \edef#1{%
230     \ifx\reserved@a\reserved@b
231       \noexpand\x@protect
232       \noexpand#1%
233     \fi
234     \noexpand\protect
235     \expandafter\noexpand\csname
236       \expandafter\@gobble\string#1 \endcsname
237   }%
238   \let\@ifdefinable\@rc@ifdefinable
239   \expandafter\new@command\csname
240     \expandafter\@gobble\string#1 \endcsname
241 }

```

(End definition for `\DeclareRobustCommand` and `\declare@robustcommand`.)

```

\@x@protect
\@x@protect
242 \def\x@protect#1{%
243   \ifx\protect\@typeset@protect\else
244     \@x@protect#1%
245   \fi
246 }
247 \def\@x@protect#1\fi#2#3{%
248   \fi\protect#1%
249 }

```

(End definition for `\@x@protect` and `\x@protect`.)

`\@typeset@protect` We set `\@typeset@protect` to `\relax` rather than `\@empty` to make sure that the protection mechanism stops the look-ahead and expansion performed at the start of `\halign` cells.

```

250 \let\@typeset@protect\relax

```

(End definition for \@typeset@protect.)

\set@display@protect These macros set \protect appropriately for typesetting or displaying.

```
\set@typeset@protect 251 \def\set@display@protect{\let\protect\string}
252 \def\set@typeset@protect{\let\protect\@typeset@protect}
```

(End definition for \set@display@protect and \set@typeset@protect.)

\protected@edef The commands \protected@edef and \protected@xdef perform ‘safe’ \edefs and

\protected@xdef \xdefs, saving and restoring \protect appropriately. For cases where restoring \protect

\unrestored@protected@xdef doesn’t matter, there’s an ‘unsafe’ \unrestored@protected@xdef, useful if you know what you’re doing!

```
\restore@protect 253 \def\protected@edef{%
254   \let\@@protect\protect
255   \let\protect\@unexpandable@protect
256   \afterassignment\restore@protect
257   \edef
258 }
259 \def\protected@xdef{%
260   \let\@@protect\protect
261   \let\protect\@unexpandable@protect
262   \afterassignment\restore@protect
263   \xdef
264 }
265 \def\unrestored@protected@xdef{%
266   \let\protect\@unexpandable@protect
267   \xdef
268 }
269 \def\restore@protect{\let\protect\@@protect}
```

(End definition for \protected@edef and others.)

\protect The normal meaning of \protect

```
270 \set@typeset@protect
```

(End definition for \protect.)

\MakeRobust This macro makes an existing fragile macro robust, but only if it hasn’t been robust in the past, i.e., it checks for the existence of the macro \<name>_␣ and if that exists it assumes that \<name> is already robust. In that case either undefine the inner macro first or use \DeclareRobustCommand to define it in a robust way directly. We could probably test the top-level definition to have the right kind of structure, but this is somewhat problematical as we then have to distinguish between \long macros and others and also take into account that sometimes the top-level is deliberately done manually (like with \begin).

The macro firstly checks if the control sequence in question exists at all.

```
271 \</2ekernel>
272 \<latexrelease>\IncludeInRelease{2020/10/01}{\MakeRobust}{\MakeRobust}%
273 \<*2ekernel | latexrelease>
274 \def\MakeRobust#1{%
275   \count@=\escapechar
276   \escapechar=‘\’
277   \@ifundefined{expandafter\gobble\string#1}{%
278     \@latexerror{Command ‘\string#1’ is undefined!%
```

```

279     \MessageBreak There is nothing here to make robust}}%
280     \@eha
281 }%

```

Then we check if the macro is already robust. We do this by testing if the internal name for a robust macro is defined, namely `\foo_`. If it is already defined do nothing, otherwise set `\foo_` equal to `\foo` and redefine `\foo` so that it acts like a macro defined with `\DeclareRobustCommand`. We use `\@kernel@rename@newcommand` to copy `\foo` over to `\foo_`, including a possible default optional argument.

```

282 {%
283   \ifundefined{\expandafter\@gobble\string#1\space}%
284   {%
285     \expandafter\@kernel@rename@newcommand
286     \csname\expandafter\@gobble\string#1\space\endcsname
287     #1%
288     \edef\reserved@a{\string#1}%
289     \def\reserved@b{#1}%
290     \edef\reserved@b{\expandafter\strip@prefix\meaning\reserved@b}%
291     \xdef#1{%
292       \ifx\reserved@a\reserved@b
293         \noexpand\x@protect\noexpand#1%
294       \fi
295       \noexpand\protect\expandafter\noexpand
296       \csname\expandafter\@gobble\string#1\space\endcsname}%
297   }%
298   {\@latex@info{Command ‘\string#1’ is already robust}}}%
299 }%
300 \escapechar=\count@
301 }%

```

`\@kernel@rename@newcommand`

This macro renames a command, possibly with an optional argument (defined with `\newcommand`) from `#2` to `#1`, by renaming the internal macro `\\#2` to `\\#1` and defining `\#1` appropriately, then undefining `\#2` and `\\#2`. The `\afterassignment` trick is to make both definitions in `\@copy@newcommand` global (which are local by default).

In case the macro was defined with `\newcommand` and an optional argument, to replicate exactly the behaviour of `\DeclareRobustCommand` we have to move also the internal `\\foo` to `\\foo_`. In that case, `#1` will be a parameterless macro (`\robust@command@chk@safe` checks that), and `\@if@newcommand` will return true (both defined below in this file). If so, we can use `\@copy@newcommand` rather than plain `\let` to copy the command over. `\@kernel@rename@newcommand` does this test and carries out the renaming.

```

302 \def\@kernel@rename@newcommand#1#2{%
303   \robust@command@chk@safe#2%
304   {\@if@newcommand#2%
305     {\afterassignment\global
306       \global\@copy@newcommand#1#2%
307       \global\let#2\undefined
308       \global\expandafter\let\csname\string#2\endcsname\@undefined}%
309     {\global\let#1=#2}}%
310   {\global\let#1=#2}}
311 \</2ekernel | latexrelease>
312 \<latexrelease>\EndIncludeInRelease

```

```

313 %
314 <latexrelease>\IncludeInRelease{2019/10/01}{\MakeRobust}{\MakeRobust}%
315 <latexrelease>\def\MakeRobust#1{%
316 <latexrelease>  \ifundefined{\expandafter\@gobble\string#1}{%
317 <latexrelease>    \@latex@error{The control sequence '\string#1' is undefined!%
318 <latexrelease>    \MessageBreak There is nothing here to make robust}%
319 <latexrelease>    \@eha
320 <latexrelease>  }%
321 <latexrelease>  {%
322 <latexrelease>    \ifundefined{\expandafter\@gobble\string#1\space}%
323 <latexrelease>    {%
324 <latexrelease>      \global\expandafter\let\csname
325 <latexrelease>        \expandafter\@gobble\string#1\space\endcsname=#1%
326 <latexrelease>      \edef\reserved@a{\string#1}%
327 <latexrelease>      \def\reserved@b{#1}%
328 <latexrelease>      \edef\reserved@b{\expandafter\strip@prefix\meaning\reserved@b}%
329 <latexrelease>      \xdef#1{%
330 <latexrelease>        \ifx\reserved@a\reserved@b
331 <latexrelease>          \noexpand\x@protect\noexpand#1%
332 <latexrelease>        \fi
333 <latexrelease>        \noexpand\protect\expandafter\noexpand
334 <latexrelease>        \csname\expandafter\@gobble\string#1\space\endcsname}%
335 <latexrelease>      }%
336 <latexrelease>    {\@latex@info{The control sequence '\string#1' is already robust}}%
337 <latexrelease>  }%
338 <latexrelease>}%
339 <latexrelease>\let\@kernel@rename@newcommand\@undefined
340 <latexrelease>\EndIncludeInRelease
341 %
342 <latexrelease>\IncludeInRelease{2015/01/01}{\MakeRobust}{\MakeRobust}%
343 <latexrelease>\def\MakeRobust#1{%
344 <latexrelease>  \ifundefined{\expandafter\@gobble\string#1}{%
345 <latexrelease>    \@latex@error{The control sequence '\string#1' is undefined!%
346 <latexrelease>    \MessageBreak There is nothing here to make robust}%
347 <latexrelease>    \@eha
348 <latexrelease>  }%
349 <latexrelease>  {%
350 <latexrelease>    \ifundefined{\expandafter\@gobble\string#1\space}%
351 <latexrelease>    {%
352 <latexrelease>      \expandafter\let\csname
353 <latexrelease>        \expandafter\@gobble\string#1\space\endcsname=#1%
354 <latexrelease>      \edef\reserved@a{\string#1}%
355 <latexrelease>      \def\reserved@b{#1}%
356 <latexrelease>      \edef\reserved@b{\expandafter\strip@prefix\meaning\reserved@b}%
357 <latexrelease>      \xdef#1{%
358 <latexrelease>        \ifx\reserved@a\reserved@b
359 <latexrelease>          \noexpand\x@protect\noexpand#1%
360 <latexrelease>        \fi
361 <latexrelease>        \noexpand\protect\expandafter\noexpand
362 <latexrelease>        \csname\expandafter\@gobble\string#1\space\endcsname}%
363 <latexrelease>      }%
364 <latexrelease>    {\@latex@info{The control sequence '\string#1' is already robust}}%
365 <latexrelease>  }%
366 <latexrelease>}%

```

```

367 <latexrelease>\let\@kernel@rename@newcommand\@undefined
368 <latexrelease>\EndIncludeInRelease
369 %
370 <latexrelease>\IncludeInRelease{0000/00/00}{\MakeRobust}{\MakeRobust}%
371 <latexrelease>\let\MakeRobust\@undefined
372 <latexrelease>\let\@kernel@rename@newcommand\@undefined
373 <latexrelease>\EndIncludeInRelease
374 <*2ekernel>

```

(End definition for \MakeRobust and \@kernel@rename@newcommand.)

\kernel@make@fragile The opposite of \MakeRobust except that it doesn't do many checks as it is internal to the kernel. Why does one want such a thing? Only for compatibility reasons if latexrelease requests a rollback of the kernel. For this reason we pretend that this command existed in all earlier versions of L^AT_EX i.e., we are not rolling it back since we need it precisely then. But we have to get it into the latexrelease file so that a roll forward is possible too.

```

375 </2ekernel>
376 <*2ekernel | latexrelease>
377 <latexrelease>\IncludeInRelease{2020/10/01}%
378 <latexrelease>          {\kernel@make@fragile}{Undo robustness}%
379 \def\kernel@make@fragile#1{%
380   \@ifundefined{\expandafter\@gobble\string#1\space}%

```

If not robust do nothing.

```

381   {}%

```

Otherwise copy \foo_␣ back to \foo. Then use \@kernel@rename@newcommand to check and copy \foo_␣ back to \foo in case the command has an optional argument. If so, also undefine \foo_␣, and at the end undefine \foo_␣.

```

382   {%
383     \global\expandafter\let\expandafter #1\csname
384       \expandafter\@gobble\string#1\space\endcsname
385     \expandafter\@kernel@rename@newcommand
386       \csname\expandafter\@gobble\string#1\expandafter\endcsname
387       \csname\expandafter\@gobble\string#1\space\endcsname
388     \global\expandafter\let\csname
389       \expandafter\@gobble\string#1\space\endcsname\@undefined
390   }%
391 }
392 <latexrelease>\EndIncludeInRelease
393 %
394 <latexrelease>\IncludeInRelease{0000/00/00}%
395 <latexrelease>          {\kernel@make@fragile}{Undo robustness}%
396 <latexrelease>\def\kernel@make@fragile#1{%
397 <latexrelease>  \@ifundefined{\expandafter\@gobble\string#1\space}%
398 <latexrelease>    {}%
399 <latexrelease>    {%
400 <latexrelease>      \global\expandafter\let\expandafter #1\csname
401 <latexrelease>        \expandafter\@gobble\string#1\space\endcsname
402 <latexrelease>      \global\expandafter\let\csname
403 <latexrelease>        \expandafter\@gobble\string#1\space\endcsname\@undefined
404 <latexrelease>    }%
405 <latexrelease>}

```

```

406 \latexrelease\EndIncludeInRelease
407 \</2ekernel\latexrelease>
408 \*2ekernel)

```

(End definition for \kernel@make@fragile.)

1.5 Acting on robust commands

```

409 \</2ekernel>
410 \latexrelease\IncludeInRelease{2020-10-01}{\robust@command@act}
411 \latexrelease {Add \robust@command@act}%
412 \*2ekernel\latexrelease>

```

With most document level commands being robust now there is more of a requirement to have a standard way of aliasing (or copying) a command to a new name, for example to save an original definition before changing a command. `\DeclareCommandCopy` is analogous to \TeX 's `\let`, except that it copes with the different types of robust commands defined by \LaTeX 's mechanisms.

A couple of “types of robustness” are defined by the \LaTeX 2_ϵ kernel, namely robust commands defined with `\DeclareRobustCommand` and commands with optional arguments defined with `\newcommand`. However there are other types of robust commands that are frequently used, which are not defined in the \LaTeX 2_ϵ kernel, like commands defined with `xparse`'s `\NewDocumentCommand` and `etoolbox`'s `\newrobustcmd`.

In this section we will define a generic extensible machinery to act on robust commands. This code will then be used to test if a command is robust, considered the different types of robustness, and then either copy that definition, if `\DeclareCommandCopy` (or similar) is used, or show the definition of the command, if `\ShowCommand` is used.

`\robust@command@act` The looping machinery is generic and knows nothing about what is to be done for each case. The syntax of the main macro `\robust@command@act` is:

```

\robust@command@act<action-list><robust-cmd>
<fallback-action><act-arg>

```

`<action-list>` is a token list of the form:

```

{\if-type-1} <act-type-1>
{\if-type-2} <act-type-2>
...

```

`\robust@command@act` will iterate over the `<action-list>`, evaluating each `<if-type-n>` `<robust-cmd>` `{\true}{\false}`. If the `<if-type-n>` conditional returns `<true>`, then `<act-type-n>``<act-arg>` is executed, and the loop ends. If the conditional returns `<false>`, then `<if-type-n + 1>` is executed in the same way, until either one of the conditionals return `<true>`, or the end of the `<action-list>` is reached. If the end is reached, then `<fallback-action>``<act-arg>` is executed before `\robust@command@act` exits.

`\robust@command@act` will start by using `\robust@command@act@chk@args` to check if the `<robust-cmd>` (#2) is a parameterless (possibly `\protected`) macro. If it is not, the command is not a robust command: these always start with a parameterless user-level macro; in that case, `\robust@command@act@end` is used to short-circuit the process and do the `<fallback-action>` (#3). This first test is necessary because later on we need to be able to expand the `<robust-cmd>` without the risk of it Breaking Badly, and as a bonus, this speeds up the process in case we used `\NewCommandCopy` in a “normal” macro.

```

413 \long\def\robust@command@act#1#2#3#4{%

```



```

414 \robust@command@chk@safe#2%
415   {\expandafter\robust@command@act@loop
416     \expandafter#2%
417     #1{\@nnil\@nnil}%
418     \robust@command@act@end}%
419   {\robust@command@act@end}%
420   {#3}{#4}}%

```

If `\robust@command@act@chk@args` branched to false, then `\robust@command@act@loop` will loop over the list of items in the $\langle action-list \rangle$ ($\#1$), and process each item as described earlier. If the $\langle if-type-n \rangle$ command expands to $\langle true \rangle$ then `\robust@command@act@do` is used to execute $\langle act-type-n \rangle$ on the $\langle act-arg \rangle$, otherwise the loop resumes with the next item.

```

421 \long\def\robust@command@act@loop#1#2{\robust@command@act@loop@aux#1#2}
422 \long\def\robust@command@act@loop@aux#1#2#3{%
423   \ifx\@nnil#2%
424   \else
425     #2{#1}%
426     {\robust@command@act@do{#3}}%
427     {\expandafter\robust@command@act@loop\expandafter#1}%
428   \fi}
429 \long\def\robust@command@act@do#1%
430   \fi#2%
431   \robust@command@act@end#3#4{%
432   \fi
433   #1#4}

```

If the end is reached and no action was taken, then do $\langle fallback-action \rangle \langle act-arg \rangle$.

```

\robust@command@act@end 434 \long\def\robust@command@act@end#1#2{#1#2}

```

```

\robust@command@chk@safe 435 \long\def\robust@command@chk@safe#1{%
\robust@command@act@chk@args 436   \begingroup
437   \escapechar='\\
438   \expandafter\endgroup\expandafter
439   \robust@command@act@chk@args\meaning#1:->\@nil}
440 \def\robust@command@act@chk@args#1:->#2\@nil{%
441   \@expl@str@if@eq@nnTF{#1}{macro}%
442   {\@firstoftwo}%
443   {\@expl@str@if@eq@nnTF{#1}{\protected macro}%
444   {\@firstoftwo}%
445   {\@secondoftwo}}}

446 </2ekernel | latexrelease>
447 <latexrelease>\EndIncludeInRelease
448 <latexrelease>\IncludeInRelease{0000-00-00}{\robust@command@act}
449 <latexrelease> {Add \robust@command@act}%
450 <latexrelease>\let\robust@command@act\@undefined
451 <latexrelease>\let\robust@command@act@loop\@undefined
452 <latexrelease>\let\robust@command@act@loop@aux\@undefined
453 <latexrelease>\let\robust@command@act@do\@undefined
454 <latexrelease>\let\robust@command@act@end\@undefined
455 <latexrelease>\let\robust@command@chk@safe\@undefined
456 <latexrelease>\let\robust@command@act@chk@args\@undefined

```

```

457 <latexrelease>\EndIncludeInRelease
458 <*2ekernel>
(End definition for \robust@command@act and others.)

```

1.5.1 Copying robust commands

```

459 </2ekernel>
460 <latexrelease>\IncludeInRelease{2020-10-01}{\DeclareCommandCopy}
461 <latexrelease> {Add \NewCommandCopy, \RenewCommandCopy, and \DeclareCommandCopy}%
462 <*2ekernel | latexrelease>

```

`\NewCommandCopy` `\NewCommandCopy` starts by checking if `#1` is already defined, and raises an error if so, otherwise the definition is carried out. `\RenewCommandCopy` does (almost) the opposite. `\DeclareCommandCopy` If the command is *not* defined, then an error is raised. But the definition is carried out anyway, so the behaviour is consistent with `\renewcommand`.

A `\ProvideCommandCopy` isn't defined because it's not reasonably useful. `\provide...` commands mean "define this if there's no other definition", but copying a command (usually) implies that the command being copied is defined, so `\ProvideCommandCopy` doesn't make a lot of sense. But more importantly, the most common use case of copying a command is to redefine it later, while preserving the old definition, as in:

```

\ProvideCommandCopy \A \B
\renewcommand \B { ... \A ... }

```

then, if `\A` is already defined the first line is skipped, and in this case `\B` won't work as expected.

The three versions call the internal `\declare@commandcopy` with the proper action. `\@firstofone` will carry out the copy. The only case when the copy is not made is the `<false>` case for `\NewCommandCopy`, in which the command already exists and the definition is aborted.

```

463 \def\NewCommandCopy{%
464   \declare@commandcopy
465   {\@firstofone}%
466   {\@firstoftwo\@notdefinable}}
467 \def\RenewCommandCopy{%
468   \declare@commandcopy
469   {\@latexerror{Command \@backslashchar\reserved@a\space undefined}\@ehc
470   \@firstofone}%
471   {\@firstofone}}
472 \def\DeclareCommandCopy{%
473   \declare@commandcopy
474   {\@firstofone}%
475   {\@firstofone}}

```

Start by checking if the command is already defined. The proper action is taken by each specific command above. If all's good, then `\robust@command@act` is called with the proper arguments as described earlier, with `\@declarecommandcopylisthook` as the `<action-list>` and `\declare@commandcopy@let` as the `<fallback-action>`.

```

\declare@commandcopy
476 \long\def\declare@commandcopy#1#2#3#4{%
477   \edef\reserved@a{\@expl@cs@to@str@{#3}}%
478   \@ifundefined\reserved@a{#1}{#2}%
479   {\robust@command@act
480    \@declarecommandcopylisthook#4%
481    \declare@commandcopy@let{#3#4}}}

```

The initial definition of `\@declarecommandcopylisthook` contains the tests for the two types of robust command in the kernel.

```

482 \def\@declarecommandcopylisthook{%
\@declarecommandcopylisthook 483   {\@if@DeclareRobustCommand \@copy@DeclareRobustCommand}%
484   {\@if@newcommand \@copy@newcommand}}

```

The initial definition of `\@declarecommandcopylisthook` contains the tests for the two types of robust command in the kernel.

```

485 \long\def\declare@commandcopy@let#1#2{\let#1=#2\relax}
\declare@commandcopy@let

```

Now the rollback code.

```

486 </2ekernel | latexrelease>
487 <latexrelease>\EndIncludeInRelease
488 <latexrelease>\IncludeInRelease{0000-00-00}{\DeclareCommandCopy}
489 <latexrelease> {Undefine \NewCommandCopy, \RenewCommandCopy, and \DeclareCommandCopy}%
490 <latexrelease>\let\NewCommandCopy\@undefined
491 <latexrelease>\let\RenewCommandCopy\@undefined
492 <latexrelease>\let\DeclareCommandCopy\@undefined
493 <latexrelease>\let\declare@commandcopy\@undefined
494 <latexrelease>\let\@declarecommandcopylisthook\@undefined
495 <latexrelease>\let\declare@commandcopy@let\@undefined
496 <latexrelease>\EndIncludeInRelease
497 <*2ekernel>

```

(End definition for \NewCommandCopy and others.)

1.5.2 Showing robust commands

`\ShowCommand` Most of the machinery defined for `\NewCommandCopy` can be used to show the definition of a robust command, in a similar fashion to `\texdef`. The difference is that after the command's is detected to has a given type of robustness, rather than making a copy, we use a separate routine to show its definition.

With all the machinery in place, `\ShowCommand` itself is quite simple: use `\robust@command@act` to iterate through the `\@showcommandlisthook` list, and if nothing is found, fallback to `\show`.

```

498 </2ekernel>
499 <latexrelease>\IncludeInRelease{2020-10-01}{\ShowCommand}%
500 <latexrelease> {Add \ShowCommand}%
501 <*2ekernel | latexrelease>
502 \long\def\ShowCommand#1{%
503   \robust@command@act
504   \@showcommandlisthook#1%
505   \show#1}

```

`\@showcommandlisthook` The initial definition of `\@showcommandlisthook` contains the same tests as used for copying, but `\@show@...` commands instead of `\@copy@...`. Same as before, it is initialized to cope with `\DeclareRobustCommand` and `\newcommand` with optional arguments.

```

506 \def\@showcommandlisthook{%
507   {\@if@DeclareRobustCommand \@show@DeclareRobustCommand}%
508   {\@if@newcommand \@show@newcommand}}

```

Now the rollback code.

```

509 </2ekernel | latexrelease>
510 <latexrelease>\EndIncludeInRelease
511 <latexrelease>\IncludeInRelease{0000-00-00}{\ShowCommand}
512 <latexrelease> {Undefine \ShowCommand}%
513 <latexrelease>\let\ShowCommand\@undefined
514 <latexrelease>\let\@showcommandlisthook\@undefined
515 <latexrelease>\EndIncludeInRelease
516 <*2ekernel>

(End definition for \ShowCommand and \@showcommandlisthook.)

517 </2ekernel>
518 <latexrelease>\IncludeInRelease{2020-10-01}{\@if@DeclareRobustCommand}
519 <latexrelease> {Add \@if@DeclareRobustCommand, \@if@newcommand,
520 <latexrelease> \@copy@DeclareRobustCommand, \@copy@newcommand,
521 <latexrelease> \@show@DeclareRobustCommand, \@show@newcommand}%
522 <*2ekernel | latexrelease>

```

1.5.3 Commands defined with \DeclareRobustCommand

`\@if@DeclareRobustCommand` Now that we provided a generic way to copy one macro to another, we need to define a way to check if a command is one of L^AT_EX 2_ε’s robust types. These tests are heavily based on Heiko’s `\LetLtxMacro`, but chopped into separate macros.

`\@if@DeclareRobustCommand` checks if a command `\cmd` was defined by `\DeclareRobustCommand`. The test returns true if the expansion of `\cmd` is exactly `\protect\cmd␣`.

```

523 \long\def\@if@DeclareRobustCommand#1{%
524   \begingroup
525     \escapechar='\\
526     \edef\reserved@a{\string#1}%
527     \edef\reserved@b{\detokenize{#1}}%
528     \xdef\@gtempa{%
529       \ifx\reserved@a\reserved@b
530         \noexpand\x@protect
531         \noexpand#1%
532       \fi
533       \noexpand\protect
534       \expandafter\noexpand\csname\@expl@cs@to@str@N#1 \endcsname}%
535   \endgroup
536   \ifx\@gtempa#1\relax
537     \expandafter\@firstoftwo
538   \else
539     \expandafter\@secondoftwo
540   \fi}

```

If a command was defined by `\DeclareRobustCommand` (that is, `\@if@DeclareRobustCommand` returns true), then to make a copy of `\cmd` into `\foo` we define the latter such that it expands to `\protect\foo␣`, then make `\foo␣` equal to `\cmd␣`.

There is one detail we need to take care of: if a command was defined with `\DeclareRobustCommand` it may still have an optional argument, in which case there is one more macro layer before the actual definition of the command. We use `\@if@newcommand` to check that and `\@copy@newcommand` to do the copying.

```

541 \long\def\@copy@DeclareRobustCommand#1#2{%

```

```

542 \begingroup
543   \escapechar='\\
544   \edef\reserved@a{\string#1}%
545   \edef\reserved@b{\detokenize{#1}}%
546   \edef\reserved@a{%
547 \endgroup
548 \def\noexpand#1{%
549   \ifx\reserved@a\reserved@b
550     \noexpand\x@protect
551     \noexpand#1%
552   \fi
553   \noexpand\protect
554   \expandafter\noexpand\csname\@expl@cs@to@str@@N#1 \endcsname}%
555 \noexpand\copy@kernel@robust@command
556   \expandafter\noexpand\csname\@expl@cs@to@str@@N#1 \endcsname
557   \expandafter\noexpand\csname\@expl@cs@to@str@@N#2 \endcsname}%
558 \reserved@a}
559 \long\def\copy@kernel@robust@command#1#2{%
560   \robust@command@chk@safe#2%
561   {\@if@newcommand#2%
562     {\@copy@newcommand}%
563     {\declare@commandcopy@let}}
564   {\declare@commandcopy@let}%
565   #1#2}

```

Showing the command is pretty simple. This command prints the top-level expansion as \TeX 's `\show` would, but with `robust macro:` rather than just `macro:`, then a blank line and then `\show` the inner command. For a macro defined with, say, `\DeclareRobustCommand\foo[1]{bar}`, it will print:

```

> \foo=robust macro:
->\protect \foo .

> \foo =\long macro:
#1->bar.

```

If the inner command is defined with an optional argument, then `\@show@newcommand` is also used.

The value of `\escapechar` is deliberately not enforced, so `\ShowCommand` behaves more like `\show`.

```

566 \long\def\@show@DeclareRobustCommand#1{%
567   \typeout{> \string#1=robust macro:}%
568   \typeout{->\@expl@cs@replacement@spec@@N#1.^~J}%
569   \expandafter\show@kernel@robust@command
570   \csname\@expl@cs@to@str@@N#1 \endcsname}
571 \long\def\show@kernel@robust@command#1{%
572   \robust@command@chk@safe#1%
573   {\@if@newcommand#1%
574     {\@show@newcommand}%
575     {\show}}}%
576   {\show}%
577   #1}

```

(End definition for \@if@DeclareRobustCommand and others.)

1.5.4 Commands defined with \newcommand (with optional argument)

`\@if@newcommand` A command `\cmd` (or `\cmd␣`, if it was defined with `\DeclareRobustCommand`) with an optional argument will expand to `\@protected@testopt\cmd\cmd{<opt>}`. To check that we look at the first three tokens in the expansion of `\cmd`, and return true or false accordingly.

This test *requires* that the command be a parameterless macro, otherwise it will not work (and probably break). This is ensured with `\robust@command@chk@safe` before calling `\@if@newcommand`.

```

578 \long\def\@if@newcommand#1{%
579   \edef\reserved@a{%
580     \noexpand\@protected@testopt
581     \noexpand#1%
582     \expandafter\noexpand\csname\@backslashchar\@expl@cs@to@str@@N#1\endcsname}%
583   \edef\reserved@b{%
584     \unexpanded\expandafter\expandafter\expandafter
585     {\expandafter\@carcube#1{}{}{}\@nil}}%
586   \ifx\reserved@a\reserved@b
587     \expandafter\@firstoftwo
588   \else
589     \expandafter\@secondoftwo
590   \fi}

```

Then, if a command `\cmd` takes an optional argument, we copy it to `\foo` by defining the latter to expand to `\@protected@testopt\foo\foo{<opt>}`.

`\@copy@newcommand`

```

591 \long\def\@copy@newcommand#1#2{%
592   \edef#1{\noexpand\@protected@testopt
593     \noexpand#1%
594     \expandafter\noexpand\csname\@backslashchar\@expl@cs@to@str@@N#1\endcsname
595     \unexpanded\expandafter\expandafter\expandafter
596     {\expandafter\@gobblethree#2}}%
597   \expandafter
598   \let\csname\@backslashchar\@expl@cs@to@str@@N#1\expandafter\endcsname
599   \csname\@backslashchar\@expl@cs@to@str@@N#2\endcsname}

```

A command being `\shown` here is guaranteed to have an optional argument. Start by showing the top-level expansion of the command (using `\typeout` to avoid TeX asking for interaction and extra context lines), then call `\@show@newcommand@aux` with the internal command, which contains the actual definition, and with the expansion of the command to extract the default value of the optional argument.

`\@show@newcommand`
`\@show@newcommand@aux`

```

600 \long\def\@show@newcommand#1{%
601   \typeout{> \string#1=robust macro:}%
602   \typeout{->\@expl@cs@replacement@spec@@N#1.^^J}%
603   \expandafter\@show@newcommand@aux
604   \csname\@backslashchar\@expl@cs@to@str@@N#1\expandafter\endcsname
605   \expandafter{#1}}

```

For a macro defined with, say, `\newcommand\foo[1][opt]{bar}`, it will print:

```

> \foo=robust macro:
->\@protected@testopt \foo \foo {opt}.

> \foo=\long macro:

```

```
> default #1=opt.
[#1]->bar.
```

If the command was defined with `\DeclareRobustCommand`, then another pair of lines show the top-level expansion `\protect_\foo__`.

The extra gymnastics with `\showtokens` ensures that `\showtokens` itself, and the internals of this macro aren't showed in the context lines.

```
606 \long\def\@show@newcommand@aux#1#2{%
607   \typeout{> \string#1=\@expl@cs@prefix@spec@N#1macro:}%
608   \edef\reserved@a{%
609     default \string##1=\expandafter\detokenize\@gobblethree#2.^~J%
610     \@expl@cs@argument@spec@N#1->\@expl@cs@replacement@spec@N#1}%
611   \showtokens\expandafter\expandafter\expandafter{\expandafter\reserved@a}}
```

Now the rollback code.

```
612 </2ekernel | latexrelease>
613 <latexrelease>\EndIncludeInRelease
614 <latexrelease>\IncludeInRelease{0000-00-00}{\@if@DeclareRobustCommand}
615 <latexrelease> {Undefine \@if@DeclareRobustCommand, \@if@newcommand,
616 <latexrelease>         \@copy@DeclareRobustCommand, \@copy@newcommand,
617 <latexrelease>         \@show@DeclareRobustCommand, \@show@newcommand}%
618 <latexrelease>\let\@if@DeclareRobustCommand\@undefined
619 <latexrelease>\let\@copy@DeclareRobustCommand\@undefined
620 <latexrelease>\let\@show@DeclareRobustCommand\@undefined
621 <latexrelease>\let\@if@newcommand\@undefined
622 <latexrelease>\let\@copy@newcommand\@undefined
623 <latexrelease>\let\@show@newcommand\@undefined
624 %
625 <latexrelease>\let\copy@kernel@robust@command\@undefined
626 <latexrelease>\let\show@kernel@robust@command\@undefined
627 <latexrelease>\let\@show@newcommand@aux\@undefined
628 <latexrelease>\EndIncludeInRelease
629 <*2ekernel>
```

(End definition for \@if@newcommand and others.)

1.6 Internal defining commands

These commands are used internally to define other L^AT_EX commands.

`\@ifundefined` Check if first arg is undefined or `\relax` and execute second or third arg depending,

```
630 </2ekernel>
631 <latexrelease>\IncludeInRelease{2018-04-01}{\@ifundefined}
632 <latexrelease>\Leave commands undefined in \@ifundefined}%
633 <*2ekernel | latexrelease>
```

Version using `\ifcsname` to avoid defining undefined tokens to `\relax`. Defined here to simplify using unmatched `\fi`.

```
634 \def\@ifundefined#1{%
635   \ifcsname#1\endcsname\@ifundefin@d@i\else\@ifundefin@d@ii\fi{#1}}
636 \long\def\@ifundefin@d@i#1\fi#2{\fi
637   \expandafter\ifx\csname #2\endcsname\relax
638     \@ifundefin@d@ii
639     \fi
640     \@secondoftwo}
```

```
641 \long\def\@ifundefin@d@ii\fi#1#2#3{\fi #2}
```

Now test of engine.

```
642 \ifx\numexpr\@undefined
```

Classic version (should not be needed as etex is assumed).

```
643 \def\@ifundefined#1{%
644   \expandafter\ifx\csname#1\endcsname\relax
645     \expandafter\@firstoftwo
646   \else
647     \expandafter\@secondoftwo
648   \fi}
649 \else\ifx\directlua\@undefined
```

Use the \ifcsname defined above.

```
650 \else
```

Optimised version for Lua_T_EX, using \lastnamedcs

```
651 \def\@ifundefined#1{%
652   \ifcsname#1\endcsname
653     \expandafter\ifx\lastnamedcs\relax\else\@ifundefin@d@i\fi
654   \fi
655   \@firstoftwo}

656 \long\def\@ifundefin@d@i#1#2#3#4#5{#1#2#5}

657 \fi
658 \fi
659 </2ekernel | latexrelease>
660 <latexrelease>\EndIncludeInRelease
661 <latexrelease>\IncludeInRelease{0000-00-00}{\@ifundefined}
662 <latexrelease>\Leave commands undefined in \@ifundefined}%
663 <latexrelease>\def\@ifundefined#1{%
664 <latexrelease>   \expandafter\ifx\csname#1\endcsname\relax
665 <latexrelease>     \expandafter\@firstoftwo
666 <latexrelease>   \else
667 <latexrelease>     \expandafter\@secondoftwo
668 <latexrelease>   \fi}
669 <latexrelease>\EndIncludeInRelease
670 <*2ekernel>
```

(End definition for \@ifundefined.)

\@qend The following define \@qend and \@qrelax to be the strings ‘end’ and ‘relax’ with the
\@qrelax characters \catcoded 12.

```
671 \edef\@qend{\expandafter\@cdr\string\end\@nil}
672 \edef\@qrelax{\expandafter\@cdr\string\relax\@nil}
```

(End definition for \@qend and \@qrelax.)

\@ifnextchar **\@ifnextchar** peeks at the following character and compares it with its first argument.
 If both are the same it executes its second argument, otherwise its third.

```
673 \long\def\@ifnextchar#1#2#3{%
674   \let\reserved@d=#1%
675   \def\reserved@a{#2}%
676   \def\reserved@b{#3}%
677   \futurelet\@let@token\@ifnch}
```


(End definition for \@ifnextchar.)

\kernel@ifnextchar This macro is the kernel version of \@ifnextchar which is used in a couple of places to prevent the AMS variant from being used since in some places this produced chaos (for example if an `fd` file is loaded in a random place then the optional argument to `\ProvidesFile` could get printed there instead of being written only in the log file. This happened when there was a space or a newline between the mandatory and optional arguments! It should really be fixed in the `amsmath` package one day, but...

Note that there may be other places in the kernel where this version should be used rather than the original, but variable, version.

```
678 \let\kernel@ifnextchar\@ifnextchar
```

(End definition for \kernel@ifnextchar.)

\@ifnch \@ifnch is a tricky macro to skip any space tokens that may appear before the character in question. If it encounters a space token, it calls xifnch.

```
679 \def\@ifnch{%
680   \ifx\@let@token\@sptoken
681     \let\reserved@c\@xifnch
682   \else
683     \ifx\@let@token\reserved@d
684       \let\reserved@c\reserved@a
685     \else
686       \let\reserved@c\reserved@b
687     \fi
688   \fi
689   \reserved@c}
```

(End definition for \@ifnch.)

\@sptoken The following code makes \@sptoken a space token. It is important here that the control sequence `\:` consists of a non-letter only, so that the following whitespace is significant. Together with the fact that the equal sign in a `\let` may be followed by only one optional space the desired effect is achieved. NOTE: the following hacking must precede the definition of `\:` as math medium space.

```
690 \def\:{\let\@sptoken= } \: % this makes \@sptoken a space token
```

(End definition for \@sptoken.)

\@xifnch In the following definition of \@xifnch, `\:` is again used to get a space token as delimiter into the definition.

```
691 \def\:{\@xifnch} \expandafter\def\:{\futurelet\@let@token\@ifnch}
```

(End definition for \@xifnch.)

\@ifstar The new implementation below avoids passing the *⟨true code⟩* Through one more `\def` than the *⟨false code⟩*, which previously meant that `#` had to be written as `####` in one argument, but `##` in the other. The `*` is gobbled by \@firstoftwo.

```
692 \def\@ifstar#1{\@ifnextchar *{\@firstoftwo{#1}}}
```

(End definition for \@ifstar.)

```

\@dblarg
\@xdblarg 693 \long\def\@dblarg#1{\kernel@ifnextchar[{#1}{\@xdblarg{#1}}}{
694 \long\def\@xdblarg#1#2{#1[{#2}]{#2}}

```

(End definition for \@dblarg and \@xdblarg.)

\@sanitize The command \@sanitize changes the catcode of all special characters except for braces to ‘other’. It can be used for commands like \index that want to write their arguments verbatim. Needless to say, this command should only be executed within a group, or chaos will ensue.

```

695 \def\@sanitize{\@makeother\ \@makeother\\\@makeother\$\@makeother\&%
696 \@makeother\#\@makeother\^\@makeother\_ \@makeother\%\@makeother\~}

```

(End definition for \@sanitize.)

\@onelevel@sanitize This makes the whole “meaning” of #1 (its one-level expansion) into catcode 12 tokens: it could be used in \DeclareRobustCommand.

If it is to be used on default float specifiers, this should be done when they are defined.

```

697 \def \@onelevel@sanitize #1{%
698   \edef #1{\expandafter\strip@prefix
699             \meaning #1}%
700 }

```

(End definition for \@onelevel@sanitize.)

\string@makeletter Iterates through a string, turning each alphabetic character into a catcode-11 token (partly undoes a \detokenize). Useful for \ifx-based string comparisons where \detokenize-ing the other string would break too much code.

The macro uses expl3’s \@expl@str@map@function@NN to iterate on the string (without losing spaces) and applies \string@makeletter on each character. The latter checks if character is between a–z or A–Z, and uses \@alph or \@Alph to get the corresponding catcode-11 token. Other tokens are passed through unchanged.

```

701 </2ekernel>
702 <latexrelease>\IncludeInRelease{2020/10/01}{\string@makeletter}
703 <latexrelease> {Add \string@makeletter}%
704 <*2ekernel | latexrelease>
705 \def\string@makeletter#1{%
706   \@expl@str@map@function@NN#1\@string@makeletter}
707 \def\@string@makeletter#1{%
708   \@char@if@alph{#1}%
709   {\@expl@char@generate@nn{‘#1’}{11}}%
710   {#1}}
711 \def\@char@if@alph#1{%
712   \ifnum0\ifnum‘#1<‘A 1\fi\ifnum‘#1>‘z 1\fi
713   \if\ifnum‘#1>‘Z @\fi\ifnum‘#1<‘a @\fi01\fi>0
714   \expandafter\@secondoftwo
715   \else
716     \expandafter\@firstoftwo
717   \fi}
718 </2ekernel | latexrelease>
719 <latexrelease>\EndIncludeInRelease
720 %

```

```

721 <latexrelease>\IncludeInRelease{0000/00/00}{\string@makeletter}
722 <latexrelease> {Undefine \string@makeletter}%
723 <latexrelease>\let\string@makeletter\@undefined
724 <latexrelease>\let\@string@makeletter\@undefined
725 <latexrelease>\let\char@if@alph\@undefined
726 <latexrelease>\EndIncludeInRelease
727 <*2ekernel>

```

(End definition for \string@makeletter, \@string@makeletter, and \char@if@alph.)

`\makeatletter` Make internal control sequences accessible or inaccessible.

```

\makeatother 728 \DeclareRobustCommand\makeatletter{\catcode'\@11\relax}
729 \DeclareRobustCommand\makeatother{\catcode'\@12\relax}

```

(End definition for \makeatletter and \makeatother.)

2 Discretionary Hyphenation

`\-`
`\@dischph`

Moved here to be after the definition of `\DeclareRobustCommand`.

The primitive `\-` command adds a discretionary hyphen using the current font's `\hyphenchar`. Monospace fonts are usually declared with `\hyphenchar` set to `-1` to suppress hyphenation.

L^AT_EX, from L^AT_EX 2.09 in 1986 defined `\-` by

```
\def\-\{\discretionary{-}{ }{ }}
```

The following comment was added when these commands were first set up, 19 April 1986:

the `\-` command is redefined to allow it to work in the `\ttfamily` type style, where automatic hyphenation is suppressed by setting `\hyphenchar` to `-1`. The original primitive T_EX definition is saved as `\@@hyph` just in case anyone needs it.

L^AT_EX 2_ε, between 1993 and 2017, had a comment at this point saying that the definition “would probably change” because the definition always uses `-`. The definition used below was given in comments at this point during time.

In 2017 we finally enabled this definition by default, with the older L^AT_EX definition accessible via `latexrelease` as usual.

In LuaL^AT_EX the primitive definition of `\-` is used directly because it's use of extended hyphenation parameters means that `\-` works correctly even with `\hyphenchar` set to `-1`. This change makes `\-` under LuaL^AT_EX compatible with language specific hyphenation characters.

Temporary definition of `\@latex@info`, final definition is later.

```

730 \def\@latex@info#1{}
731 </2ekernel>
732 <latexrelease>\IncludeInRelease{2020/10/01}{\-\}{Use primitive \- in Lua\LaTeX}%
733 <*2ekernel | latexrelease>
734 \ifx\directlua\@undefined
735   \DeclareRobustCommand{\-\}{%
736     \discretionary{%
737       \char \ifnum\hyphenchar\font<\z@

```

```

738         \defaultthyphenchar
739     \else
740         \hyphenchar\font
741     \fi
742     }{}{}%
743 }
744 \else
745     \let\-\@@hyph
746 \fi
747 </2ekernel | latexrelease>
748 <latexrelease>\EndIncludeInRelease
749 <latexrelease>\IncludeInRelease{2017/04/15}{\-\}{Use \hyphenchar in \-}%
750 <latexrelease>\DeclareRobustCommand{\-\}{}%
751 <latexrelease> \discretionary{%
752 <latexrelease>     \char \ifnum\hyphenchar\font<\z@
753 <latexrelease>         \defaultthyphenchar
754 <latexrelease>     \else
755 <latexrelease>         \hyphenchar\font
756 <latexrelease>     \fi
757 <latexrelease>         }{}{}%
758 <latexrelease>}
759 <latexrelease>\EndIncludeInRelease
760 <latexrelease>\IncludeInRelease{0000/00/00}{\-\}{Use \hyphenchar in \-}%
761 <latexrelease>\def\-\{\discretionary{\-\}{}{}}
762 <latexrelease>\EndIncludeInRelease

763 <*2ekernel | latexrelease>
764 \let\@dischyph=\-
765 </2ekernel | latexrelease>
766 <*2ekernel>

(End definition for \- and \@dischyph.)
    Delayed from ltvers.dtx

767 \newif\if@includeinrelease
768 \@includeinreleasefalse

    Delayed from ltplain.dtx

769 </2ekernel>
770 <*2ekernel | latexrelease>
771 <latexrelease>\IncludeInRelease{2019/10/01}%
772 <latexrelease>         {\allowbreak}{Make various commands robust}%
773 \MakeRobust\allowbreak
774 \MakeRobust\bigbreak
775 \MakeRobust\break
776 \MakeRobust\dotfill
777 \MakeRobust\frenchspacing
778 \MakeRobust\goodbreak
779 \MakeRobust\hrulefill
780 \MakeRobust\medbreak
781 \MakeRobust\nobreak
782 \MakeRobust\nonfrenchspacing
783 \MakeRobust\obeylines
784 \MakeRobust\obeyspaces
785 \MakeRobust\slash
786 \MakeRobust\smallbreak

```

```

787 \MakeRobust\strut
788 \MakeRobust\underbar
789 </2ekernel | latexrelease>
790 <latexrelease>\EndIncludeInRelease
791 <latexrelease>\IncludeInRelease{0000/00/00}%
792 <latexrelease>                {\allowbreak}{Make various commands robust}%
793 <latexrelease>
794 <latexrelease>\kernel@make@fragile\allowbreak
795 <latexrelease>\kernel@make@fragile\bigbreak
796 <latexrelease>\kernel@make@fragile\break
797 <latexrelease>\kernel@make@fragile\dotfill
798 <latexrelease>\kernel@make@fragile\frenchspacing
799 <latexrelease>\kernel@make@fragile\goodbreak
800 <latexrelease>\kernel@make@fragile\hrulefill
801 <latexrelease>\kernel@make@fragile\medbreak
802 <latexrelease>\kernel@make@fragile\nobreak
803 <latexrelease>\kernel@make@fragile\nonfrenchspacing
804 <latexrelease>\kernel@make@fragile\obeylines
805 <latexrelease>\kernel@make@fragile\obeyspaces
806 <latexrelease>\kernel@make@fragile\slash
807 <latexrelease>\kernel@make@fragile\smallbreak
808 <latexrelease>\kernel@make@fragile\strut
809 <latexrelease>\kernel@make@fragile\underbar
810 <latexrelease>
811 <latexrelease>\EndIncludeInRelease
812 <*2ekernel>

```

`\g@addto@macro` Globally add to the end of a macro. This macro is used by the kernel to add to its internal hooks.

```

813 \long\def\g@addto@macro#1#2{%
814   \begingroup
815     \toks@\expandafter{#1#2}%
816     \xdef#1{\the\toks@}%
817   \endgroup}

```

(End definition for \g@addto@macro.)

```

818 </2ekernel>

```

File g

ltxcmd.dtx

1 Creating document commands

Document commands should be created using the tools provided by this module: `\NewDocumentCommand`, etc., in almost all cases. This allows clean separation of document-level syntax from code-level interfaces. Users have a need to create new document commands, and as such a significant amount of documentation for `ltxcmd` is provided as part of `usrguide3`. Here, additional material aimed at programmers is provided

```

1 <@@=cmd>
2 <*2ekernel>
3 \message{document commands,}
4 </2ekernel>

```

`ltxcmd` code contains an `^^@` character, which usually has catcode 15, so `\IncludeInRelease` will break when this code is being skipped, so we'll save the catcode of `^^@` to restore later:

```

5 <*2ekernel | latexrelease>
6 <latexrelease>\edef\@latexrelease@catcode@null{\the\catcode'\^^@ }
7 <latexrelease>\catcode'\^^@=12
8 \ExplSyntaxOn
9 <latexrelease>\NewModuleRelease{2020/10/01}{ltxcmd}
10 <latexrelease>{Document~command~parser}%

```

1.1 Variables and constants

<u><code>\l__cmd_arg_spec_tl</code></u>	Holds the argument specification after normalization of shorthands.
	11 \tl_new:N \l__cmd_arg_spec_tl
<u><code>\l__cmd_args_tl</code></u>	Token list variable for grabbed arguments.
	12 \tl_new:N \l__cmd_args_tl
<u><code>\l__cmd_args_i_tl</code></u> <u><code>\l__cmd_args_ii_tl</code></u>	Hold the modified arguments when dealing with default values or processors.
	13 \tl_new:N \l__cmd_args_i_tl
	14 \tl_new:N \l__cmd_args_ii_tl
<u><code>\l__cmd_current_arg_int</code></u>	The number of the current argument being set up: this is used to make sure there are at most 9 arguments, then for creating the expandable auxiliary functions and knowing how many arguments the code function should take.
	15 \int_new:N \l__cmd_current_arg_int

<u>\l__cmd_defaults_bool</u> <u>\l__cmd_defaults_tl</u>	<p>The boolean indicates whether there are any argument with default value other than <code>-NoValue-</code>; the token list holds the code to determine these default values in terms of other arguments.</p> <pre> 16 \bool_new:N \l__cmd_defaults_bool 17 \tl_new:N \l__cmd_defaults_tl </pre>
<u>\l__cmd_environment_bool</u>	<p>Generating environments uses the same mechanism as generating functions. However, full processing of arguments is always needed for environments, and so the function-generating code needs to know this. This variable is also used at run time to give correct error messages.</p> <pre> 18 \bool_new:N \l__cmd_environment_bool </pre>
<u>\l__cmd_environment_str</u>	<p>Name of the environment, used at definition time and at run time.</p> <pre> 19 \str_new:N \l__cmd_environment_str </pre>
<u>\l__cmd_expandable_bool</u>	<p>Used to indicate if an expandable command is begin generated, as this affects both the acceptable argument types and how they are implemented.</p> <pre> 20 \bool_new:N \l__cmd_expandable_bool </pre>
<u>\l__cmd_expandable_aux_name_tl</u>	<p>Used to create pretty-printing names for the auxiliaries: although the immediate definition does not vary, the full expansion does and so it does not count as a constant.</p> <pre> 21 \tl_new:N \l__cmd_expandable_aux_name_tl 22 \tl_set:Nn \l__cmd_expandable_aux_name_tl 23 { 24 \l__cmd_function_tl \c_space_tl 25 (arg~ \int_use:N \l__cmd_current_arg_int) 26 } </pre>
<u>\g__cmd_grabber_int</u>	<p>Used (in exceptional cases) to get unique names for grabbers used by expandable commands.</p> <pre> 27 \int_new:N \g__cmd_grabber_int </pre>
<u>\l__cmd_fn_tl</u>	<p>For passing the pre-formed name of the auxiliary to be used as the parsing function.</p> <pre> 28 \tl_new:N \l__cmd_fn_tl </pre>
<u>\l__cmd_fn_code_tl</u>	<p>For passing the pre-formed name of the auxiliary that contains the actual code.</p> <pre> 29 \tl_new:N \l__cmd_fn_code_tl </pre>

<u>\l__cmd_function_tl</u>	<p>Holds the control sequence name of the function currently being defined: used to avoid passing this as an argument and to avoid repeated use of <code>\cs_to_str:N</code>.</p> <pre>30 \tl_new:N \l__cmd_function_tl</pre>
<u>\l__cmd_grab_expandably_bool</u>	<p>When defining a non-expandable command, indicates whether the arguments can all safely be grabbed by expandable grabbers. This is to support abuses of <code>xparse</code> that use protected functions inside <code>csname</code> constructions.</p> <pre>31 \bool_new:N \l__cmd_grab_expandably_bool</pre>
<u>\l__cmd_obey_spaces_bool</u>	<p>For trailing optionals.</p> <pre>32 \bool_new:N \l__cmd_obey_spaces_bool</pre>
<u>\l__cmd_last_delimiters_tl</u>	<p>Holds the delimiters (first tokens) of all optional arguments since the previous mandatory argument, to warn about cases where it would be impossible to omit optional arguments completely because the following mandatory argument has the same delimiter as one of the optional arguments.</p> <pre>33 \tl_new:N \l__cmd_last_delimiters_tl</pre>
<u>\l__cmd_long_bool</u>	<p>Used to indicate that an argument is long, on a per-argument basis.</p> <pre>34 \bool_new:N \l__cmd_long_bool</pre>
<u>\l__cmd_m_args_int</u>	<p>The number of <code>m</code> arguments: if this is the same as the total number of arguments, then a short-cut can be taken in the creation of the grabber code.</p> <pre>35 \int_new:N \l__cmd_m_args_int</pre>
<u>\l__cmd_prefixed_bool</u>	<p>When preparing the signature of non-expandable commands, indicates that the current argument is affected by a processor or by <code>+</code> (namely is long).</p> <pre>36 \bool_new:N \l__cmd_prefixed_bool</pre>
<u>\l__cmd_process_all_tl</u> <u>\l__cmd_process_one_tl</u> <u>\l__cmd_process_some_bool</u>	<p>When preparing the signature, the processors that will be applied to a given argument are collected in <code>\l__cmd_process_one_tl</code>, while <code>\l__cmd_process_all_tl</code> contains processors for all arguments. The boolean indicates whether there are any processors (to bypass the whole endeavour otherwise).</p> <pre>37 \tl_new:N \l__cmd_process_all_tl 38 \tl_new:N \l__cmd_process_one_tl 39 \bool_new:N \l__cmd_process_some_bool</pre>

\l__cmd_saved_args_tl

Stores \l__cmd_args_tl to deal with space-trimming of b-type arguments.

```
40 \tl_new:N \l__cmd_saved_args_tl
```

\l__cmd_signature_tl

Used when constructing the signature (code for argument grabbing) to hold what will become the implementation of the main function. When arguments are grabbed (at point of use of the command/environment), it also stores the code for grabbing the remaining arguments.

```
41 \tl_new:N \l__cmd_signature_tl
```

\l__cmd_some_obey_spaces_bool**\l__cmd_some_long_bool****\l__cmd_some_short_bool**

These flags are set while normalizing the argument specification. The `obey_spaces` one is used to detect when `!` is used on an argument that is not a trailing optional argument. The other two are used to check whether all short arguments appear before long arguments: this is needed to grab arguments expandably. As soon as the first long argument is seen (other than t-type, whose long status is ignored) the `some_long` flag is set. The `some_short` flag is used for expandable commands, to know whether to define a short auxiliary too.

```
42 \bool_new:N \l__cmd_some_obey_spaces_bool
```

```
43 \bool_new:N \l__cmd_some_long_bool
```

```
44 \bool_new:N \l__cmd_some_short_bool
```

\l__cmd_tmp_prop**\l__cmd_tmppa_tl****\l__cmd_tmppb_tl**

Scratch space.

```
45 \prop_new:N \l__cmd_tmp_prop
```

```
46 \tl_new:N \l__cmd_tmppa_tl
```

```
47 \tl_new:N \l__cmd_tmppb_tl
```

```
48 \cs_new_eq:NN \__cmd_tmp:w ?
```

(End definition for __cmd_tmp:w.)

With `xparse`, information about commands being (re)defined was switched off by default, unless the `log-declarations` package option was used, so here we'll switch that off as well.

```
49 \msg_redirect_module:nnn { cmd } { info } { none }
```

Also add `cmd` to the LaTeX messages.

```
50 \prop_gput:Nnn \g_msg_module_type_prop { cmd } { LaTeX }
```

1.2 Declaring commands and environments

The main functions for creating commands set the appropriate flag then use the same internal code to do the definition.

__cmd_declare_cmd:Nnn**__cmd_declare_expandable_cmd:Nnn****__cmd_declare_cmd_aux:Nnn****__cmd_declare_cmd_internal:Nnnn****__cmd_declare_cmd_internal:cnxn**

```
51 \cs_new_protected:Npn \__cmd_declare_cmd:Nnn
```

```
52 {
```

```
53   \bool_set_false:N \l__cmd_expandable_bool
```

```

54   \__cmd_declare_cmd_aux:Nnn
55 }
56 \cs_new_protected:Npn \__cmd_declare_expandable_cmd:Nnn
57 {
58   \bool_set_true:N \l__cmd_expandable_bool
59   \__cmd_declare_cmd_aux:Nnn
60 }

```

The first stage is to log information, both for the user in the log and for programmatic use in a property list of all declared commands.

```

61 \cs_new_protected:Npn \__cmd_declare_cmd_aux:Nnn #1#2#3
62 {
63   \cs_if_exist:NTF #1
64   {
65     \msg_info:nxxx { cmd } { redefine-command }
66     { \token_to_str:N #1 } { \tl_to_str:n {#2} }
67   }
68   {
69     \bool_lazy_or:nnT
70     { \cs_if_exist_p:c { \cs_to_str:N #1 ~ code } }
71     { \cs_if_exist_p:c { \cs_to_str:N #1 ~ defaults } }
72     {
73       \msg_warning:nxx { cmd } { unsupported-let }
74       { \token_to_str:N #1 }
75     }
76     \msg_info:nxxx { cmd } { define-command }
77     { \token_to_str:N #1 } { \tl_to_str:n {#2} }
78   }
79   \bool_set_false:N \l__cmd_environment_bool
80   \__cmd_declare_cmd_internal:Nnnn #1 {#2} {#3} { }
81 }

```

At definition time, the variable `\l__cmd_fn_tl` is only used for error messages. The real business of defining a document command starts with setting up the appropriate name, then normalizing the argument specification to get rid of shorthands.

```

82 \cs_new_protected:Npn \__cmd_declare_cmd_internal:Nnnn #1#2#3#4
83 {
84   \tl_set:Nx \l__cmd_function_tl { \cs_to_str:N #1 }
85   \tl_set:Nx \l__cmd_fn_tl
86   { \exp_not:c { \l__cmd_function_tl \c_space_tl } }
87   \__cmd_normalize_arg_spec:n {#2}
88   \exp_args:No \__cmd_prepare_signature:n \l__cmd_arg_spec_tl
89   \__cmd_declare_cmd_code:Nnn #1 {#2} {#3}
90   #4
91   \__cmd_break_point:n {#2}
92 }
93 \cs_generate_variant:Nn \__cmd_declare_cmd_internal:Nnnn { cnx }

```

(End definition for `__cmd_declare_cmd:Nnn` and others.)

`__cmd_break_point:n` A marker used to escape from creating a definition if necessary.

```

94 \cs_new_eq:NN \__cmd_break_point:n \use_none:n

```

(End definition for `__cmd_break_point:n`.)

```

\__cmd_declare_cmd_code:Nnn
  \__cmd_declare_cmd_code_aux:Nnn
  \__cmd_declare_cmd_code_expandable:Nnn

```

The appropriate auxiliary is called.

```

95 \cs_new_protected:Npn \__cmd_declare_cmd_code:Nnn
96 {
97   \bool_if:NTF \l__cmd_grab_expandably_bool
98   { \__cmd_declare_cmd_code_expandable:Nnn }
99   { \__cmd_declare_cmd_code_aux:Nnn }
100 }

```

Standard functions call `__cmd_start:nNNnnn`, which receives the argument specification, an auxiliary used for grabbing arguments, an auxiliary containing the code, and then the signature, default arguments, and processors.

```

101 \cs_new_protected:Npn \__cmd_declare_cmd_code_aux:Nnn #1#2#3
102 {
103   \cs_generate_from_arg_count:cNnn
104   { \l__cmd_function_tl \c_space_tl code }
105   \cs_set_protected:Npn \l__cmd_current_arg_int {#3}
106   \cs_set_protected_nopar:Npx #1
107   {
108     \bool_if:NTF \l__cmd_environment_bool
109     {
110       \__cmd_start_env:nnnnn { \exp_not:n {#2} }
111       { \l__cmd_environment_str }
112     }
113     {
114       \__cmd_start:nNNnnn { \exp_not:n {#2} }
115       \exp_not:c { \l__cmd_function_tl \c_space_tl }
116       \exp_not:c { \l__cmd_function_tl \c_space_tl code }
117     }
118     { \exp_not:o \l__cmd_signature_tl }
119     {
120       \bool_if:NT \l__cmd_defaults_bool
121       { \exp_not:o \l__cmd_defaults_tl }
122     }
123     {
124       \bool_if:NT \l__cmd_process_some_bool
125       { \exp_not:o \l__cmd_process_all_tl }
126     }
127   }
128 }

```

Expandable functions and functions whose arguments can be grabbed expandably call `__cmd_start_expandable:nNNNNn`, which receives the argument specification, four auxiliaries (two for grabbing arguments, one for the code, and one for default arguments), and finally the signature. Non-expandable functions that take this branch should nevertheless be protected, as well as their `code` function. They will only be expanded in contexts such as constructing a `csname`. The two grabbers (named after the function with one or two spaces) are needed when there are both short and long arguments; otherwise the same grabber is included twice in the definition. If all arguments are long or all are short the (only) grabber is defined correspondingly to be long/short. Otherwise two grabbers are defined, one long, one short.

```

129 \cs_new_protected:Npn \__cmd_declare_cmd_code_expandable:Nnn #1#2#3
130 {
131   \exp_args:Ncc \cs_generate_from_arg_count:NNnn
132   { \l__cmd_function_tl \c_space_tl code }

```

```

133     { cs_set \bool_if:NF \l__cmd_expandable_bool { _protected } :Npn }
134     \l__cmd_current_arg_int {#3}
135   \bool_if:NT \l__cmd_defaults_bool
136   {
137     \use:x
138     {
139       \cs_generate_from_arg_count:cNnn
140       { \l__cmd_function_tl \c_space_tl defaults }
141       \cs_set:Npn \l__cmd_current_arg_int
142       { \exp_not:o \l__cmd_defaults_tl }
143     }
144   }
145   \bool_if:NTF \l__cmd_expandable_bool
146   { \cs_set_nopar:Npx } { \cs_set_protected_nopar:Npx } #1
147   {
148     \exp_not:N \__cmd_start_expandable:nNNNNn
149     { \exp_not:n {#2} }
150     \exp_not:c { \l__cmd_function_tl \c_space_tl }
151     \exp_not:c
152     {
153       \l__cmd_function_tl \c_space_tl
154       \bool_if:NT \l__cmd_some_short_bool
155       { \bool_if:NT \l__cmd_some_long_bool { \c_space_tl } }
156     }
157     \exp_not:c { \l__cmd_function_tl \c_space_tl code }
158     \bool_if:NTF \l__cmd_defaults_bool
159     { \exp_not:c { \l__cmd_function_tl \c_space_tl defaults } }
160     { ? }
161     { \exp_not:o \l__cmd_signature_tl }
162   }
163   \bool_if:NTF \l__cmd_some_long_bool
164   {
165     \bool_if:NT \l__cmd_some_short_bool
166     {
167       \cs_set_nopar:cpx { \l__cmd_function_tl \c_space_tl \c_space_tl }
168       ##1##2 { ##1 {##2} }
169     }
170     \cs_set:cpx
171   }
172   { \cs_set_nopar:cpx }
173   { \l__cmd_function_tl \c_space_tl } ##1##2 { ##1 {##2} }
174 }

```

(End definition for __cmd_declare_cmd_code:Nnn, __cmd_declare_cmd_code_aux:Nnn, and __cmd_declare_cmd_code_expandable:Nnn.)

__cmd_declare_env:nnnn
 __cmd_declare_env_internal:nnnn

The lead-off to creating an environment is much the same as that for creating a command: issue the appropriate message, store the argument specification then hand off to an internal function.

```

175 \cs_new_protected:Npn \__cmd_declare_env:nnnn #1#2
176 {
177   \str_set:Nx \l__cmd_environment_str {#1}
178   \str_set:Nx \l__cmd_environment_str
179   { \tl_trim_spaces:o { \l__cmd_environment_str } }

```

```

180 \cs_if_exist:cTF { \l__cmd_environment_str }
181 {
182     \msg_info:nnxx { cmd } { redefine-environment }
183     { \l__cmd_environment_str } { \tl_to_str:n {#2} }
184 }
185 {
186     \msg_info:nnxx { cmd } { define-environment }
187     { \l__cmd_environment_str } { \tl_to_str:n {#2} }
188 }
189 \bool_set_false:N \l__cmd_expandable_bool
190 \bool_set_true:N \l__cmd_environment_bool
191 \exp_args:NV \__cmd_declare_env_internal:nnnn
192     \l__cmd_environment_str {#2}
193 }

```

Creating a document environment requires a few more steps than creating a single command. In order to pass the arguments of the command to the end of the function, it is necessary to store the grabbed arguments. To do that, the function used at the end of the environment has to be redefined to contain the appropriate information. To minimize the amount of expansion at point of use, the code here is expanded now as well as when used. The last argument of `__cmd_declare_cmd_internal:Nnnn` is only run if the definition succeeded. In package mode this ensures that the original definition of the environment is not changed if the definition fails for any reason. This also avoids an error when defining the `end_au_x_` function when the user asks for more than 9 arguments.

```

194 \cs_new_protected:Npn \__cmd_declare_env_internal:nnnn #1#2#3#4
195 {
196     \__cmd_declare_cmd_internal:cnxn { environment~ #1 } {#2}
197     {
198         \cs_set_nopar:Npx \exp_not:c { environment~ #1 ~end~aux }
199         {
200             \exp_not:N \exp_not:N \exp_not:c { environment~ #1~end~aux~ }
201             \exp_not:n { \exp_not:o \l__cmd_args_tl }
202         }
203         \exp_not:n {#3}
204     }
205     {
206         \cs_set_nopar:cpx { environment~ #1 ~end }
207         { \exp_not:c { environment~ #1 ~end~aux } }
208         \cs_generate_from_arg_count:cNnn
209             { environment~ #1 ~end~aux~ } \cs_set:Npn
210             \l__cmd_current_arg_int {#4}
211         \cs_set_eq:cc {#1} { environment~ #1 }
212         \cs_set_eq:cc { end #1 } { environment~ #1 ~end }
213     }
214 }

```

(End definition for `__cmd_declare_env:nnnn` and `__cmd_declare_env_internal:nnnn`.)

1.3 Structure of `xparse` commands

`__cmd_start_env:nnnnn` For error messages that occur during run-time when getting arguments of environments
`__cmd_start:nNNnnn` it is necessary to keep track of the environment name. We begin non-expandable commands with a token equal to `\scan_stop:`, whose name gives a reasonable error message if the command is used inside a `csname` and protects against `f`-expansion. This is useless

for environments since `\begin` is already not expandable. Both the command and environment codes start with `\group_align_safe_begin:`, then `__cmd_run_code:` (used by both) does `\group_align_safe_end:`, so that delimited arguments may be grabbed in alignments if they contain an alignment tab token (see latex3/latex3/issues/839).

```

215 \cs_new_protected:Npn \__cmd_start_env:nnnnn #1#2
216 {
217   \conditionally@traceoff
218   \group_align_safe_begin:
219   \str_set:Nn \l__cmd_environment_str {#2}
220   \bool_set_true:N \l__cmd_environment_bool
221   \__cmd_start_aux:ccnnnn
222   { environment~ \l__cmd_environment_str \c_space_tl }
223   { environment~ \l__cmd_environment_str \c_space_tl code }
224   {#1}
225 }
226 \cs_new_protected:Npx \__cmd_start:nNNnnn #1#2#3
227 {
228   \exp_not:c { xparse~function~is~not~expandable }
229   \exp_not:N \conditionally@traceoff
230   \exp_not:N \group_align_safe_begin:
231   \exp_not:n { \bool_set_false:N \l__cmd_environment_bool }
232   \exp_not:N \__cmd_start_aux:NNnnnn
233   #2 #3 {#1}
234 }

```

(End definition for `__cmd_start_env:nnnnn` and `__cmd_start:nNNnnn`.)

`__cmd_start_aux:NNnnnn` This sets up a few variables to minimize the boilerplate code included in all xparse-defined commands. It then runs the grabbers #4. Again, the argument specification #1 is only for diagnostics.

`__cmd_start_aux:ccnnnn`

```

235 \cs_new_protected:Npn \__cmd_start_aux:NNnnnn #1#2#3#4#5#6
236 {
237   \tl_clear:N \l__cmd_args_tl
238   \tl_set:Nn \l__cmd_fn_tl {#1}
239   \tl_set:Nn \l__cmd_fn_code_tl {#2}
240   \tl_set:Nn \l__cmd_defaults_tl {#5}
241   \tl_set:Nn \l__cmd_process_all_tl {#6}
242   #4 \__cmd_run_code:
243 }
244 \cs_generate_variant:Nn \__cmd_start_aux:NNnnnn { cc }

```

(End definition for `__cmd_start_aux:NNnnnn`.)

`__cmd_run_code:` After arguments are grabbed, this function is responsible for inserting default values, running processors, and finally doing `\group_align_safe_end:` as promised, and running the code.

```

245 \cs_new_protected:Npn \__cmd_run_code:
246 {
247   \tl_if_empty:NF \l__cmd_defaults_tl { \__cmd_defaults: }
248   \tl_if_empty:NF \l__cmd_process_all_tl { \__cmd_args_process: }
249   \group_align_safe_end:
250   \conditionally@traceon
251   \exp_after:wN \l__cmd_fn_code_tl \l__cmd_args_tl
252 }

```

(End definition for `_cmd_run_code:`.)

`_cmd_defaults:` First construct `_cmd_tmp:w` (see below) that will receive the arguments found so far and determine default values for any missing argument. Then call it repeatedly until the set of arguments stabilizes. Since that could lead to an infinite loop we only call it up to nine times, the maximal number needed for stabilization if there is a chain of arguments that depend on each other. If that fails to stabilize raise an error.

```

\__cmd_defaults_error:w
253 \cs_new_protected:Npn \__cmd_defaults:
254 {
255   \__cmd_defaults_def:
256   \tl_set_eq:NN \l__cmd_args_i_tl \l__cmd_args_tl
257   \__cmd_defaults_aux: \__cmd_defaults_aux: \__cmd_defaults_aux:
258   \__cmd_defaults_aux: \__cmd_defaults_aux: \__cmd_defaults_aux:
259   \__cmd_defaults_aux: \__cmd_defaults_aux: \__cmd_defaults_aux:
260   \__cmd_defaults_error:w
261   \q_recursion_stop
262   \tl_set_eq:NN \l__cmd_args_tl \l__cmd_args_i_tl
263 }
264 \cs_new_protected:Npn \__cmd_defaults_aux:
265 {
266   \tl_set:Nx \l__cmd_args_ii_tl
267   { \exp_after:wN \__cmd_tmp:w \l__cmd_args_i_tl }
268   \tl_if_eq:NNT \l__cmd_args_ii_tl \l__cmd_args_i_tl
269   { \use_none_delimit_by_q_recursion_stop:w }
270   \tl_set_eq:NN \l__cmd_args_i_tl \l__cmd_args_ii_tl
271 }
272 \cs_new_protected:Npn \__cmd_defaults_error:w \q_recursion_stop
273 {
274   \msg_error:nnx { cmd } { loop-in-defaults }
275   { \__cmd_environment_or_command: }
276 }

```

To construct `_cmd_tmp:w`, first go through the arguments found and the corresponding defaults, building a token list with `{#<arg number>}` for arguments found in the input (whose default will not be used) and otherwise `{\exp_not:n{<default>}}` for arguments whose default will be used.

```

277 \cs_new_protected:Npn \__cmd_defaults_def:
278 {
279   \tl_clear:N \l__cmd_tmpa_tl
280   \int_zero:N \l__cmd_current_arg_int
281   \__cmd_tl_mapthread_function:NNN \l__cmd_args_tl \l__cmd_defaults_tl
282   \__cmd_defaults_def:nn
283   \cs_generate_from_arg_count:NNVo \__cmd_tmp:w \cs_set:Npn
284   \l__cmd_current_arg_int \l__cmd_tmpa_tl
285 }
286 \cs_generate_variant:Nn \cs_generate_from_arg_count:NNnn { NNVo }
287 \cs_new_protected:Npn \__cmd_defaults_def:nn
288 {
289   \int_incr:N \l__cmd_current_arg_int
290   \exp_args:NV \__cmd_defaults_def:nnn \l__cmd_current_arg_int
291 }
292 \cs_new_protected:Npn \__cmd_defaults_def:nnn #1#2#3
293 {
294   \tl_put_right:Nx \l__cmd_tmpa_tl

```

```

295     {
296     {
297         \exp_not:N \exp_not:n
298         {
299             \tl_if_novalue:nTF {#2}
300             { \exp_not:o {#3} }
301             { \exp_not:n { ## #1 } }
302         }
303     }
304 }
305 }

```

(End definition for `__cmd_defaults:` and others.)

`__cmd_args_process:` Loop through arguments (stored in `\l__cmd_args_tl`) and the corresponding processors (in `\l__cmd_process_all_tl`) simultaneously, apply all processors for each argument and store the result back into `\l__cmd_args_tl`. To allow processors to depend on other arguments, for every processor define a temporary auxiliary that receives all arguments `\l__cmd_args_tl`.

```

306 \cs_new_protected:Npn \__cmd_args_process:
307 {
308     \tl_clear:N \l__cmd_args_ii_tl
309     \__cmd_tl_mapthread_function:NNN
310     \l__cmd_args_tl
311     \l__cmd_process_all_tl
312     \__cmd_args_process_loop:nn
313     \tl_set_eq:NN \l__cmd_args_tl \l__cmd_args_ii_tl
314 }
315 \cs_new_protected:Npn \__cmd_args_process_loop:nn #1#2
316 {
317     \tl_set:Nn \ProcessedArgument {#1}
318     \tl_if_novalue:nF {#1}
319     { \tl_map_function:nN {#2} \__cmd_args_process_aux:n }
320     \tl_put_right:No \l__cmd_args_ii_tl
321     { \exp_after:wN { \ProcessedArgument } }
322 }
323 \cs_new_protected:Npn \__cmd_args_process_aux:n #1
324 {
325     \cs_generate_from_arg_count:NNnn \__cmd_tmp:w \cs_set:Npn
326     { \tl_count:N \l__cmd_args_tl } {#1}
327     \exp_args:NNNo \exp_after:wN \__cmd_tmp:w \l__cmd_args_tl
328     { \ProcessedArgument }
329 }

```

(End definition for `__cmd_args_process:`, `__cmd_args_process_loop:nn`, and `__cmd_args_process_aux:n`.)

`__cmd_start_expandable:nNNNNn` This is called for all expandable commands. **#6** is the signature, responsible for grabbing arguments. **#5** is used to determine default values (or is ? if there are none). **#4** is the code to run. **#2** and **#3** are functions (named after the command) that grab a single argument in the input stream (**#3** is short). The argument specification **#1** is only used by diagnostic functions. Same as for the non-expandable version, this starts with `\group_align_safe_begin:`, which expands to nothing, so may be safely used in an expandable context.


```

330 \cs_new:Npn \__cmd_start_expandable:nnNNNn #1#2#3#4#5#6
331 {
332   \group_align_safe_begin:
333   #6 \__cmd_end_expandable:NNw #5 #4 \q__cmd #2#3
334 }

```

(End definition for __cmd_start_expandable:nnNNNn.)

```

\__cmd_end_expandable:NNw
\__cmd_end_expandable_aux:w
  \__cmd_end_expandable_aux:nnNNN
\__cmd_end_expandable_defaults:nnnNNn
  \__cmd_end_expandable_defaults:nnw
  \__cmd_end_expandable_defaults:nw

```

Followed by a function #1 to determine default values (or ? if there are no defaults), the code #2, arguments that have been grabbed, then \q__cmd and two generic grabbers. The idea to find default values is similar to the non-expandable case but we cannot define an auxiliary function, so at every step in the loop we need to go through all arguments searching for which ones started out as -NoValue- and replacing these by the newly computed values. In fact we need to keep track of three versions of all arguments: the original version, the previous version with default values, and the currently built version (first argument of __cmd_end_expandable_defaults:nnnNNn).

```

335 \cs_new:Npn \__cmd_end_expandable:NNw #1#2
336 { \__cmd_end_expandable_aux:w #1#2 \prg_do_nothing: }
337 \cs_new:Npn \__cmd_end_expandable_aux:w #1#2#3 \q__cmd
338 { \exp_args:No \__cmd_end_expandable_aux:nnNNN {#3} #1 #2 }
339 \cs_new:Npn \__cmd_end_expandable_aux:nnNNN #1#2#3#4#5
340 {
341   \token_if_eq_charcode:NNT ? #2 { \exp_after:wN \use_iv:nnnn }
342   \__cmd_end_expandable_defaults:nnnNNn {#1} { } {#1} #2#3
343   { } { } { } { } { } { } { } { } { } { } { } { }
344   {
345     \msg_expandable_error:nmf { cmd } { loop-in-defaults }
346     { \exp_args:Nf \tl_trim_spaces:n { \token_to_str:N #4 } }
347     \use_iv:nnnn
348   }
349   \q_stop
350 }
351 \cs_new:Npn \__cmd_end_expandable_defaults:nnnNNn #1#2#3#4#5#6
352 {
353   #6
354   \str_if_eq:nnTF {#1} {#2}
355   { \use_i_delimit_by_q_stop:nw { \group_align_safe_end: #5 #1 } }
356   {
357     \exp_args:No \__cmd_tl_mapthread_function:nnN
358     { #4 #1 } {#3}
359     \__cmd_end_expandable_defaults:nnw
360     \__cmd_end_expandable_defaults:nnnNNn { } {#1} {#3} #4 #5
361   }
362 }
363 \cs_new:Npn \__cmd_end_expandable_defaults:nnw #1#2
364 {
365   \tl_if_novalue:nTF {#2}
366   { \exp_args:No \__cmd_end_expandable_defaults:nw {#1} }
367   { \__cmd_end_expandable_defaults:nw {#2} }
368 }
369 \cs_new:Npn \__cmd_end_expandable_defaults:nw
370 #1#2 \__cmd_end_expandable_defaults:nnnNNn #3
371 { #2 \__cmd_end_expandable_defaults:nnnNNn { #3 {#1} } }

```

(End definition for __cmd_end_expandable:NNw and others.)

1.4 Normalizing the argument specifications

The goal here is to expand aliases and check that the argument specification is valid before the main parsing run. If it is not valid the entire set up is abandoned to avoid any strange internal errors. A function is provided for each argument type that will grab any extra data items and call the loop function after performing the following checks and tasks.

- Check that each argument has the correct number of data items associated with it, and that where a single character is required, one has actually been supplied.
- Check that processors and the markers + and ! are followed by an argument for which they make sense, and are not redundant.
- Check the absence of forbidden types for expandable commands, namely G/v always, and l/u after optional arguments (xparse may have inserted braces due to a failed search for an optional argument).
- Check that no optional argument is followed by a mandatory argument with the same delimiter, as otherwise the optional argument could never be omitted.
- Keep track in \l__cmd_some_long_bool and \l__cmd_some_short_bool of whether the command has some long/short arguments.
- Keep track in \l__cmd_grab_expandably_bool of whether all arguments are m/l/u type and short arguments appear before long ones, in which case they can be grabbed expandably just as safely as they could be grabbed nonexpandably. Regardless of that, arguments of expandable commands will be grabbed expandably and arguments of environments will not (because the list of arguments built by non-expandable grabbing is used to pass them to the end-environment code).

Further checks happen at the end of the loop:

- that there are at most 9 arguments;
- that an expandable command does not end with an optional argument (this case is detected by using the fact that \l__cmd_last_delimiters_tl is cleared by every mandatory argument and filled by every optional argument).

```

\__cmd_normalize_arg_spec:n Loop through the argument specification, calling an auxiliary specific to each argument
\__cmd_normalize_arg_spec_loop:n type. If any argument is unknown stop the definition.
372 \cs_new_protected:Npn \__cmd_normalize_arg_spec:n #1
373 {
374   \int_zero:N \l__cmd_current_arg_int
375   \tl_clear:N \l__cmd_last_delimiters_tl
376   \tl_clear:N \l__cmd_arg_spec_tl
377   \bool_set_true:N \l__cmd_grab_expandably_bool
378   \bool_set_false:N \l__cmd_obey_spaces_bool
379   \bool_set_false:N \l__cmd_long_bool
380   \bool_set_false:N \l__cmd_some_obey_spaces_bool
381   \bool_set_false:N \l__cmd_some_long_bool
382   \bool_set_false:N \l__cmd_some_short_bool
383   \__cmd_normalize_arg_spec_loop:n #1
384   \q_recursion_tail \q_recursion_tail \q_recursion_tail \q_recursion_stop
385   \int_compare:nNnT \l__cmd_current_arg_int > 9

```

```

386     {
387         \msg_error:nnxx { cmd } { too-many-arguments }
388         { \__cmd_environment_or_command: } { \tl_to_str:n {#1} }
389         \__cmd_bad_def:wn
390     }
391 \bool_if:NT \l__cmd_expandable_bool
392 {
393     \tl_if_empty:NF \l__cmd_last_delimiters_tl
394     {
395         \msg_error:nnxx { cmd } { expandable-ending-optional }
396         { \iow_char:N \ \ \l__cmd_function_tl } { \tl_to_str:n {#1} }
397         \__cmd_bad_def:wn
398     }
399 }
400 \bool_if:NT \l__cmd_expandable_bool
401 { \bool_set_true:N \l__cmd_grab_expandably_bool }
402 \bool_if:NT \l__cmd_environment_bool
403 { \bool_set_false:N \l__cmd_grab_expandably_bool }
404 }
405 \cs_new_protected:Npn \__cmd_normalize_arg_spec_loop:n #1
406 {
407     \quark_if_recursion_tail_stop:n {#1}
408     \int_incr:N \l__cmd_current_arg_int
409     \cs_if_exist_use:cF { __cmd_normalize_type_ \tl_to_str:n {#1} :w }
410     {
411         \bool_lazy_any:nTF
412         {
413             { \str_if_eq_p:nn {#1} { G } }
414             { \str_if_eq_p:nn {#1} { g } }
415             { \str_if_eq_p:nn {#1} { l } }
416             { \str_if_eq_p:nn {#1} { u } }
417         }
418         {
419             \msg_error:nnxx { cmd } { xparse-argument-type }
420             { \__cmd_environment_or_command: } { \tl_to_str:n {#1} }
421         }
422         {
423             \msg_error:nnxx { cmd } { unknown-argument-type }
424             { \__cmd_environment_or_command: } { \tl_to_str:n {#1} }
425         }
426         \__cmd_bad_def:wn
427     }
428 }

```

(End definition for __cmd_normalize_arg_spec:n and __cmd_normalize_arg_spec_loop:n.)

__cmd_normalize_type_d:w These argument types are aliases of more general ones, for example with the default argument -NoValue-. To easily insert that marker expanded in the definitions we call __cmd_tmp:w with the argument -NoValue-. For argument types that need additional data, check that the data is present (not \q_recursion_tail) before proceeding.

```

429 \cs_set_protected:Npn \__cmd_tmp:w #1
430 {
431     \cs_new_protected:Npn \__cmd_normalize_type_d:w ##1##2
432     {

```

```

433     \quark_if_recursion_tail_stop_do:nn {##2} { \__cmd_bad_arg_spec:wn }
434     \__cmd_normalize_type_D:w {##1} {##2} {#1}
435   }
436   \cs_new_protected:Npn \__cmd_normalize_type_e:w ##1
437   {
438     \quark_if_recursion_tail_stop_do:nn {##1} { \__cmd_bad_arg_spec:wn }
439     \__cmd_normalize_type_E:w {##1} { }
440   }
441   \cs_new_protected:Npn \__cmd_normalize_type_o:w
442   { \__cmd_normalize_type_D:w [ ] {#1} }
443   \cs_new_protected:Npn \__cmd_normalize_type_O:w
444   { \__cmd_normalize_type_D:w [ ] }
445   \cs_new_protected:Npn \__cmd_normalize_type_r:w ##1##2
446   {
447     \quark_if_recursion_tail_stop_do:nn {##2} { \__cmd_bad_arg_spec:wn }
448     \__cmd_normalize_type_R:w {##1} {##2} {#1}
449   }
450   \cs_new_protected:Npn \__cmd_normalize_type_s:w
451   { \__cmd_normalize_type_t:w * }
452 }
453 \exp_args:No \__cmd_tmp:w { \c_novalue_tl }

```

(End definition for __cmd_normalize_type_d:w and others.)

__cmd_normalize_type_>:w
 __cmd_normalize_type_+:w
 __cmd_normalize_type_!:w

Check that these prefixes have arguments, namely that the next token is not \q-recursion_tail, and remember to leave it after the looping macro. Processors are forbidden in expandable commands. If all is good, store the prefix in the cleaned up \l__cmd_arg_spec_tl, and decrement the argument number as prefixes do not correspond to arguments.

```

454 \cs_new_protected:cpn { __cmd_normalize_type_>:w } #1#2
455 {
456   \quark_if_recursion_tail_stop_do:nn {#2} { \__cmd_bad_arg_spec:wn }
457   \bool_if:NT \l__cmd_expandable_bool
458   {
459     \msg_error:nnxx { cmd } { processor-in-expandable }
460     { \iow_char:N \ \ \l__cmd_function_tl } { \tl_to_str:n {#1} }
461     \__cmd_bad_def:wn
462   }
463   \tl_put_right:Nx \l__cmd_arg_spec_tl { > { \tl_trim_spaces:n {#1} } }
464   \int_decr:N \l__cmd_current_arg_int
465   \bool_set_false:N \l__cmd_grab_expandably_bool
466   \__cmd_normalize_arg_spec_loop:n {#2}
467 }
468 \cs_new_protected:cpn { __cmd_normalize_type_+:w } #1
469 {
470   \quark_if_recursion_tail_stop_do:nn {#1} { \__cmd_bad_arg_spec:wn }
471   \bool_if:NT \l__cmd_long_bool
472   {
473     \msg_error:nnxx { cmd } { two-markers }
474     { \__cmd_environment_or_command: } { + }
475     \__cmd_bad_def:wn
476   }
477   \bool_set_true:N \l__cmd_long_bool
478   \int_decr:N \l__cmd_current_arg_int

```

```

479   \__cmd_normalize_arg_spec_loop:n {#1}
480   }
481 \cs_new_protected:cpn { __cmd_normalize_type_! :w } #1
482   {
483     \quark_if_recursion_tail_stop_do:nn {#1} { \__cmd_bad_arg_spec:wn }
484     \bool_if:NT \l__cmd_obey_spaces_bool
485     {
486       \msg_error:nnxx { cmd } { two-markers }
487       { \__cmd_environment_or_command: } { ! }
488       \__cmd_bad_def:wn
489     }
490     \bool_set_true:N \l__cmd_obey_spaces_bool
491     \bool_set_true:N \l__cmd_some_obey_spaces_bool
492     \int_decr:N \l__cmd_current_arg_int
493     \__cmd_normalize_arg_spec_loop:n {#1}
494   }

```

(End definition for __cmd_normalize_type_>:w, __cmd_normalize_type_+:w, and __cmd_normalize_type_! :w.)

__cmd_normalize_type_D:w
 __cmd_normalize_type_E:w
 __cmd_normalize_type_t:w
 __cmd_normalize_E_unique_check:w

Optional argument types. Check that all required data is present (and consists of single characters if applicable) and check for forbidden types for expandable commands. For E-type require that there is at least one embellishment, that each one is a single character, and that there aren't more optional arguments than embellishments; also remember that each embellishment counts as one argument for \l__cmd_current_arg_int. Then in each case store the data in \l__cmd_arg_spec_tl, and for later checks store in \l__cmd_last_delimiters_tl the tokens whose presence determines whether there is an optional argument (for braces store {}, seen later as an empty delimiter).

```

495 \cs_new_protected:Npn \__cmd_normalize_type_D:w #1#2#3
496   {
497     \quark_if_recursion_tail_stop_do:nn {#3} { \__cmd_bad_arg_spec:wn }
498     \__cmd_single_token_check:n {#1} \__cmd_allowed_token_check:N #1
499     \__cmd_single_token_check:n {#2}
500     \__cmd_add_arg_spec:n { D #1 #2 {#3} }
501     \tl_put_right:Nn \l__cmd_last_delimiters_tl {#1}
502     \bool_set_false:N \l__cmd_grab_expandably_bool
503     \__cmd_normalize_arg_spec_loop:n
504   }
505 \cs_new_protected:Npn \__cmd_normalize_type_E:w #1#2
506   {
507     \quark_if_recursion_tail_stop_do:nn {#2} { \__cmd_bad_arg_spec:wn }
508     \tl_if_blank:nT {#1} { \__cmd_bad_arg_spec:wn }
509     \tl_map_function:nN {#1} \__cmd_single_token_check:n
510     \tl_map_function:nN {#1} \__cmd_allowed_token_check:N
511     \__cmd_normalize_E_unique_check:w #1 \q_nil \q_stop
512     \int_compare:nNnT { \tl_count:n {#2} } > { \tl_count:n {#1} }
513       { \__cmd_bad_arg_spec:wn }
514     \__cmd_add_arg_spec:n { E {#1} {#2} }
515     \tl_put_right:Nn \l__cmd_last_delimiters_tl {#1}
516     \bool_set_false:N \l__cmd_grab_expandably_bool
517     \int_add:Nn \l__cmd_current_arg_int { \tl_count:n {#1} - 1 }
518     \__cmd_normalize_arg_spec_loop:n
519   }
520 \cs_new_protected:Npn \__cmd_normalize_E_unique_check:w #1#2 \q_stop

```

```

521 {
522   \quark_if_nil:NF #1
523   {
524     \tl_if_in:nnT {#2} {#1} { \__cmd_bad_arg_spec:wn }
525     \__cmd_normalize_E_unique_check:w #2 \q_stop
526   }
527 }
528 \cs_new_protected:Npn \__cmd_normalize_type_t:w #1
529 {
530   \quark_if_recursion_tail_stop_do:Nn #1 { \__cmd_bad_arg_spec:wn }
531   \__cmd_single_token_check:n {#1} \__cmd_allowed_token_check:N #1
532   \tl_put_right:Nx \l__cmd_arg_spec_tl
533   {
534     \bool_if:NT \l__cmd_obey_spaces_bool { ! }
535     t \exp_not:n {#1}
536   }
537   \tl_put_right:Nn \l__cmd_last_delimiters_tl {#1}
538   \bool_set_false:N \l__cmd_grab_expandably_bool
539   \bool_set_false:N \l__cmd_obey_spaces_bool
540   \bool_set_false:N \l__cmd_long_bool
541   \__cmd_normalize_arg_spec_loop:n
542 }

```

(End definition for __cmd_normalize_type_D:w and others.)

__cmd_normalize_type_m:w
 __cmd_normalize_type_R:w
 __cmd_normalize_type_v:w

Mandatory arguments. First check the required data is present, consists of single characters where applicable, and that the argument type is allowed for expandable commands if applicable. For the m and R argument types check that they do not follow some optional argument with that delimiter as otherwise the optional argument could not be omitted. Then save data in \l__cmd_arg_spec_tl, count the mandatory argument, and empty the list of last delimiters.

```

543 \cs_new_protected:Npn \__cmd_normalize_type_m:w
544 {
545   \__cmd_delimiter_check:nnn { } { m } { \iow_char:N \{ }
546   \__cmd_add_arg_spec_mandatory:n { m }
547   \__cmd_normalize_arg_spec_loop:n
548 }
549 \cs_new_protected:Npn \__cmd_normalize_type_R:w #1#2#3
550 {
551   \quark_if_recursion_tail_stop_do:nn {#3} { \__cmd_bad_arg_spec:wn }
552   \__cmd_single_token_check:n {#1} \__cmd_allowed_token_check:N #1
553   \__cmd_single_token_check:n {#2}
554   \__cmd_delimiter_check:nnn {#1} { R/r } { \tl_to_str:n {#1} }
555   \bool_set_false:N \l__cmd_grab_expandably_bool
556   \__cmd_add_arg_spec_mandatory:n { R #1 #2 {#3} }
557   \__cmd_normalize_arg_spec_loop:n
558 }
559 \cs_new_protected:Npn \__cmd_normalize_type_v:w
560 {
561   \__cmd_normalize_check_gv:N v
562   \__cmd_add_arg_spec_mandatory:n { v }
563   \__cmd_normalize_arg_spec_loop:n
564 }

```

(End definition for `_cmd_normalize_type_m:w`, `_cmd_normalize_type_R:w`, and `_cmd_normalize_type_v:w`.)

`_cmd_normalize_type_b:w` This argument type is not allowed for commands. This is only allowed at the end of the argument specification, hence we check that #1 is the end.

```

565 \cs_new_protected:Npn \_cmd_normalize_type_b:w #1
566 {
567   \bool_if:NF \l__cmd_environment_bool
568   {
569     \msg_error:nnxx { cmd } { invalid-command-arg }
570     { \_cmd_environment_or_command: } { b }
571     \_cmd_bad_def:wn
572   }
573   \tl_clear:N \l__cmd_last_delimiters_tl
574   \_cmd_add_arg_spec:n { b }
575   \quark_if_recursion_tail_stop:n {#1}
576   \msg_error:nnxx { cmd } { arg-after-body }
577   { \_cmd_environment_or_command: }
578   { \tl_to_str:n {#1} }
579   \_cmd_bad_def:wn
580 }

```

(End definition for `_cmd_normalize_type_b:w`.)

`_cmd_single_token_check:n` Checks that the argument is a single (non-space) token (possibly surrounded by spaces), and aborts the definition otherwise.

```

581 \cs_new_protected:Npn \_cmd_single_token_check:n #1
582 {
583   \tl_trim_spaces_apply:nN {#1} \tl_if_single_token:nF
584   {
585     \msg_error:nnxx { cmd } { not-single-token }
586     { \_cmd_environment_or_command: } { \tl_to_str:n {#1} }
587     \_cmd_bad_def:wn
588   }
589 }

```

(End definition for `_cmd_single_token_check:n`.)

`_cmd_allowed_token_check:N` Some tokens are now allowed as delimiters for some argument types, notably implicit begin/end-group tokens (`\bgroup`/`\egroup`). The major problem with these tokens is that for `\peek_...` functions, a literal `{_1}` is virtually indistinguishable from a `\bgroup` or other token which was `\let` to a `{_1}`, and the same goes for `_2`. All other tokens can be easily distinguished from their implicit counterparts by grabbing them and looking at the string length (see `_cmd_token_if_cs:NTF`), but for begin/end group tokens that is not possible without the risk of mistakenly grabbing the entire brace group (potentially leading to a ! Runaway argument error) or trying to grab a `_2`, leading to an ! Argument of `\dots` has an extra `}` error.

```

590 \cs_new_protected:Npn \_cmd_allowed_token_check:N #1
591 {
592   \token_if_eq_meaning:NNTF #1 \c_group_begin_token
593   { \use:n }
594   {
595     \token_if_eq_meaning:NNTF #1 \c_group_end_token
596     { \use:n }

```

```

597         { \use_none:n }
598     }
599 {
600     \msg_error:nnxxx { cmd } { forbidden-implicit-group-token }
601     { \__cmd_environment_or_command: } { \tl_to_str:n {#1} }
602     {
603         \token_if_eq_meaning:NNTF #1 \c_group_begin_token
604         { begin } { end }
605     }
606     \__cmd_bad_def:wn
607 }
608 }

```

(End definition for __cmd_allowed_token_check:N.)

__cmd_normalize_check_gv:N Called for arguments that are always forbidden, or forbidden after an optional argument,
__cmd_normalize_check_lu:N for expandable commands.

```

609 \cs_new_protected:Npn \__cmd_normalize_check_gv:N #1
610 {
611     \bool_if:NT \l__cmd_expandable_bool
612     {
613         \msg_error:nnxx { cmd } { invalid-expandable-argument-type }
614         { \iow_char:N \ \ \l__cmd_function_tl } { \tl_to_str:n {#1} }
615         \__cmd_bad_def:wn
616     }
617     \bool_set_false:N \l__cmd_grab_expandably_bool
618 }
619 \cs_new_protected:Npn \__cmd_normalize_check_lu:N #1
620 {
621     \bool_if:NT \l__cmd_expandable_bool
622     {
623         \tl_if_empty:NF \l__cmd_last_delimiters_tl
624         {
625             \msg_error:nnxx { cmd } { invalid-after-optional-expandably }
626             { \iow_char:N \ \ \l__cmd_function_tl } { \tl_to_str:n {#1} }
627             \__cmd_bad_def:wn
628         }
629     }
630 }

```

(End definition for __cmd_normalize_check_gv:N and __cmd_normalize_check_lu:N.)

__cmd_delimiter_check:nnn Called for m and R arguments. Checks that the leading token does not coincide with the
token denoting the presence of a previous optional argument. Instead of dealing with
braces for the m-type we use an empty delimiter to denote that case.

```

631 \cs_new_protected:Npn \__cmd_delimiter_check:nnn #1#2#3
632 {
633     \tl_map_inline:Nn \l__cmd_last_delimiters_tl
634     {
635         \tl_if_eq:nnT {##1} {#1}
636         {
637             \msg_warning:nnxx { cmd } { optional-mandatory }
638             {#2} {#3}
639         }
640     }

```



```

640     }
641 }

```

(End definition for `_cmd_delimiter_check:nnn`.)

`_cmd_bad_arg_spec:wn` If the argument specification is wrong, this provides an escape from the entire definition process.

```

642 \cs_new_protected:Npn \_cmd_bad_arg_spec:wn #1 \_cmd_break_point:n #2
643 {
644   \msg_error:nnxx { cmd } { bad-arg-spec }
645   { \_cmd_environment_or_command: } { \tl_to_str:n {#2} }
646 }
647 \cs_new_protected:Npn \_cmd_bad_def:wn #1 \_cmd_break_point:n #2 { }

```

(End definition for `_cmd_bad_arg_spec:wn` and `_cmd_bad_def:wn`.)

`_cmd_add_arg_spec:n` When adding an argument to the argument specification, set the `some_long` or `some_short` booleans as appropriate and clear the booleans keeping track of + and ! markers. Before that, test for a short argument following some long arguments: this is forbidden for expandable commands and prevents grabbing arguments expandably.

For mandatory arguments do some more work, in particular complain if they were preceded by !.

```

648 \cs_new_protected:Npn \_cmd_add_arg_spec:n #1
649 {
650   \bool_lazy_and:nnT
651   { ! \l__cmd_long_bool }
652   { \l__cmd_some_long_bool }
653   {
654     \bool_if:NT \l__cmd_expandable_bool
655     {
656       \msg_error:nnx { cmd } { inconsistent-long }
657       { \iow_char:N \ \ \l__cmd_function_tl }
658       \_cmd_bad_def:wn
659     }
660     \bool_set_false:N \l__cmd_grab_expandably_bool
661   }
662   \bool_if:NTF \l__cmd_long_bool
663   { \bool_set_true:N \l__cmd_some_long_bool }
664   { \bool_set_true:N \l__cmd_some_short_bool }
665   \tl_put_right:Nx \l__cmd_arg_spec_tl
666   {
667     \bool_if:NT \l__cmd_long_bool { + }
668     \bool_if:NT \l__cmd_obey_spaces_bool { ! }
669     \exp_not:n {#1}
670   }
671   \bool_set_false:N \l__cmd_long_bool
672   \bool_set_false:N \l__cmd_obey_spaces_bool
673 }
674 \cs_new_protected:Npn \_cmd_add_arg_spec_mandatory:n #1
675 {
676   \bool_if:NT \l__cmd_some_obey_spaces_bool
677   {
678     \msg_error:nnxx { cmd } { non-trailing-obey-spaces }
679     { \_cmd_environment_or_command: } { \tl_to_str:n {#1} }

```

```

680         \__cmd_bad_def:wn
681     }
682     \tl_clear:N \l__cmd_last_delimiters_tl
683     \__cmd_add_arg_spec:n {#1}
684 }

```

(End definition for __cmd_add_arg_spec:n and __cmd_add_arg_spec_mandatory:n.)

1.5 Preparing the signature: general mechanism

Actually creating the signature uses the same loop approach as normalizing the signature. There are first a number of variables which need to be set to track what is going on. Many of these variables are unused when defining expandable commands.

```

\__cmd_prepare_signature:n
\__cmd_prepare_signature:N
  \__cmd_prepare_signature_bypass:N
685 \cs_new_protected:Npn \__cmd_prepare_signature:n #1
686 {
687     \int_zero:N \l__cmd_current_arg_int
688     \bool_set_false:N \l__cmd_long_bool
689     \bool_set_false:N \l__cmd_obey_spaces_bool
690     \int_zero:N \l__cmd_m_args_int
691     \bool_set_false:N \l__cmd_defaults_bool
692     \tl_clear:N \l__cmd_defaults_tl
693     \tl_clear:N \l__cmd_process_all_tl
694     \tl_clear:N \l__cmd_process_one_tl
695     \bool_set_false:N \l__cmd_process_some_bool
696     \tl_clear:N \l__cmd_signature_tl
697     \__cmd_prepare_signature:N #1 \q_recursion_tail \q_recursion_stop
698     \bool_if:NF \l__cmd_expandable_bool { \__cmd_flush_m_args: }
699 }

```

The main looping function does not take an argument, but carries out the reset on the processor boolean. This is split off from the rest of the process so that when actually setting up processors the flag-reset can be bypassed.

For each known argument type there is an appropriate function to actually do the addition to the signature. These are separate for expandable and standard functions, as the approaches are different.

```

700 \cs_new_protected:Npn \__cmd_prepare_signature:N
701 {
702     \bool_set_false:N \l__cmd_prefixed_bool
703     \__cmd_prepare_signature_bypass:N
704 }
705 \cs_new_protected:Npn \__cmd_prepare_signature_bypass:N #1
706 {
707     \quark_if_recursion_tail_stop:N #1
708     \use:c
709     {
710         __cmd_add
711         \bool_if:NT \l__cmd_grab_expandably_bool { _expandable }
712         _type_ \token_to_str:N #1 :w
713     }
714 }

```

(End definition for __cmd_prepare_signature:n, __cmd_prepare_signature:N, and __cmd_prepare_signature_bypass:N.)

1.6 Setting up a standard signature

Each argument-adding function appends to the signature a grabber (and for some types, the delimiters or default value), except the one for `m` arguments. These are collected and added to the signature all at once by `__cmd_flush_m_args:`, called for every other argument type. All of the functions then call the loop function `__cmd_prepare_signature:N`. Default values of arguments are collected by `__cmd_add_default:n` rather than being stored with the argument; this function and `__cmd_add_default:` are also responsible for keeping track of `\l__cmd_current_arg_int`.

`__cmd_add_type_+:w` Making the next argument long means setting the flag. The `m` arguments are recorded here as this has to be done for every case where there is then a long argument.

```

715 \cs_new_protected:cpn { __cmd_add_type_+:w }
716 {
717   \__cmd_flush_m_args:
718   \bool_set_true:N \l__cmd_long_bool
719   \bool_set_true:N \l__cmd_prefixed_bool
720   \__cmd_prepare_signature_bypass:N
721 }
```

(End definition for __cmd_add_type_+:w.)

`__cmd_add_type_!:w` Much the same for controlling trailing optional arguments.

```

722 \cs_new_protected:cpn { __cmd_add_type_!:w }
723 {
724   \__cmd_flush_m_args:
725   \bool_set_true:N \l__cmd_obey_spaces_bool
726   \bool_set_true:N \l__cmd_prefixed_bool
727   \__cmd_prepare_signature_bypass:N
728 }
```

(End definition for __cmd_add_type_!:w.)

`__cmd_add_type_>:w` When a processor is found, the processor code is stored. It will be used by `__cmd_args_process:` once arguments are all found. Here too the loop calls `__cmd_prepare_signature_bypass:N` rather than `__cmd_prepare_signature:N` so that the flag is not reset.

```

729 \cs_new_protected:cpn { __cmd_add_type_>:w } #1
730 {
731   \__cmd_flush_m_args:
732   \bool_set_true:N \l__cmd_prefixed_bool
733   \bool_set_true:N \l__cmd_process_some_bool
734   \tl_put_left:Nn \l__cmd_process_one_tl { {#1} }
735   \__cmd_prepare_signature_bypass:N
736 }
```

(End definition for __cmd_add_type_>:w.)

`__cmd_add_type_b:w`

```

737 \cs_new_protected:Npn \__cmd_add_type_b:w
738 {
739   \__cmd_flush_m_args:
740   \__cmd_add_default:
741   \__cmd_add_grabber:N b
```

```

742     \__cmd_prepare_signature:N
743   }

```

(End definition for __cmd_add_type_b:w.)

__cmd_add_type_D:w

```

744 \cs_new_protected:Npn \__cmd_add_type_D:w #1#2#3
745 {
746   \__cmd_flush_m_args:
747   \__cmd_add_default:n {#3}
748   \__cmd_add_grabber:N D
749   \tl_put_right:Nn \l__cmd_signature_tl { #1 #2 }
750   \__cmd_prepare_signature:N
751 }

```

(End definition for __cmd_add_type_D:w.)

__cmd_add_type_E:w The E-type argument needs a special handling of default values.

```

752 \cs_new_protected:Npn \__cmd_add_type_E:w #1#2
753 {
754   \__cmd_flush_m_args:
755   \__cmd_add_default_E:nn {#1} {#2}
756   \__cmd_add_grabber:N E
757   \tl_put_right:Nn \l__cmd_signature_tl { {#1} }
758   \__cmd_prepare_signature:N
759 }

```

(End definition for __cmd_add_type_E:w.)

__cmd_add_type_m:w The m type is special as short arguments which are not post-processed are simply counted at this stage. Thus there is a check to see if either of these cases apply. If so, a one-argument grabber is added to the signature. On the other hand, if a standard short argument is required it is simply counted at this stage, to be added later using __cmd_flush_m_args:.

```

760 \cs_new_protected:Npn \__cmd_add_type_m:w
761 {
762   \__cmd_add_default:
763   \bool_if:NTF \l__cmd_prefixed_bool
764     { \__cmd_add_grabber:N m }
765     { \int_incr:N \l__cmd_m_args_int }
766   \__cmd_prepare_signature:N
767 }

```

(End definition for __cmd_add_type_m:w.)

__cmd_add_type_R:w The R-type argument is very similar to the D-type.

```

768 \cs_new_protected:Npn \__cmd_add_type_R:w #1#2#3
769 {
770   \__cmd_flush_m_args:
771   \__cmd_add_default:n {#3}
772   \__cmd_add_grabber:N R
773   \tl_put_right:Nn \l__cmd_signature_tl { #1 #2 }
774   \__cmd_prepare_signature:N
775 }

```

(End definition for `__cmd_add_type_R:w`.)

`__cmd_add_type_t:w` Setting up a `t` argument means collecting one token for the test, and adding it along with the grabber to the signature.

```

776 \cs_new_protected:Npn \__cmd_add_type_t:w #1
777 {
778   \__cmd_flush_m_args:
779   \__cmd_add_default:
780   \__cmd_add_grabber:N t
781   \tl_put_right:Nn \l__cmd_signature_tl {#1}
782   \__cmd_prepare_signature:N
783 }

```

(End definition for `__cmd_add_type_t:w`.)

`__cmd_add_type_v:w` At this stage, the `v` argument is identical to `l` except that since the grabber may fail to read a verbatim argument we need a default value.

```

784 \cs_new_protected:Npn \__cmd_add_type_v:w
785 {
786   \__cmd_flush_m_args:
787   \exp_args:No \__cmd_add_default:n \c_novalue_tl
788   \__cmd_add_grabber:N v
789   \__cmd_prepare_signature:N
790 }

```

(End definition for `__cmd_add_type_v:w`.)

`__cmd_flush_m_args:` As `m` arguments are simply counted, there is a need to add them to the token register in a block. As this function can only be called if something other than `m` turns up, the flag can be switched here.

```

791 \cs_new_protected:Npn \__cmd_flush_m_args:
792 {
793   \int_compare:nNnT \l__cmd_m_args_int > 0
794   {
795     \tl_put_right:Nx \l__cmd_signature_tl
796     { \exp_not:c { __cmd_grab_m_ \int_use:N \l__cmd_m_args_int :w } }
797     \tl_put_right:Nx \l__cmd_process_all_tl
798     { \prg_replicate:nn { \l__cmd_m_args_int } { { } } }
799   }
800   \int_zero:N \l__cmd_m_args_int
801 }

```

(End definition for `__cmd_flush_m_args:`.)

`__cmd_add_grabber:N` To keep the various checks needed in one place, adding the grabber to the signature is done here. The only questions are whether the grabber should be long or not, and whether to obey spaces. The `\l__cmd_obey_spaces_bool` boolean can only be `true` for trailing optional arguments. In that case spaces will not be ignored when looking for that optional argument.

```

802 \cs_new_protected:Npn \__cmd_add_grabber:N #1
803 {
804   \tl_put_right:Nx \l__cmd_signature_tl
805   {
806     \exp_not:c

```

```

807     {
808         __cmd_grab_ #1
809         \bool_if:NT \l__cmd_long_bool { _long }
810         \bool_if:NT \l__cmd_obey_spaces_bool { _obey_spaces }
811         :w
812     }
813 }
814 \bool_set_false:N \l__cmd_long_bool
815 \bool_set_false:N \l__cmd_obey_spaces_bool
816 \tl_put_right:Nx \l__cmd_process_all_tl
817 { { \exp_not:o \l__cmd_process_one_tl } }
818 \tl_clear:N \l__cmd_process_one_tl
819 }

```

(End definition for __cmd_add_grabber:N.)

__cmd_add_default:n Store the default value of an argument, or rather code that gives that default value (it may involve other arguments). This is \c_novalue_tl for arguments with no actual default or with default -NoValue-; and (in a brace group) \prg_do_nothing: followed by a default value for others. For E-type arguments, pad the defaults #2 with some \c_novalue_tl until there are as many as embellishments #1. These functions are also used when defining expandable commands.

```

820 \cs_new_protected:Npn \__cmd_add_default:n #1
821 {
822     \tl_if_novalue:nTF {#1}
823     { \__cmd_add_default: }
824     {
825         \int_incr:N \l__cmd_current_arg_int
826         \bool_set_true:N \l__cmd_defaults_bool
827         \tl_put_right:Nn \l__cmd_defaults_tl { { \prg_do_nothing: #1 } }
828     }
829 }
830 \cs_new_protected:Npn \__cmd_add_default:
831 {
832     \int_incr:N \l__cmd_current_arg_int
833     \tl_put_right:Nn \l__cmd_defaults_tl { \c_novalue_tl }
834 }
835 \cs_new_protected:Npn \__cmd_add_default_E:nn #1#2
836 {
837     \tl_map_function:nN {#2} \__cmd_add_default:n
838     \prg_replicate:nn
839     { \tl_count:n {#1} - \tl_count:n {#2} }
840     { \__cmd_add_default: }
841 }

```

(End definition for __cmd_add_default:n, __cmd_add_default:, and __cmd_add_default_E:nn.)

1.7 Setting up expandable types

The approach here is not dissimilar to that for standard types, but fewer types are supported. There is also a need to define the per-function auxiliaries: this is done here, while the general grabbers are dealt with later.

`_cmd_add_expandable_type_+ :w` We have already checked that short arguments are before long arguments, so `\l__cmd_long_bool` only changes from `false` to `true` once (and there is no need to reset it after each argument). Also knock back the argument count because `+` is not an argument. Continue the loop.

```

842 \cs_new_protected:cpn { \_cmd_add_expandable_type_+ :w }
843 {
844   \bool_set_true:N \l__cmd_long_bool
845   \__cmd_prepare_signature:N
846 }

```

(End definition for `_cmd_add_expandable_type_+ :w`.)

`_cmd_add_expandable_type_D :w` The set up for D-type arguments involves constructing a rather complex auxiliary which is used repeatedly when grabbing. There is an auxiliary here so that the R-type can share code readily: `#1` is D or R. The `_aux:NN` auxiliary is needed if the two delimiting tokens are identical: in contrast to the non-expandable route, the grabber here has to act differently for this case.

```

847 \cs_new_protected:Npn \_cmd_add_expandable_type_D :w
848 { \_cmd_add_expandable_type_D_aux:NNNn D }
849 \cs_new_protected:Npn \_cmd_add_expandable_type_D_aux:NNNn #1#2#3#4
850 {
851   \__cmd_add_default:n {#4}
852   \tl_if_eq:nnTF {#2} {#3}
853     { \_cmd_add_expandable_type_D_aux:NN #1 #2 }
854     { \_cmd_add_expandable_type_D_aux:NNN #1 #2 #3 }
855   \__cmd_prepare_signature:N
856 }
857 \cs_new_protected:Npn \_cmd_add_expandable_type_D_aux:NNN #1#2#3
858 {
859   \bool_if:NTF \l__cmd_long_bool
860     { \cs_set:cpx }
861     { \cs_set_nopar:cpx }
862     { \l__cmd_expandable_aux_name_tl } ##1 ##2 #2 ##3 \q__cmd ##4 #3
863     { ##1 {##2} {##3} {##4} }
864   \_cmd_add_expandable_grabber:nn {#1}
865   {
866     \exp_not:c { \l__cmd_expandable_aux_name_tl }
867     \exp_not:n { #2 #3 }
868   }
869 }
870 \cs_new_protected:Npn \_cmd_add_expandable_type_D_aux:NN #1#2
871 {
872   \bool_if:NTF \l__cmd_long_bool
873     { \cs_set:cpx }
874     { \cs_set_nopar:cpx }
875     { \l__cmd_expandable_aux_name_tl } ##1 #2 ##2 #2
876     { ##1 {##2} }
877   \_cmd_add_expandable_grabber:nn { #1_alt }
878   {
879     \exp_not:c { \l__cmd_expandable_aux_name_tl }
880     \exp_not:n { #2 }
881   }
882 }

```

(End definition for _cmd_add_expandable_type_D:w and others.)

_cmd_add_expandable_type_E:w
_cmd_add_expandable_type_E_aux:n

For each embellishment, use _cmd_get_grabber:NN to obtain an auxiliary delimited by that token and store a pair constituted of the auxiliary and the token in \l_cmd_tmpb_tl, before appending the whole set of these pairs to the signature, and an equal number of -NoValue- markers (regardless of the default values of arguments). Set the current argument appropriately.

```

883 \cs_new_protected:Npn \_cmd_add_expandable_type_E:w #1#2
884 {
885   \_cmd_add_default_E:nn {#1} {#2}
886   \tl_clear:N \l\_cmd_tmpb_tl
887   \tl_map_function:nN {#1} \_cmd_add_expandable_type_E_aux:n
888   \_cmd_add_expandable_grabber:nn
889   { E \bool_if:NT \l\_cmd_long_bool { _long } }
890   {
891     { \exp_not:o \l\_cmd_tmpb_tl }
892     {
893       \prg_replicate:nn { \tl_count:n {#1} }
894       { { \c_novalue_tl } }
895     }
896   }
897   \_cmd_prepare_signature:N
898 }
899 \cs_new_protected:Npn \_cmd_add_expandable_type_E_aux:n #1
900 {
901   \_cmd_get_grabber:NN #1 \l\_cmd_tmpa_tl
902   \tl_put_right:Nx \l\_cmd_tmpb_tl
903   { \exp_not:o \l\_cmd_tmpa_tl \exp_not:N #1 }
904 }

```

(End definition for _cmd_add_expandable_type_E:w and _cmd_add_expandable_type_E_aux:n.)

_cmd_add_expandable_type_m:w

Unlike the standard case, when working expandably each argument is always grabbed separately.

```

905 \cs_new_protected:Npn \_cmd_add_expandable_type_m:w
906 {
907   \_cmd_add_default:
908   \_cmd_add_expandable_grabber:nn
909   { m \bool_if:NT \l\_cmd_long_bool { _long } } { }
910   \_cmd_prepare_signature:N
911 }

```

(End definition for _cmd_add_expandable_type_m:w.)

_cmd_add_expandable_type_R:w

The R-type is very similar to the D-type argument, and so the same internals are used.

```

912 \cs_new_protected:Npn \_cmd_add_expandable_type_R:w
913 { \_cmd_add_expandable_type_D_aux:NNNn R }

```

(End definition for _cmd_add_expandable_type_R:w.)

_cmd_add_expandable_type_t:w

An auxiliary delimited by #1 is built now. It will be used to test for the presence of that token.

```

914 \cs_new_protected:Npn \_cmd_add_expandable_type_t:w #1
915 {

```



```

916 \__cmd_add_default:
917 \__cmd_get_grabber:NN #1 \l__cmd_tmpa_tl
918 \__cmd_add_expandable_grabber:nn { t }
919 {
920     \exp_not:o \l__cmd_tmpa_tl
921     \exp_not:N #1
922 }
923 \__cmd_prepare_signature:N
924 }

```

(End definition for __cmd_add_expandable_type_t:w.)

__cmd_add_expandable_grabber:nn This is called for all arguments to place the right grabber in the signature.

```

925 \cs_new_protected:Npn \__cmd_add_expandable_grabber:nn #1#2
926 {
927     \tl_put_right:Nx \l__cmd_signature_tl
928     { \exp_not:c { __cmd_expandable_grab_ #1 :w } #2 }
929 }

```

(End definition for __cmd_add_expandable_grabber:nn.)

__cmd_get_grabber:NN Given a token #1, defines an expandable function delimited by that token and stores it in the token list #2. The function is named after the token, unless that function name is already taken by some other grabber (this can happen in the rare case where delimiters with different category codes are used in the same document): in that case use a global counter to get a unique name. Since the grabbers are not named after xparse commands they should not be used to get material from the input stream.

```

930 \cs_new_protected:Npn \__cmd_get_grabber:NN #1#2
931 {
932     \cs_set:Npn \__cmd_tmp:w ##1 #1 {##1}
933     \exp_args:Nc \__cmd_get_grabber_auxi:NN
934     { __cmd_grabber_ \token_to_str:N #1 :w } #2
935 }
936 \cs_new_protected:Npn \__cmd_get_grabber_auxi:NN #1#2
937 {
938     \cs_if_eq:NNTF \__cmd_tmp:w #1
939     { \tl_set:Nn #2 {#1} }
940     {
941         \cs_if_exist:NTF #1
942         {
943             \int_gincr:N \g__cmd_grabber_int
944             \exp_args:Nc \__cmd_get_grabber_auxi:NN
945             {
946                 __cmd_grabber_
947                 - \int_use:N \g__cmd_grabber_int :w
948             }
949             #2
950         }
951         { \__cmd_get_grabber_auxii:NN #1 #2 }
952     }
953 }
954 \cs_new_protected:Npn \__cmd_get_grabber_auxii:NN #1#2
955 {
956     \cs_set_eq:NN #1 \__cmd_tmp:w

```

```

957 \tl_set:Nn #2 {#1}
958 }

```

(End definition for `__cmd_get_grabber:NN`, `__cmd_get_grabber_auxi:NN`, and `__cmd_get_grabber_auxii:NN`.)

1.8 Grabbing arguments

All of the grabbers follow the same basic pattern. The initial function stores in `\l__cmd_signature_tl` the code to grab further arguments, defines (the function in) `\l__cmd_fn_tl` that will grab the argument, and calls it.

Defining `\l__cmd_fn_tl` means determining whether to use `\cs_set:Npn` or `\cs_set_nopar:Npn`, and for optional arguments whether to skip spaces. Once the argument is found, `\l__cmd_fn_tl` calls `__cmd_add_arg:n`, responsible for calling processors and grabbing further arguments.

This uses the well-tested code of D-type arguments, skipping the peeking step because the b-type argument is always present, and adding a cleanup stage at the end by hijacking the signature. The clean-up consists of properly dealing with `\l__cmd_args_tl` and also putting back the `\end` that served as an end-delimiter: this `\end` receives the environment name as its argument and is run normally. The D-type code stores the argument found (body of the environment) as a brace group in `\l__cmd_args_tl` and depending on the presence of a prefix `!` we trim spaces or not before adding this braced argument into the saved `\l__cmd_args_tl`. The strange `\begin_` control sequence is there for display purposes only: it has to look like `\begin` in the terminal but not to delimited arguments.

```

959 \cs_new_protected:Npn \__cmd_grab_b:w
960 { \__cmd_grab_b_aux:NNw \cs_set_protected_nopar:Npn \tl_trim_spaces:n }
961 \cs_new_protected:Npn \__cmd_grab_b_long:w
962 { \__cmd_grab_b_aux:NNw \cs_set_protected:Npn \tl_trim_spaces:n }
963 \cs_new_protected:Npn \__cmd_grab_b_obey_spaces:w
964 { \__cmd_grab_b_aux:NNw \cs_set_protected_nopar:Npn \exp_not:n }
965 \cs_new_protected:Npn \__cmd_grab_b_long_obey_spaces:w
966 { \__cmd_grab_b_aux:NNw \cs_set_protected:Npn \exp_not:n }
967 \cs_new_protected:Npn \__cmd_grab_b_aux:NNw #1#2#3 \__cmd_run_code:
968 {
969   \__cmd_grab_D_aux:NNnN \begin \end {#3} #1
970   \tl_put_left:Nn \l__cmd_signature_tl { \__cmd_grab_b_end:Nw #2 }
971   \tl_set_eq:NN \l__cmd_saved_args_tl \l__cmd_args_tl
972   \tl_clear:N \l__cmd_args_tl
973   \exp_args:Nc \l__cmd_fn_tl { begin ~ }
974 }
975 \cs_new_protected:Npn \__cmd_grab_b_end:Nw #1#2 \__cmd_run_code:
976 {
977   \tl_set:Nx \l__cmd_args_tl
978   {
979     \exp_not:V \l__cmd_saved_args_tl
980     { \exp_after:wN #1 \l__cmd_args_tl }
981   }
982   #2
983   \__cmd_run_code:
984   \end
985 }

```

(End definition for `_cmd_grab_b:w` and others.)

```

    \_cmd_grab_D:w The generic delimited argument grabber. The auxiliary function does a peek test before
    \_cmd_grab_D_long:w calling \_cmd_grab_D_call:Nw, so that the optional nature of the argument works as
    \_cmd_grab_D_obey_spaces:w expected.
    \_cmd_grab_D_long_obey_spaces:w
986 \cs_new_protected:Npn \_cmd_grab_D:w #1#2#3 \_cmd_run_code:
987 {
988   \_cmd_grab_D_aux:NNnNN #1 #2 {#3} \cs_set_protected_nopar:Npn
989     \_cmd_peek_nonspace_remove:NTF
990 }
991 \cs_new_protected:Npn \_cmd_grab_D_long:w #1#2#3 \_cmd_run_code:
992 {
993   \_cmd_grab_D_aux:NNnNN #1 #2 {#3} \cs_set_protected:Npn
994     \_cmd_peek_nonspace_remove:NTF
995 }
996 \cs_new_protected:Npn \_cmd_grab_D_obey_spaces:w #1#2#3 \_cmd_run_code:
997 {
998   \_cmd_grab_D_aux:NNnNN #1 #2 {#3} \cs_set_protected_nopar:Npn
999     \_cmd_peek_meaning_remove:NTF
1000 }
1001 \cs_new_protected:Npn \_cmd_grab_D_long_obey_spaces:w #1#2#3 \_cmd_run_code:
1002 {
1003   \_cmd_grab_D_aux:NNnNN #1 #2 {#3} \cs_set_protected:Npn
1004     \_cmd_peek_meaning_remove:NTF
1005 }

```

This is a bit complicated. The idea is that, in order to check for nested optional argument tokens (`[[...]]` and so on) the argument needs to be grabbed without removing any braces at all. If this is not done, then cases like `[{[]}]` fail. So after testing for an optional argument, it is collected piece-wise. Inserting a quark prevents loss of braces, and there is then a test to see if there are nested delimiters to handle.

```

\_cmd_grab_D_aux:NNnNN
\_cmd_grab_D_aux:NNnN
1006 \cs_new_protected:Npn \_cmd_grab_D_aux:NNnNN #1#2#3#4#5
1007 {
1008   \_cmd_grab_D_aux:NNnN #1#2 {#3} #4
1009   #5 #1
1010   { \_cmd_grab_D_call:Nw #1 }
1011   { \_cmd_add_arg:o \c_novalue_tl }
1012 }

```

Inside the “standard” grabber, there is a test to see if the grabbed argument is entirely enclosed by braces. There are a couple of extra factors to allow for: the argument might be entirely empty, and spaces at the start and end of the input must be retained around a brace group. Also notice that a *blank* argument might still contain spaces.

```

1013 \cs_new_protected:Npn \_cmd_grab_D_aux:NNnN #1#2#3#4
1014 {
1015   \tl_set:Nn \l_cmd_signature_tl {#3}
1016   \exp_after:wN #4 \l_cmd_fn_tl ##1 #2
1017   {
1018     \tl_if_in:nnTF {##1} {#1}
1019     { \_cmd_grab_D_nested:NNnN #1 #2 {##1} #4 }
1020     {
1021       \tl_if_blank:oTF { \use_none:n ##1 }
1022       { \_cmd_add_arg:o { \use_none:n ##1 } }

```

```

1023         {
1024             \str_if_eq:eeTF
1025             { \exp_not:o { \use_none:n ##1 } }
1026             { { \exp_not:o { \use_ii:nnn ##1 \q_nil } } }
1027             { \__cmd_add_arg:o { \use_ii:nn ##1 } }
1028             { \__cmd_add_arg:o { \use_none:n ##1 } }
1029         }
1030     }
1031 }
1032 }

```

(End definition for `__cmd_grab_D:w` and others.)

```

\__cmd_grab_D_nested:NnnN
\__cmd_grab_D_nested:w
\l__cmd_nesting_a_tl
\l__cmd_nesting_b_tl
\q__cmd

```

Catching nested optional arguments means more work. The aim here is to collect up each pair of optional tokens without \TeX helping out, and without counting anything. The code above will already have removed the leading opening token and a closing token, but the wrong one. The aim is then to work through the material grabbed so far and divide it up on each opening token, grabbing a closing token to match (thus working in pairs). Once there are no opening tokens, then there is a second check to see if there are any opening tokens in the second part of the argument (for things like `[] []`). Once everything has been found, the entire collected material is added to the output as a single argument. The only tricky part here is ensuring that any grabbing function that might run away is named after the function currently being parsed and not after `xparse`. That leads to some rather complex nesting! There is also a need to prevent the loss of any braces, hence the insertion and removal of quarks along the way.

```

1033 \tl_new:N \l__cmd_nesting_a_tl
1034 \tl_new:N \l__cmd_nesting_b_tl
1035 \quark_new:N \q__cmd
1036 \cs_new_protected:Npn \__cmd_grab_D_nested:NnnN #1#2#3#4
1037 {
1038     \tl_clear:N \l__cmd_nesting_a_tl
1039     \tl_clear:N \l__cmd_nesting_b_tl
1040     \exp_after:wN #4 \l__cmd_fn_tl ##1 #1 ##2 \q__cmd ##3 #2
1041     {
1042         \tl_put_right:No \l__cmd_nesting_a_tl { \use_none:n ##1 #1 }
1043         \tl_put_right:No \l__cmd_nesting_b_tl { \use_i:nn #2 ##3 }
1044         \tl_if_in:nnTF {##2} {#1}
1045         {
1046             \l__cmd_fn_tl
1047             \q_nil ##2 \q__cmd \ERROR
1048         }
1049         {
1050             \tl_put_right:Nx \l__cmd_nesting_a_tl
1051             { \__cmd_grab_D_nested:w \q_nil ##2 \q_stop }
1052             \tl_if_in:NnTF \l__cmd_nesting_b_tl {#1}
1053             {
1054                 \tl_set_eq:NN \l__cmd_tmpa_tl \l__cmd_nesting_b_tl
1055                 \tl_clear:N \l__cmd_nesting_b_tl
1056                 \exp_after:wN \l__cmd_fn_tl \exp_after:wN
1057                 \q_nil \l__cmd_tmpa_tl \q_nil \q__cmd \ERROR
1058             }
1059             {
1060                 \tl_put_right:No \l__cmd_nesting_a_tl

```

```

1061         \l__cmd_nesting_b_tl
1062         \__cmd_add_arg:V \l__cmd_nesting_a_tl
1063     }
1064 }
1065 }
1066 \l__cmd_fn_tl #3 \q_nil \q__cmd \ERROR
1067 }
1068 \cs_new:Npn \__cmd_grab_D_nested:w #1 \q_nil \q_stop
1069 { \exp_not:o { \use_none:n #1 } }

```

(End definition for __cmd_grab_D_nested:NnN and others.)

__cmd_grab_D_call:Nw For D and R-type arguments, to avoid losing any braces, a token needs to be inserted before the argument to be grabbed. If the argument runs away because the closing token is missing then this inserted token shows up in the terminal. Ideally, #1 would therefore be used directly, but that is no good as it will mess up the rest of the grabber. Instead, a copy of #1 with an altered category code is used, as this will look right in the terminal but will not mess up the grabber. The only issue then is that the category code of #1 is unknown. So there is a quick test to ensure that the inserted token can never be matched by the grabber. (This assumes that the open and close delimiters are not the same character with different category codes, but that really should not happen in any sensible document-level syntax.) An exception is when #1 is a control sequence token, in which case the character-token treatment is no good because if hit with \token_to_str:N it would add sputios tokens to the argument. In this case a different branch is taken. The token inserted is then the same *<cname>* as #1, but with a space appended, so that the grabber don't see it as another of the same delimiter.

```

1070 \cs_new_protected_nopar:Npn \__cmd_grab_D_call:Nw #1
1071 {
1072     \token_if_eq_catcode:NNTF + #1
1073     {
1074         \exp_after:wN \exp_after:wN \exp_after:wN
1075         \l__cmd_fn_tl \char_generate:nn { '#1 } { 11 }
1076     }
1077     {
1078         \__cmd_token_if_cs:NNTF #1
1079         {
1080             \exp_after:wN \l__cmd_fn_tl
1081             \cs:w \cs_to_str:N #1 ~ \cs_end:
1082         }
1083         {
1084             \exp_after:wN \l__cmd_fn_tl
1085             \token_to_str:N #1
1086         }
1087     }
1088 }

```

(End definition for __cmd_grab_D_call:Nw.)

__cmd_grab_E:w Everything here needs to point to a loop.

```

\__cmd_grab_E_long:w
\__cmd_grab_E_obey_spaces:w
\__cmd_grab_E_long_obey_spaces:w
\__cmd_grab_E:nnN
\__cmd_grab_E_loop:NnN
\__cmd_grab_E_finalise:
1089 \cs_new_protected:Npn \__cmd_grab_E:w #1#2 \__cmd_run_code:
1090 {
1091     \__cmd_grab_E:nnNN {#1} {#2}
1092     \cs_set_protected_nopar:Npn

```

```

1093     \_cmd_peek_nonspace_remove:NTF
1094   }
1095 \cs_new_protected:Npn \__cmd_grab_E_long:w #1#2 \__cmd_run_code:
1096 {
1097   \_cmd_grab_E:nnNN {#1} {#2}
1098   \cs_set_protected:Npn
1099     \_cmd_peek_nonspace_remove:NTF
1100   }
1101 \cs_new_protected:Npn \__cmd_grab_E_obey_spaces:w #1#2 \__cmd_run_code:
1102 {
1103   \_cmd_grab_E:nnNN {#1} {#2}
1104   \cs_set_protected_nopar:Npn
1105     \_cmd_peek_meaning_remove:NTF
1106   }
1107 \cs_new_protected:Npn \__cmd_grab_E_long_obey_spaces:w #1#2 \__cmd_run_code:
1108 {
1109   \_cmd_grab_E:nnNN {#1} {#2}
1110   \cs_set_protected:Npn
1111     \_cmd_peek_meaning_remove:NTF
1112   }

```

A loop is needed here to allow a random ordering of keys. These are searched for one at a time, with any not found needing to be tracked: they can appear later. The grabbed values are held in a property list which is then turned into an ordered list to be passed back to the user.

```

1113 \cs_new_protected:Npn \__cmd_grab_E:nnNN #1#2#3#4
1114 {
1115   \exp_after:wN #3 \l__cmd_fn_tl ##1##2##3
1116   {
1117     \prop_put:Nnn \l__cmd_tmp_prop {##1} {##3}
1118     \__cmd_grab_E_loop:NnN #4 { } ##2 \q_recursion_stop
1119   }
1120   \prop_clear:N \l__cmd_tmp_prop
1121   \tl_set:Nn \l__cmd_signature_tl {#2}
1122   \cs_set_protected:Npn \__cmd_grab_E_finalise:
1123   {
1124     \tl_map_inline:nn {#1}
1125     {
1126       \prop_get:NnNF \l__cmd_tmp_prop {####1} \l__cmd_tmptb_tl
1127       { \tl_set_eq:NN \l__cmd_tmptb_tl \c_novalue_tl }
1128       \tl_put_right:Nx \l__cmd_args_tl
1129         { { \exp_not:V \l__cmd_tmptb_tl } }
1130     }
1131     \l__cmd_signature_tl \__cmd_run_code:
1132   }
1133   \__cmd_grab_E_loop:NnN #4 { } #1 \q_recursion_tail \q_recursion_stop
1134 }
1135 \cs_new_protected:Npn \__cmd_grab_E_loop:NnN #1#2#3#4 \q_recursion_stop
1136 {
1137   \cs_if_eq:NNTF #3 \q_recursion_tail
1138   { \__cmd_grab_E_finalise: }
1139   {
1140     #1 #3
1141     { \l__cmd_fn_tl #3 {#2#4} }

```

```

1142         { \_cmd_grab_E_loop:NnN #1 {#2#3} #4 \q_recursion_stop }
1143     }
1144 }
1145 \cs_new_protected:Npn \_cmd_grab_E_finalise: { }

```

(End definition for _cmd_grab_E:w and others.)

```

\_cmd_grab_m:w Collecting a single mandatory argument is quite easy.
\_cmd_grab_m_long:w
1146 \cs_new_protected:Npn \_cmd_grab_m:w #1 \_cmd_run_code:
1147 {
1148     \tl_set:Nn \l__cmd_signature_tl {#1}
1149     \exp_after:wN \cs_set_protected_nopar:Npn \l__cmd_fn_tl ##1
1150     { \_cmd_add_arg:n {##1} }
1151     \l__cmd_fn_tl
1152 }
1153 \cs_new_protected:Npn \_cmd_grab_m_long:w #1 \_cmd_run_code:
1154 {
1155     \tl_set:Nn \l__cmd_signature_tl {#1}
1156     \exp_after:wN \cs_set_protected:Npn \l__cmd_fn_tl ##1
1157     { \_cmd_add_arg:n {##1} }
1158     \l__cmd_fn_tl
1159 }

```

(End definition for _cmd_grab_m:w and _cmd_grab_m_long:w.)

_cmd_grab_m_1:w Grabbing 1–8 mandatory arguments is done by giving 8–1 known arguments to a 9-argument function that stores them in \l__cmd_args_tl. For simplicity, grabbing 9 mandatory arguments is done by grabbing 5 then 4 arguments.

```

\_cmd_grab_m_2:w
\_cmd_grab_m_3:w
\_cmd_grab_m_4:w 1160 \cs_new_protected_nopar:Npn \_cmd_grab_m_aux:Nnnnnnnnn #1#2#3#4#5#6#7#8#9
\_cmd_grab_m_5:w 1161 {
\_cmd_grab_m_6:w 1162     \tl_put_right:No \l__cmd_args_tl
\_cmd_grab_m_7:w 1163     { #1 {#2} {#3} {#4} {#5} {#6} {#7} {#8} {#9} }
\_cmd_grab_m_8:w 1164     \l__cmd_signature_tl \_cmd_run_code:
\_cmd_grab_m_9:w 1165 }
\_cmd_grab_m_aux:Nnnnnnnnn 1166 \cs_new_protected:cpn { \_cmd_grab_m_1:w } #1 \_cmd_run_code:
1167 {
1168     \tl_set:Nn \l__cmd_signature_tl {#1}
1169     \exp_after:wN \cs_set_eq:NN \l__cmd_fn_tl \_cmd_grab_m_aux:Nnnnnnnnn
1170     \l__cmd_fn_tl \use_none:nnnnnn { } { } { } { } { } { } { } { }
1171 }
1172 \cs_new_protected:cpn { \_cmd_grab_m_2:w } #1 \_cmd_run_code:
1173 {
1174     \tl_set:Nn \l__cmd_signature_tl {#1}
1175     \exp_after:wN \cs_set_eq:NN \l__cmd_fn_tl \_cmd_grab_m_aux:Nnnnnnnnn
1176     \l__cmd_fn_tl \use_none:nnnnnn { } { } { } { } { } { } { } { }
1177 }
1178 \cs_new_protected:cpn { \_cmd_grab_m_3:w } #1 \_cmd_run_code:
1179 {
1180     \tl_set:Nn \l__cmd_signature_tl {#1}
1181     \exp_after:wN \cs_set_eq:NN \l__cmd_fn_tl \_cmd_grab_m_aux:Nnnnnnnnn
1182     \l__cmd_fn_tl \use_none:nnnnnn { } { } { } { } { } { } { } { }
1183 }
1184 \cs_new_protected:cpn { \_cmd_grab_m_4:w } #1 \_cmd_run_code:
1185 {

```

```

1186 \tl_set:Nn \l__cmd_signature_tl {#1}
1187 \exp_after:wN \cs_set_eq:NN \l__cmd_fn_tl \__cmd_grab_m_aux:Nnnnnnnnn
1188 \l__cmd_fn_tl \use_none:nnnn { } { } { } { }
1189 }
1190 \cs_new_protected:cpn { __cmd_grab_m_5:w } #1 \__cmd_run_code:
1191 {
1192 \tl_set:Nn \l__cmd_signature_tl {#1}
1193 \exp_after:wN \cs_set_eq:NN \l__cmd_fn_tl \__cmd_grab_m_aux:Nnnnnnnnn
1194 \l__cmd_fn_tl \use_none:nnn { } { } { }
1195 }
1196 \cs_new_protected:cpn { __cmd_grab_m_6:w } #1 \__cmd_run_code:
1197 {
1198 \tl_set:Nn \l__cmd_signature_tl {#1}
1199 \exp_after:wN \cs_set_eq:NN \l__cmd_fn_tl \__cmd_grab_m_aux:Nnnnnnnnn
1200 \l__cmd_fn_tl \use_none:nn { } { }
1201 }
1202 \cs_new_protected:cpn { __cmd_grab_m_7:w } #1 \__cmd_run_code:
1203 {
1204 \tl_set:Nn \l__cmd_signature_tl {#1}
1205 \exp_after:wN \cs_set_eq:NN \l__cmd_fn_tl \__cmd_grab_m_aux:Nnnnnnnnn
1206 \l__cmd_fn_tl \use_none:n { }
1207 }
1208 \cs_new_protected:cpn { __cmd_grab_m_8:w } #1 \__cmd_run_code:
1209 {
1210 \tl_set:Nn \l__cmd_signature_tl {#1}
1211 \exp_after:wN \cs_set_eq:NN \l__cmd_fn_tl \__cmd_grab_m_aux:Nnnnnnnnn
1212 \l__cmd_fn_tl \prg_do_nothing:
1213 }
1214 \cs_new_protected:cpx { __cmd_grab_m_9:w }
1215 {
1216 \exp_not:c { __cmd_grab_m_5:w }
1217 \exp_not:c { __cmd_grab_m_4:w }
1218 }

```

(End definition for __cmd_grab_m_1:w and others.)

__cmd_grab_R:w The grabber for R-type arguments is basically the same as that for D-type ones, but
 __cmd_grab_R_long:w always skips spaces (as it is mandatory) and has a hard-coded error message.
 __cmd_grab_R_aux:NNnN

```

1219 \cs_new_protected:Npn \__cmd_grab_R:w #1#2#3 \__cmd_run_code:
1220 { \__cmd_grab_R_aux:NNnN #1 #2 {#3} \cs_set_protected_nopar:Npn }
1221 \cs_new_protected:Npn \__cmd_grab_R_long:w #1#2#3 \__cmd_run_code:
1222 { \__cmd_grab_R_aux:NNnN #1 #2 {#3} \cs_set_protected:Npn }
1223 \cs_new_protected:Npn \__cmd_grab_R_aux:NNnN #1#2#3#4
1224 {
1225 \__cmd_grab_D_aux:NNnN #1 #2 {#3} #4
1226 \__cmd_peek_nonspace_remove:NTF #1
1227 { \__cmd_grab_D_call:Nw #1 }
1228 {
1229 \msg_error:nnxx { cmd } { missing-required }
1230 { \__cmd_environment_or_command: }
1231 { \token_to_str:N #1 }
1232 \__cmd_add_arg:o \c_novalue_tl
1233 }
1234 }

```


(End definition for `__cmd_grab_R:w`, `__cmd_grab_R_long:w`, and `__cmd_grab_R_aux:NNnN`.)

`__cmd_grab_t:w` Dealing with a token is quite easy. Check the match, remove the token if needed and add
`__cmd_grab_t_obey_spaces:w` a flag to the output.
`__cmd_grab_t_aux:NNw`

```

1235 \cs_new_protected:Npn \__cmd_grab_t:w
1236 { \__cmd_grab_t_aux:NNw \__cmd_peek_nonspace_remove:NTF }
1237 \cs_new_protected:Npn \__cmd_grab_t_obey_spaces:w
1238 { \__cmd_grab_t_aux:NNw \__cmd_peek_meaning_remove:NTF }
1239 \cs_new_protected:Npn \__cmd_grab_t_aux:NNw #1#2#3 \__cmd_run_code:
1240 {
1241   \tl_set:Nn \l__cmd_signature_tl {#3}
1242   \exp_after:wN \cs_set_protected:Npn \l__cmd_fn_tl
1243   {
1244     #1 #2
1245     { \__cmd_add_arg:n { \BooleanTrue } }
1246     { \__cmd_add_arg:n { \BooleanFalse } }
1247   }
1248   \l__cmd_fn_tl
1249 }

```

(End definition for `__cmd_grab_t:w`, `__cmd_grab_t_obey_spaces:w`, and `__cmd_grab_t_aux:NNw`.)

`\l__cmd_v_arg_tl`

```

1250 \tl_new:N \l__cmd_v_arg_tl

```

`__cmd_grab_v:w` The opening delimiter is the first non-space token, and is never read verbatim. This is
`__cmd_grab_v_long:w` required by consistency with the case where the preceding argument was optional and
`__cmd_grab_v_aux:w` absent: then T_EX has already read and tokenized that token when looking for the optional
`__cmd_grab_v_group_end:` argument. The first thing is thus to check is that this delimiter is a character, and to
distinguish the case of a left brace (in that case, `\group_align_safe_end:` is needed
to compensate for the begin-group character that was just seen). Then set verbatim
catcodes with `__cmd_grab_v_aux_catcodes:`.

The group keep catcode changes local, and `\group_align_safe_begin/end:` allow
to use a character with category code 4 (normally &) as the delimiter (all commands
do `\group_align_safe_begin/end:`, so there's no need to do that again here). It is
ended by `__cmd_grab_v_group_end:`, which smuggles the collected argument out of
the group.

```

1251 \cs_new_protected:Npn \__cmd_grab_v:w
1252 {
1253   \bool_set_false:N \l__cmd_long_bool
1254   \__cmd_grab_v_aux:w
1255 }
1256 \cs_new_protected:Npn \__cmd_grab_v_long:w
1257 {
1258   \bool_set_true:N \l__cmd_long_bool
1259   \__cmd_grab_v_aux:w
1260 }
1261 \cs_new_protected:Npn \__cmd_grab_v_aux:w #1 \__cmd_run_code:
1262 {
1263   \tl_set:Nn \l__cmd_signature_tl {#1}
1264   \group_begin:
1265   \tex_escapechar:D = 92 \scan_stop:

```

```

1266 \tl_clear:N \l__cmd_v_arg_tl
1267 \peek_remove_spaces:n
1268 {
1269   \peek_meaning_remove:NTF \c_group_begin_token
1270   {
1271     \group_align_safe_end:
1272     \__cmd_grab_v_bgroup:
1273   }
1274   {
1275     \peek_N_type:TF
1276     { \__cmd_grab_v_aux_test:N }
1277     { \__cmd_grab_v_aux_abort:n { } }
1278   }
1279 }
1280 }
1281 \cs_new_protected:Npn \__cmd_grab_v_group_end:
1282 {
1283   \exp_args:NNNo
1284   \group_end:
1285   \tl_set:Nn \l__cmd_v_arg_tl { \l__cmd_v_arg_tl }
1286 }

```

(End definition for __cmd_grab_v:w and others.)

```

\__cmd_grab_v_aux_test:N
\__cmd_grab_v_aux_loop:N
\__cmd_grab_v_aux_loop:NN
\__cmd_grab_v_aux_loop_end:

```

Check that the opening delimiter is a character, setup category codes, then start reading tokens one by one, keeping the delimiter as an argument. If the verbatim was not nested, we will be grabbing one character at each step. Unfortunately, it can happen that what follows the verbatim argument is already tokenized. Thus, we check at each step that the next token is indeed a “nice” character, *i.e.*, is not a character with category code 1 (begin-group), 2 (end-group) or 6 (macro parameter), nor the space character, with category code 10 and character code 32, nor a control sequence. The partially built argument is stored in \l__cmd_v_arg_tl. If we ever meet a token which we cannot grab (non-N-type), or which is not a character according to __cmd_grab_v_token_if_char:NTF, then we bail out with __cmd_grab_v_aux_abort:n. Otherwise, we stop at the first character matching the delimiter.

```

1287 \cs_new_protected:Npn \__cmd_grab_v_aux_test:N #1
1288 {
1289   \__cmd_grab_v_token_if_char:NTF #1
1290   {
1291     \__cmd_grab_v_aux_put:N #1
1292     \__cmd_grab_v_aux_catcodes:
1293     \__cmd_grab_v_aux_loop:N #1
1294   }
1295   { \__cmd_grab_v_aux_abort:n {#1} #1 }
1296 }
1297 \cs_new_protected:Npn \__cmd_grab_v_aux_loop:N #1
1298 {
1299   \peek_N_type:TF
1300   { \__cmd_grab_v_aux_loop:NN #1 }
1301   { \__cmd_grab_v_aux_abort:n { } }
1302 }
1303 \cs_new_protected:Npn \__cmd_grab_v_aux_loop:NN #1#2
1304 {

```

```

1305 \__cmd_grab_v_token_if_char:NNTF #2
1306 {
1307     \token_if_eq_charcode:NNTF #1 #2
1308     { \__cmd_grab_v_aux_loop_end: }
1309     {
1310         \__cmd_grab_v_aux_put:N #2
1311         \__cmd_grab_v_aux_loop:N #1
1312     }
1313 }
1314 { \__cmd_grab_v_aux_abort:n {#2} #2 }
1315 }
1316 \cs_new_protected:Npn \__cmd_grab_v_aux_loop_end:
1317 {
1318     \__cmd_grab_v_group_end:
1319     \__cmd_add_arg:x { \tl_tail:N \l__cmd_v_arg_tl }
1320 }

```

(End definition for __cmd_grab_v_aux_test:N and others.)

\l__cmd_v_nesting_int

```

1321 \int_new:N \l__cmd_v_nesting_int

```

```

\__cmd_grab_v_bgroup:
\__cmd_grab_v_bgroup_loop:
\__cmd_grab_v_bgroup_loop:N

```

If the opening delimiter is a left brace, we keep track of how many left and right braces were encountered so far in \l__cmd_v_nesting_int (the methods used for optional arguments cannot apply here), and stop as soon as it reaches 0.

Some care was needed when removing the opening delimiter, which has already been assigned category code 1: using \peek_meaning_remove:NNTF in the __cmd_grab_v_aux:w function would break within alignments. Instead, we first convert that token to a string, and remove the result as a normal undelimited argument.

```

1322 \cs_new_protected:Npx \__cmd_grab_v_bgroup:
1323 {
1324     \exp_not:N \__cmd_grab_v_aux_catcodes:
1325     \exp_not:n { \int_set:Nn \l__cmd_v_nesting_int { 1 } }
1326     \exp_not:N \__cmd_grab_v_aux_put:N \iow_char:N \{
1327     \exp_not:N \__cmd_grab_v_bgroup_loop:
1328 }
1329 \cs_new_protected:Npn \__cmd_grab_v_bgroup_loop:
1330 {
1331     \peek_N_type:TF
1332     { \__cmd_grab_v_bgroup_loop:N }
1333     { \__cmd_grab_v_aux_abort:n { } }
1334 }
1335 \cs_new_protected:Npn \__cmd_grab_v_bgroup_loop:N #1
1336 {
1337     \__cmd_grab_v_token_if_char:NNTF #1
1338     {
1339         \token_if_eq_charcode:NNTF \c_group_end_token #1
1340         {
1341             \int_decr:N \l__cmd_v_nesting_int
1342             \int_compare:nNnTF \l__cmd_v_nesting_int > 0
1343             {
1344                 \__cmd_grab_v_aux_put:N #1
1345                 \__cmd_grab_v_bgroup_loop:

```

```

1346         }
1347         { \_cmd_grab_v_aux_loop_end: }
1348     }
1349     {
1350         \token_if_eq_charcode:NNT \c_group_begin_token #1
1351         { \int_incr:N \l__cmd_v_nesting_int }
1352         \_cmd_grab_v_aux_put:N #1
1353         \_cmd_grab_v_bgroup_loop:
1354     }
1355 }
1356 { \_cmd_grab_v_aux_abort:n {#1} #1 }
1357 }

```

(End definition for _cmd_grab_v_bgroup:, _cmd_grab_v_bgroup_loop:, and
_cmd_grab_v_bgroup_loop:N.)

_cmd_grab_v_aux_catcodes: The approach for short verbatim arguments is to make the end-line character a macro
parameter character: this is forbidden by the rest of the code. Then the error branch
_cmd_grab_v_aux_abort:n can check what caused the bail out and give the appropriate error message.

```

1358 \cs_new_protected:Npn \_cmd_grab_v_aux_catcodes:
1359 {
1360     \cs_set_eq:NN \do \char_set_catcode_other:N
1361     \dospecials
1362     \tex_endlinechar:D = '\^M \scan_stop:
1363     \bool_if:NTF \l__cmd_long_bool
1364     { \char_set_catcode_other:n { \tex_endlinechar:D } }
1365     { \char_set_catcode_parameter:n { \tex_endlinechar:D } }
1366 }
1367 \cs_new_protected:Npn \_cmd_grab_v_aux_abort:n #1
1368 {
1369     \_cmd_grab_v_group_end:
1370     \exp_after:wN \exp_after:wN \exp_after:wN
1371     \peek_meaning_remove:NTF \char_generate:nn { \tex_endlinechar:D } { 6 }
1372     {
1373         \msg_error:nnxxx { cmd } { verbatim-newline }
1374         { \_cmd_environment_or_command: }
1375         { \tl_to_str:N \l__cmd_v_arg_tl }
1376         { \tl_to_str:n {#1} }
1377         \_cmd_add_arg:o \c_novalue_tl
1378     }
1379     {
1380         \msg_error:nnxxx { cmd } { verbatim-tokenized }
1381         { \_cmd_environment_or_command: }
1382         { \tl_to_str:N \l__cmd_v_arg_tl }
1383         { \tl_to_str:n {#1} }
1384         \_cmd_add_arg:o \c_novalue_tl
1385     }
1386 }

```

(End definition for _cmd_grab_v_aux_catcodes: and _cmd_grab_v_aux_abort:n.)

_cmd_grab_v_aux_put:N Storing one token in the collected argument. Most tokens are converted to category code
12, with the exception of active characters, and spaces (not sure what should be done for
those).

```

1387 \cs_new_protected:Npn \__cmd_grab_v_aux_put:N #1
1388 {
1389   \tl_put_right:Nx \l__cmd_v_arg_tl
1390   {
1391     \token_if_active:NTF #1
1392     { \exp_not:N #1 } { \token_to_str:N #1 }
1393   }
1394 }

```

(End definition for __cmd_grab_v_aux_put:N.)

__cmd_grab_v_token_if_char:N

This function assumes that the escape character is printable. Then the string representation of control sequences is at least two characters, and \str_tail:n only removes the escape character. Macro parameter characters are doubled by \tl_to_str:n, and will also yield a non-empty result, hence are not considered as characters.

```

1395 \cs_new_protected:Npn \__cmd_grab_v_token_if_char:N #1
1396 { \str_if_eq:eeTF { } { \str_tail:n {#1} } }

```

(End definition for __cmd_grab_v_token_if_char:N.)

__cmd_add_arg:n
__cmd_add_arg:V
__cmd_add_arg:o
__cmd_add_arg:x

When an argument is found it is stored, then further arguments are grabbed by calling \l__cmd_signature_tl.

```

1397 \cs_new_protected:Npn \__cmd_add_arg:n #1
1398 {
1399   \tl_put_right:Nn \l__cmd_args_tl { {#1} }
1400   \l__cmd_signature_tl \__cmd_run_code:
1401 }
1402 \cs_generate_variant:Nn \__cmd_add_arg:n { V , o , x }

```

(End definition for __cmd_add_arg:n.)

1.9 Grabbing arguments expandably

__cmd_expandable_grab_D:w
__cmd_expandable_grab_D:NNNwNNn
__cmd_expandable_grab_D:NNNwNNnnn
__cmd_expandable_grab_D:Nw
__cmd_expandable_grab_D:nnNNNwNN

The first step is to grab the first token or group. The generic grabbers \<function>_ and \<function>_ are just after \q__cmd, we go and find them (and use the long one).

```

1403 \cs_new:Npn \__cmd_expandable_grab_D:w #1 \q__cmd #2#3
1404 { #2 { \__cmd_expandable_grab_D:NNNwNNn #1 \q__cmd #2 #3 } }

```

We then wish to test whether #7, which we just grabbed, is exactly #2. A preliminary test is whether their string representations coincide, then expand the only grabber function we have, #1, once: the two strings below are equal if and only if #7 matches #2 exactly.² The preliminary test is needed as #7 could validly contain \par (because a later mandatory argument could be long) and our grabber may be short. If #7 does not match #2, then the optional argument is missing, we use the default -NoValue-, and put back the argument #7 in the input stream.

If it does match, then interesting things need to be done. We will grab the argument piece by piece, with the following pattern:

²It is obvious that if #7 matches #2 then the strings are equal. We must check the converse. The right-hand-side of \str_if_eq:onTF does not end with #3, implying that the grabber function took everything as its arguments. The first brace group can only be empty if #7 starts with #2, otherwise the brace group preceding #7 would not vanish. The third brace group is empty, thus the \q__cmd that was used by our grabber #1 must be the one that we inserted (not some token in #7), hence the second brace group contains the end of #7 followed by #2. Since this is #2 on the right-hand-side, and no brace can be lost there, #7 must contain nothing else than its leading #2.

```

    <grabber> {<tokens>}
    \q_nil {<piece 1>} <piece 2> \ERROR \q__cmd
    \q_nil <input stream>

```

The *<grabber>* will find an opening delimiter in *<piece 2>*, take the *\q__cmd* as a second delimiter, and find more material delimited by the closing delimiter in the *<input stream>*. We then move the part before the opening delimiter from *<piece 2>* to *<piece 1>*, and the material taken from the *<input stream>* to the *<piece 2>*. Thus, the argument moves gradually from the *<input stream>* to the *<piece 2>*, then to the *<piece 1>* when we have made sure to find all opening and closing delimiters. This two-step process ensures that nesting works: the number of opening delimiters minus closing delimiters in *<piece 1>* is always equal to the number of closing delimiters in *<piece 2>*. We stop grabbing arguments once the *<piece 2>* contains no opening delimiter any more, hence the balance is reached, and the final argument is *<piece 1>* *<piece 2>*. The indirection via *__cmd_tmp:w* allows to insert *-NoValue-* expanded.

```

1405 \cs_set_protected:Npn \__cmd_tmp:w #1
1406 {
1407   \cs_new:Npn \__cmd_expandable_grab_D:NNNwNNn ##1##2##3##4 \q__cmd ##5##6##7
1408   {
1409     \str_if_eq:nnTF {##2} {##7}
1410     {
1411       \str_if_eq:onTF
1412       { ##1 { } { } ##7 ##2 \q__cmd ##3 }
1413       { { } {##2} { } }
1414     }
1415     { \use_ii:nn }
1416     {
1417       ##1
1418       { \__cmd_expandable_grab_D:NNNwNNnnn ##1##2##3##4 \q__cmd ##5##6 }
1419       \q_nil { } ##2 \ERROR \q__cmd \ERROR
1420     }
1421     { ##4 {##1} \q__cmd ##5 ##6 {##7} }
1422   }
1423 }
1424 \exp_args:No \__cmd_tmp:w { \c_novalue_tl }

```

At this stage, #7 is *\q_nil {<piece 1>} <more for piece 1>*, and we want to concatenate all that, removing *\q_nil*, and keeping the opening delimiter #2. Simply use *\use_ii:nn*. Also, #8 is *<remainder of piece 2> \ERROR*, and #9 is *\ERROR <more for piece 2>*. We concatenate those, replacing the two *\ERROR* by the closing delimiter #3.

```

1425 \cs_new:Npn \__cmd_expandable_grab_D:NNNwNNnnn #1#2#3#4 \q__cmd #5#6#7#8#9
1426 {
1427   \exp_args:Nof \__cmd_expandable_grab_D:nnNNNwNN
1428   { \use_ii:nn #7 #2 }
1429   { \__cmd_expandable_grab_D:Nw #3 \exp_stop_f: #8 #9 }
1430   #1#2#3 #4 \q__cmd #5 #6
1431 }
1432 \cs_new:Npn \__cmd_expandable_grab_D:Nw #1#2 \ERROR \ERROR { #2 #1 }

```

Armed with our two new *<pieces>*, we are ready to loop. However, we must first see if *<piece 2>* (here #2) contains any opening delimiter #4. Again, we expand #3, this time removing its whole output with *\use_none:nnn*. The test is similar to *\tl_if_in:nnTF*. The token list is empty if and only if #2 does not contain the opening delimiter. In

that case, we are done, and put the argument (from which we remove a spurious pair of delimiters coming from how we started the loop). Otherwise, we go back to looping with `__cmd_expandable_grab_D:NNNwNNnnn`. The code to deal with brace stripping is much the same as for the non-expandable case.

```

1433 \cs_new:Npn \__cmd_expandable_grab_D:nnNNNwNN #1#2#3#4#5#6 \q__cmd #7#8
1434 {
1435   \exp_args:No \tl_if_empty:oTF
1436   { #3 { \use_none:nnn } #2 \q__cmd #5 #4 \q__cmd #5 }
1437   {
1438     \tl_if_blank:oTF { \use_none:nn #1#2 }
1439     { \__cmd_put_arg_expandable:ow { \use_none:nn #1#2 } }
1440     {
1441       \str_if_eq:eeTF
1442       { \exp_not:o { \use_none:nn #1#2 } }
1443       { { \exp_not:o { \use_iii:nnnn #1#2 \q_nil } } }
1444       { \__cmd_put_arg_expandable:ow { \use_iii:nnn #1#2 } }
1445       { \__cmd_put_arg_expandable:ow { \use_none:nn #1#2 } }
1446     }
1447     #6 \q__cmd #7 #8
1448   }
1449   {
1450     #3
1451     { \__cmd_expandable_grab_D:NNNwNNnnn #3#4#5#6 \q__cmd #7 #8 }
1452     \q_nil {#1} #2 \ERROR \q__cmd \ERROR
1453   }
1454 }

```

(End definition for `__cmd_expandable_grab_D:w` and others.)

```

\__cmd_expandable_grab_D_alt:w
\__cmd_expandable_grab_D_alt:NNwNNn
\__cmd_expandable_grab_D_alt:Nwn

```

When the delimiters are identical, nesting is not possible and a simplified approach is used. The test concept here is the same as for the case where the delimiters are different but there cannot be any nesting.

```

1455 \cs_new:Npn \__cmd_expandable_grab_D_alt:w #1 \q__cmd #2#3
1456 { #2 { \__cmd_expandable_grab_D_alt:NNwNNn #1 \q__cmd #2 #3 } }
1457 \cs_set_protected:Npn \__cmd_tmp:w #1
1458 {
1459   \cs_new:Npn \__cmd_expandable_grab_D_alt:NNwNNn ##1##2##3 \q__cmd ##4##5##6
1460   {
1461     \str_if_eq:nnTF {##6} {##2}
1462     {
1463       \str_if_eq:onTF
1464       { ##1 { } ##6 ##2 ##2 }
1465       { { } ##2 }
1466     }
1467     { \use_ii:nn }
1468     {
1469       ##1
1470       { \__cmd_expandable_grab_D_alt:NNwn ##4 ##5 ##3 \q__cmd }
1471       ##6 \ERROR
1472     }
1473     { ##3 {#1} \q__cmd ##4 ##5 {##6} }
1474   }
1475 }
1476 \exp_args:No \__cmd_tmp:w { \c_novalue_tl }

```

```

1477 \cs_new:Npn \__cmd_expandable_grab_D_alt:NNwn #1#2#3 \q__cmd #4
1478 {
1479   \tl_if_blank:oTF { \use_none:n #4 }
1480   { \__cmd_put_arg_expandable:ow { \use_none:n #4 } }
1481   {
1482     \str_if_eq:eeTF
1483     { \exp_not:o { \use_none:n #4 } }
1484     { { \exp_not:o { \use_ii:nnn #4 \q_nil } } }
1485     { \__cmd_put_arg_expandable:ow { \use_ii:nn #4 } }
1486     { \__cmd_put_arg_expandable:ow { \use_none:n #4 } }
1487   }
1488   #3 \q__cmd #1 #2
1489 }

```

(End definition for __cmd_expandable_grab_D_alt:w, __cmd_expandable_grab_D_alt:NNwNNn, and __cmd_expandable_grab_D_alt:Nwn.)

__cmd_expandable_grab_E:w
 __cmd_expandable_grab_E_long:w
 __cmd_expandable_grab_E_aux:w
 __cmd_expandable_grab_E_test:nnw
 __cmd_expandable_grab_E_loop:nnnNNw
 __cmd_expandable_grab_E_find:w
 __cmd_expandable_grab_E_find:nnw
 __cmd_expandable_grab_E_end:nnw

We keep track of long/short by placing the appropriate grabber as the third token after \q__cmd; it is eventually removed by the end:nnw auxiliary. The aux:w auxiliary will be called repeatedly with two arguments: the set of pairs $\langle parser \rangle \langle token \rangle$, and the set of arguments found so far (initially all {-NoValue-}). At each step, grab what follows in the input stream then call the loop:nnnNNw auxiliary to compare it with each possible embellishment in turn. This auxiliary's #1 is what was found in the input, #2 collects $\langle parser \rangle \langle token \rangle$ pairs that did not match, #3 collects the corresponding arguments found previously, #4 and #5 is the current pair, #6 is the remaining pairs, #7 is empty or two \q_nil, and #8 is the current argument. If none of the pairs matched (determined by \quark_if_nil:NTF) then call the end auxiliary to stop looking for embellishments, remembering to put what was grabbed in the input back where it belongs, and storing the arguments found just before \q__cmd. If the current argument #8 is not -NoValue- or if the input #1 does not match #5 (see t-type arguments below for a similar \str_if_eq:onTF test) then carry on the loop. Otherwise, we found a new embellishment: grab the corresponding argument in the input using the find:w auxiliary. To avoid losing braces around that auxiliary's argument we include a space, which will be eliminated in the next loop through embellishments.

```

1490 \cs_new:Npn \__cmd_expandable_grab_E:w #1 \q__cmd #2#3
1491 { \__cmd_expandable_grab_E_aux:w #1 \q__cmd #2 #3 #3 }
1492 \cs_new:Npn \__cmd_expandable_grab_E_long:w #1 \q__cmd #2#3
1493 { \__cmd_expandable_grab_E_aux:w #1 \q__cmd #2 #3 #2 }
1494 \cs_new:Npn \__cmd_expandable_grab_E_aux:w #1 \q__cmd #2#3#4
1495 { #2 { \__cmd_expandable_grab_E_test:nnw #1 \q__cmd #2 #3 #4 } }
1496 \cs_new:Npn \__cmd_expandable_grab_E_test:nnw #1#2#3 \q__cmd #4#5#6#7
1497 {
1498   \__cmd_expandable_grab_E_loop:nnnNNw {#7} { } { }
1499   #1 \q_nil \q_nil \q_nil \q_mark #2 \q_nil
1500   #3 \q__cmd #4 #5 #6
1501 }
1502 \cs_new:Npn \__cmd_expandable_grab_E_loop:nnnNNw
1503 #1#2#3#4#5#6 \q_nil #7 \q_mark #8
1504 {
1505   \quark_if_nil:NTF #4
1506   { \__cmd_expandable_grab_E_end:nnw {#1} {#3} }
1507   {
1508     \tl_if_novalue:NTF {#8}

```



```

1509 { \str_if_eq:onTF { #4 { } #1 #5 } {#5} }
1510 { \use_ii:nn }
1511 { \__cmd_expandable_grab_E_find:w { #2 #4 #5 #6 } {#3} ~ }
1512 {
1513   \__cmd_expandable_grab_E_loop:nnnNNw
1514   {#1} { #2 #4 #5 } { #3 {#8} }
1515   #6 \q_nil #7 \q_mark
1516 }
1517 }
1518 }
1519 \cs_new:Npn \__cmd_expandable_grab_E_find:w #1 \q__cmd #2#3#4
1520 { #4 { \__cmd_expandable_grab_E_find:nnw #1 \q__cmd #2 #3 #4 } }
1521 \cs_new:Npn \__cmd_expandable_grab_E_find:nnw #1#2#3 \q_nil #4 \q__cmd #5#6#7#8
1522 { \__cmd_expandable_grab_E_aux:w {#1} { #2 {#8} #3 } #4 \q__cmd #5 #6 #7 }
1523 \cs_new:Npn \__cmd_expandable_grab_E_end:nnw #1#2#3 \q__cmd #4#5#6
1524 { #3 #2 \q__cmd #4 #5 {#1} }

```

(End definition for __cmd_expandable_grab_E:w and others.)

__cmd_expandable_grab_m:w
 __cmd_expandable_grab_m_long:w
 __cmd_expandable_grab_m_aux:wNn

The mandatory case is easy: find the auxiliary after the \q__cmd, and use it directly to grab the argument, then correctly position the argument before \q__cmd.

```

1525 \cs_new:Npn \__cmd_expandable_grab_m:w #1 \q__cmd #2#3
1526 { #3 { \__cmd_expandable_grab_m_aux:wNn #1 \q__cmd #2 #3 } }
1527 \cs_new:Npn \__cmd_expandable_grab_m_long:w #1 \q__cmd #2#3
1528 { #2 { \__cmd_expandable_grab_m_aux:wNn #1 \q__cmd #2 #3 } }
1529 \cs_new:Npn \__cmd_expandable_grab_m_aux:wNn #1 \q__cmd #2#3#4
1530 { #1 {#4} \q__cmd #2 #3 }

```

(End definition for __cmd_expandable_grab_m:w, __cmd_expandable_grab_m_long:w, and __cmd_expandable_grab_m_aux:wNn.)

__cmd_expandable_grab_R:w
 __cmd_expandable_grab_R_aux:NNNwNn

Much the same as for the D-type argument, with only the lead-off function varying.

```

1531 \cs_new:Npn \__cmd_expandable_grab_R:w #1 \q__cmd #2#3
1532 { #2 { \__cmd_expandable_grab_R_aux:NNNwNn #1 \q__cmd #2#3 } }
1533 \cs_set_protected:Npn \__cmd_tmp:w #1
1534 {
1535   \cs_new:Npn \__cmd_expandable_grab_R_aux:NNNwNn ##1##2##3##4 \q__cmd ##5##6##7
1536   {
1537     \str_if_eq:nnTF {##7} {##2}
1538     {
1539       \str_if_eq:onTF
1540       { ##1 { } { } ##7 ##2 \q__cmd ##3 }
1541       { { } {##2} { } }
1542     }
1543     { \use_ii:nn }
1544     {
1545       ##1
1546       { \__cmd_expandable_grab_D:NNNwNNnnn ##1##2##3##4 \q__cmd ##5##6 }
1547       \q_nil { } ##2 \ERROR \q__cmd \ERROR
1548     }
1549     {
1550       \msg_expandable_error:nnff { cmd } { missing-required }
1551       { \exp_args:Nf \tl_trim_spaces:n { \token_to_str:N ##5 } }
1552       { \tl_to_str:n {##2} }
1553       ##4 {#1} \q__cmd ##5 ##6 {##7}

```

```

1554     }
1555   }
1556 }
1557 \exp_args:No \__cmd_tmp:w { \c_novalue_tl }

(End definition for \__cmd_expandable_grab_R:w and \__cmd_expandable_grab_R_aux:NNwNNn.)

\__cmd_expandable_grab_R_alt:w When the delimiters are identical, nesting is not possible and a simplified approach is
\__cmd_expandable_grab_R_alt_aux:NNwNNn used. The test concept here is the same as for the case where the delimiters are different.

1558 \cs_new:Npn \__cmd_expandable_grab_R_alt:w #1 \q__cmd #2#3
1559   { #2 { \__cmd_expandable_grab_R_alt_aux:NNwNNn #1 \q__cmd #2#3 } }
1560 \cs_set_protected:Npn \__cmd_tmp:w #1
1561   {
1562     \cs_new:Npn \__cmd_expandable_grab_R_alt_aux:NNwNNn ##1##2##3 \q__cmd ##4##5##6
1563     {
1564       \str_if_eq:nnTF {##6} {##2}
1565       {
1566         \str_if_eq:onTF
1567           { ##1 { } ##6 ##2 ##2 }
1568           { { } ##2 }
1569       }
1570       { \use_ii:nn }
1571       {
1572         ##1
1573         { \__cmd_expandable_grab_D_alt:NNwn ##4 ##5 ##3 \q__cmd }
1574         ##6 \ERROR
1575       }
1576       {
1577         \msg_expandable_error:nnff { cmd } { missing-required }
1578         { \exp_args:Nf \tl_trim_spaces:n { \token_to_str:N ##4 } }
1579         { \tl_to_str:n {##2} }
1580         ##3 {##1} \q__cmd ##4 ##5 {##6}
1581       }
1582     }
1583   }
1584 \exp_args:No \__cmd_tmp:w { \c_novalue_tl }

(End definition for \__cmd_expandable_grab_R_alt:w and
\__cmd_expandable_grab_R_alt_aux:NNwNNn.)

```

__cmd_expandable_grab_t:w As for a D-type argument, here we compare the grabbed tokens using the only parser we
__cmd_expandable_grab_t_aux:NNwn have in order to work out if #2 is exactly equal to the output of the grabber.

```

1585 \cs_new:Npn \__cmd_expandable_grab_t:w #1 \q__cmd #2#3
1586   { #2 { \__cmd_expandable_grab_t_aux:NNwn #1 \q__cmd #2 #3 } }
1587 \cs_new:Npn \__cmd_expandable_grab_t_aux:NNwn #1#2#3 \q__cmd #4#5#6
1588   {
1589     \str_if_eq:onTF { #1 { } #6 #2 } {##2}
1590     { #3 { \BooleanTrue } \q__cmd #4 #5 }
1591     { #3 { \BooleanFalse } \q__cmd #4 #5 {##6} }
1592   }

(End definition for \__cmd_expandable_grab_t:w and \__cmd_expandable_grab_t_aux:NNwn.)

```

__cmd_put_arg_expandable:nw A useful helper, to store arguments when they are ready.
__cmd_put_arg_expandable:ow

```

1593 \cs_new:Npn \__cmd_put_arg_expandable:nw #1#2 \q__cmd { #2 {##1} \q__cmd }
1594 \cs_generate_variant:Nn \__cmd_put_arg_expandable:nw { o }

```

(End definition for _cmd_put_arg_expandable:nw.)

1.10 Argument processors

```
\_cmd_bool_reverse:N A simple reversal.  
1595 \cs_new_protected:Npn \_cmd_bool_reverse:N #1  
1596 {  
1597   \bool_if:NTF #1  
1598     { \tl_set:Nn \ProcessedArgument { \c_false_bool } }  
1599     { \tl_set:Nn \ProcessedArgument { \c_true_bool } }  
1600 }  
  
(End definition for \_cmd_bool_reverse:N.)
```

```

\l__cmd_split_list_seq
\l__cmd_split_list_tl
\__cmd_split_list_multi:nn
\__cmd_split_list_multi:nV
\__cmd_split_list_single:Nn

```

Splitting can take place either at a single token or at a longer identifier. To deal with single active tokens, a two-part procedure is needed.

```

1601 \seq_new:N \l__cmd_split_list_seq
1602 \tl_new:N \l__cmd_split_list_tl
1603 \cs_new_protected:Npn \__cmd_split_list:nn #1#2
1604 {
1605   \tl_if_single:nTF {#1}
1606   {
1607     \token_if_cs:NTF #1
1608     { \__cmd_split_list_multi:nn {#1} {#2} }
1609     { \__cmd_split_list_single:Nn #1 {#2} }
1610   }
1611   { \__cmd_split_list_multi:nn {#1} {#2} }
1612 }
1613 \cs_new_protected:Npn \__cmd_split_list_multi:nn #1#2
1614 {
1615   \seq_set_split:Nnn \l__cmd_split_list_seq {#1} {#2}
1616   \tl_clear:N \ProcessedArgument
1617   \seq_map_inline:Nn \l__cmd_split_list_seq
1618   { \tl_put_right:Nn \ProcessedArgument { {##1} } }
1619 }
1620 \cs_generate_variant:Nn \__cmd_split_list_multi:nn { nV }
1621 \group_begin:
1622 \char_set_catcode_active:N ^^@
1623 \cs_new_protected:Npn \__cmd_split_list_single:Nn #1#2
1624 {
1625   \tl_set:Nn \l__cmd_split_list_tl {#2}
1626   \group_begin:
1627   \char_set_lccode:nn { '^^@ } { '#1 }
1628   \tex_lowercase:D
1629   {
1630     \group_end:
1631     \tl_replace_all:Nnn \l__cmd_split_list_tl { ^^@ }
1632     } {#1}
1633   \__cmd_split_list_multi:nV {#1} \l__cmd_split_list_tl
1634 }
1635 \group_end:

```

(End definition for __cmd_split_list:nn, __cmd_split_list_multi:nn, and __cmd_split_list_single:Nn.)

```

\__cmd_split_argument:nnn
\__cmd_split_argument_aux:nnnn
\__cmd_split_argument_aux:nV
\__cmd_split_argument_aux:wn

```

Splitting to a known number of items is a special version of splitting a list, in which the limit is hard-coded and where there will always be exactly the correct number of output items. An auxiliary function is used to save on working out the token list length several times.

```

1636 \cs_new_protected:Npn \__cmd_split_argument:nnn #1#2#3
1637 {
1638   \__cmd_split_list:nn {#2} {#3}
1639   \exp_args:Nf \__cmd_split_argument_aux:nnnn
1640   { \tl_count:N \ProcessedArgument }
1641   {#1} {#2} {#3}
1642 }

```

```

1643 \cs_new_protected:Npn \__cmd_split_argument_aux:nnnn #1#2#3#4
1644 {
1645   \int_compare:nNnF {#1} = { #2 + 1 }
1646   {
1647     \int_compare:nNnTF {#1} > { #2 + 1 }
1648     {
1649       \tl_set:Nx \ProcessedArgument
1650       {
1651         \exp_last_unbraced:NnNo
1652         \__cmd_split_argument_aux:n
1653         { #2 + 1 }
1654         \use_none_delimit_by_q_stop:w
1655         \ProcessedArgument
1656         \q_stop
1657       }
1658       \msg_error:nnxxx { cmd } { split-excess-tokens }
1659       { \tl_to_str:n {#3} } { \int_eval:n { #2 + 1 } }
1660       { \tl_to_str:n {#4} }
1661     }
1662   }
1663   \tl_put_right:Nx \ProcessedArgument
1664   {
1665     \prg_replicate:nn { #2 + 1 - (#1) }
1666     { { \exp_not:V \c_novalue_tl } }
1667   }
1668 }
1669 }
1670 }

```

Auxiliaries to leave exactly the correct number of arguments in \ProcessedArgument.

```

1671 \cs_new:Npn \__cmd_split_argument_aux:n #1
1672 { \prg_replicate:nn {#1} { \__cmd_split_argument_aux:wn } }
1673 \cs_new:Npn \__cmd_split_argument_aux:wn #1 \use_none_delimit_by_q_stop:w #2
1674 {
1675   \exp_not:n { {#2} }
1676   #1
1677   \use_none_delimit_by_q_stop:w
1678 }

```

(End definition for __cmd_split_argument:nnn and others.)

__cmd_trim_spaces:n This one is almost trivial.

```

1679 \cs_new_protected:Npn \__cmd_trim_spaces:n #1
1680 { \tl_set:Nx \ProcessedArgument { \tl_trim_spaces:n {#1} } }

```

(End definition for __cmd_trim_spaces:n.)

1.11 Access to the argument specification

__cmd_get_arg_spec_error:N Provide an informative error when trying to get the argument specification of a non-xparse command or environment.

```

\__cmd_get_arg_spec_error:n
\__cmd_get_arg_spec_error_aux:n
1681 \cs_new_protected:Npn \__cmd_get_arg_spec_error:N #1
1682 {
1683   \bool_set_false:N \l__cmd_environment_bool

```

```

1684 \tl_set:Nn \l__cmd_fn_tl {#1}
1685 \__cmd_get_arg_spec_error_aux:n { \cs_if_exist:NTF #1 }
1686 }
1687 \cs_new_protected:Npn \__cmd_get_arg_spec_error:n #1
1688 {
1689   \bool_set_true:N \l__cmd_environment_bool
1690   \str_set:Nx \l__cmd_environment_str {#1}
1691   \__cmd_get_arg_spec_error_aux:n
1692     { \cs_if_exist:cTF { \l__cmd_environment_str } }
1693 }
1694 \cs_new_protected:Npn \__cmd_get_arg_spec_error_aux:n #1
1695 {
1696   #1
1697   {
1698     \msg_error:nnx { cmd } { non-xparse }
1699     { \__cmd_environment_or_command: }
1700   }
1701   {
1702     \msg_error:nnx { cmd } { unknown }
1703     { \__cmd_environment_or_command: }
1704   }
1705 }

```

(End definition for __cmd_get_arg_spec_error:N, __cmd_get_arg_spec_error:n, and
__cmd_get_arg_spec_error_aux:n.)

__cmd_get_arg_spec:NTF If the command is not an xparse command, complain. If it is, its second “item” is the argument specification.

```

1706 \cs_new_protected:Npn \__cmd_get_arg_spec:NTF #1#2#3
1707 {
1708   \__kernel_cmd_if_xparse:NTF #1
1709   {
1710     \tl_set:Nx \ArgumentSpecification { \tl_item:Nn #1 { 2 } }
1711     #2
1712   }
1713   {#3}
1714 }

```

(End definition for __cmd_get_arg_spec:NTF.)

Rolling forward from 2020-10-01 is tricky because the entire `ltxcmd` module is new, but the user-level commands have the same name, so only these will clash. To work around that, in `latexrelease` mode we will (temporarily) disable `__kernel_chk_if_free_cs:N` for this final part of the code, then restore at the end.

```

1715 <latexrelease>\cs_new_eq:NN \__cmd_chk_if_free_cs:N \__kernel_chk_if_free_cs:N
1716 <latexrelease>\cs_gset_eq:NN \__kernel_chk_if_free_cs:N \use_none:n

```

\ArgumentSpecification

```

1717 \tl_new:N \ArgumentSpecification

```

__cmd_get_arg_spec:N Recovering the argument specification is now trivial.

```

\__cmd_get_arg_spec:n
1718 \cs_new_protected:Npn \__cmd_get_arg_spec:N #1
1719 {
1720   \__cmd_get_arg_spec:NTF #1 { }

```

```

1721     { \_cmd_get_arg_spec_error:N #1 }
1722   }
1723 \cs_new_protected:Npn \_cmd_get_arg_spec:n #1
1724 {
1725   \exp_args:Nc \_cmd_get_arg_spec:NTF
1726     { environment~ \tl_to_str:n {#1} }
1727   { }
1728   { \_cmd_get_arg_spec_error:n {#1} }
1729 }

```

(End definition for _cmd_get_arg_spec:N and _cmd_get_arg_spec:n.)

_cmd_show_arg_spec:N Showing the argument specification simply means finding it and then calling the \tl_show:N function.

```

\_cmd_show_arg_spec:n
1730 \cs_new_protected:Npn \_cmd_show_arg_spec:N #1
1731 {
1732   \_cmd_get_arg_spec:NTF #1
1733     { \tl_show:N \ArgumentSpecification }
1734     { \_cmd_get_arg_spec_error:N #1 }
1735 }
1736 \cs_new_protected:Npn \_cmd_show_arg_spec:n #1
1737 {
1738   \exp_args:Nc \_cmd_get_arg_spec:NTF
1739     { environment~ \tl_to_str:n {#1} }
1740     { \tl_show:N \ArgumentSpecification }
1741     { \_cmd_get_arg_spec_error:n {#1} }
1742 }

```

(End definition for _cmd_show_arg_spec:N and _cmd_show_arg_spec:n.)

1.12 Utilities

_cmd_check_definable:nNT
_cmd_check_definable_aux:nN

Check that a token list is appropriate as a first argument of \NewDocumentCommand and similar functions and otherwise produce an error. First trim whitespace to allow for spaces around the actual command to be defined. If the result has multiple tokens, it is not a valid argument. The single token is a control sequence exactly if its string representation has more than one character (using \token_to_str:N rather than \tl_to_str:n to avoid problems with macro parameter characters, and setting \tex_escapechar:D to prevent it from being non-printable). Finally, check for an active character: this is done by lowercasing the token to fix its character code (arbitrarily to that of ?) and comparing the result to an active ?. Both control sequences and active characters are valid arguments, and non-active character tokens are not. In all cases, the group opened to keep assignments local must be closed.

```

1743 \cs_new_protected:Npn \_cmd_check_definable:nNT #1
1744 { \tl_trim_spaces_apply:nN {#1} \_cmd_check_definable_aux:nN }
1745 \group_begin:
1746   \char_set_catcode_active:n { '?' }
1747   \cs_new_protected:Npn \_cmd_check_definable_aux:nN #1#2
1748   {
1749     \group_begin:
1750     \tl_if_single_token:nTF {#1}
1751     {
1752       \int_set:Nn \tex_escapechar:D { 92 }

```

```

1753 \exp_args:Nx \tl_if_empty:nTF
1754 { \exp_args:No \str_tail:n { \token_to_str:N #1 } }
1755 {
1756   \exp_args:Nx \char_set_lccode:nn
1757   { ' \str_head:n {#1} } { ? }
1758   \tex_lowercase:D { \tl_if_eq:nnTF {#1} } { ? }
1759   { \group_end: \use_iii:nnn }
1760   { \group_end: \use_i:nnn }
1761 }
1762 { \group_end: \use_iii:nnn }
1763 }
1764 { \group_end: \use_ii:nnn }
1765 {
1766   \msg_error:nnxx { cmd } { not-definable }
1767   { \tl_to_str:n {#1} } { \token_to_str:N #2 }
1768 }
1769 {
1770   \msg_error:nnxx { cmd } { not-one-token }
1771   { \tl_to_str:n {#1} } { \token_to_str:N #2 }
1772 }
1773 }
1774 \group_end:

```

(End definition for `__cmd_check_definable:nNT` and `__cmd_check_definable_aux:nN`.)

`__cmd_token_if_cs:NNTF` Based on the definition of `__cmd_check_definable_aux:nN` above, but only checks for an actual control sequence (*i.e.*, `\<anything>`). `\tex_escapechar:D` is temporarily changed to a known value and then it checks if `\string#1` contains more than one character: if it does, it's a control sequence. This test differs from `\token_if_cs:NNTF` for example in `\token_if_cs:NNTF \c_group_begin_token {T}{F}`, where `\token_if_cs:NNTF` returns false.

```

1775 \cs_new_protected:Npn \__cmd_token_if_cs:NNTF #1
1776 {
1777   \group_begin:
1778   \int_set:Nn \tex_escapechar:D { 92 }
1779   \exp_args:Nx \tl_if_empty:nTF
1780   { \exp_args:No \str_tail:n { \token_to_str:N #1 } }
1781   { \group_end: \use_ii:nn }
1782   { \group_end: \use_i:nn }
1783 }

```

(End definition for `__cmd_token_if_cs:NNTF`.)

`__cmd_tl_mapthread_function:NNN` Analogue of `\seq_mapthread_function:NNN` for token lists.
`__cmd_tl_mapthread_function:nnN`
`__cmd_tl_mapthread_loop:w`

```

1784 \cs_new:Npn \__cmd_tl_mapthread_function:NNN #1#2#3
1785 {
1786   \exp_after:wN \exp_after:wN
1787   \exp_after:wN \__cmd_tl_mapthread_loop:w
1788   \exp_after:wN \exp_after:wN
1789   \exp_after:wN #3
1790   \exp_after:wN #1
1791   \exp_after:wN \q_recursion_tail
1792   \exp_after:wN \q_mark
1793   #2

```



```

1794 \q_recursion_tail
1795 \q_recursion_stop
1796 }
1797 \cs_new:Npn \__cmd_tl_mapthread_function:nnN #1#2#3
1798 {
1799   \__cmd_tl_mapthread_loop:w #3
1800   #1 \q_recursion_tail \q_mark
1801   #2 \q_recursion_tail \q_recursion_stop
1802 }
1803 \cs_new:Npn \__cmd_tl_mapthread_loop:w #1#2#3 \q_mark #4
1804 {
1805   \quark_if_recursion_tail_stop:n {#2}
1806   \quark_if_recursion_tail_stop:n {#4}
1807   #1 {#2} {#4}
1808   \__cmd_tl_mapthread_loop:w #1#3 \q_mark
1809 }

```

(End definition for `__cmd_tl_mapthread_function:NNN`, `__cmd_tl_mapthread_function:nnN`, and `__cmd_tl_mapthread_loop:w`.)

`__kernel_cmd_if_xparse:NTF` To determine whether the command is an xparse command check that its `arg_spec` is empty (this also excludes non-macros) and that its `replacement_spec` starts with either `__cmd_start:nNNnnn` (non-expandable command) or `__cmd_start_expandable:nNNNNn` (expandable command) or `__cmd_start_env:nnnnn` (environment).

This conditional is needed in several kernel modules and is therefore has a kernel-internal name.

```

1810 \cs_new_protected:Npn \__kernel_cmd_if_xparse:NTF #1
1811 {
1812   \exp_args:Nf \str_case_e:nnTF
1813   {
1814     \exp_args:Nf \tl_if_empty:nT { \cs_argument_spec:N #1 }
1815     {
1816       \exp_last_unbraced:Nf \__cmd_cmd_if_xparse_aux:w
1817       { \cs_replacement_spec:N #1 } ~ \q_stop
1818     }
1819   }
1820   {
1821     { \token_to_str:N \__cmd_start:nNNnnn } { }
1822     { \token_to_str:N \__cmd_start_expandable:nNNNNn } { }
1823     { \token_to_str:N \__cmd_start_env:nnnnn } { }
1824   }
1825 }
1826 \cs_new:Npn \__cmd_cmd_if_xparse_aux:w #1 ~ #2 \q_stop {#1}

```

(End definition for `__kernel_cmd_if_xparse:NTF` and `__cmd_cmd_if_xparse_aux:N`.)

`__cmd_peek_nonspace:NTF` Collect spaces in a loop, and put the collected spaces back in the false branch of a call to `\peek_meaning:NTF` or `\peek_meaning_remove:NTF`.

`__cmd_peek_nonspace_remove:NTF`
`__cmd_peek_nonspace_aux:nNNTF`

```

1827 \cs_new_protected:Npn \__cmd_peek_nonspace:NTF
1828 { \__cmd_peek_nonspace_aux:nNNTF { } \__cmd_peek_meaning:NTF }
1829 \cs_new_protected:Npn \__cmd_peek_nonspace_remove:NTF
1830 { \__cmd_peek_nonspace_aux:nNNTF { } \__cmd_peek_meaning_remove:NTF }
1831 \cs_new_protected:Npn \__cmd_peek_nonspace_aux:nNNTF #1#2#3#4#5
1832 {

```

```

1833 \peek_meaning_remove:NTF \c_space_token
1834 { \__cmd_peek_nonspace_aux:nNTF { #1 ~ } #2 #3 {#4} {#5} }
1835 { #2 #3 { #4 } { #5 #1 } }
1836 }

```

(End definition for __cmd_peek_nonspace:NTF, __cmd_peek_nonspace_remove:NTF, and __cmd_peek_nonspace_aux:nNTF.)

```

\__cmd_peek_meaning:NTF
\__cmd_peek_meaning_remove:NTF
\__cmd_peek_cs_check_equal:NNN
\__cmd_peek_meaning_aux:nNTF
\__cmd_peek_true_remove:NNw

```

Peek ahead for a token with a given meaning. In case the search token is a control sequence, also check that the *<csname>* is the same as the control sequence peeked at. This extra verification is necessary when the command is delimited by control sequence tokens (as opposed to character tokens), and we want the the exact same control sequence to match.

```

1837 \cs_new_protected:Npn \__cmd_peek_meaning:NTF
1838 { \__cmd_peek_meaning_aux:nNTF \c_false_bool }
1839 \cs_new_protected:Npn \__cmd_peek_meaning_remove:NTF
1840 { \__cmd_peek_meaning_aux:nNTF \c_true_bool }
1841 \cs_new_protected:Npn \__cmd_peek_meaning_aux:nNTF #1#2#3#4
1842 {
1843   \tl_set:Nn \l__cmd_tmppa_tl {#3}
1844   \tl_set:Nn \l__cmd_tmppb_tl {#4}
1845   \peek_meaning:NTF #2
1846   {
1847     \token_if_eq_meaning:nNTF #2 \c_group_begin_token
1848     { \__cmd_peek_true_remove:Nw #1 }
1849     {
1850       \__cmd_token_if_cs:NTF #2
1851       { \__cmd_peek_cs_check_equal:NNN #1 #2 }
1852       { \__cmd_peek_true_remove:Nw #1 }
1853     }
1854   }
1855   { \l__cmd_tmppb_tl }
1856 }
1857 \cs_new_protected:Npn \__cmd_peek_cs_check_equal:NNN #1#2#3
1858 {
1859   \str_if_eq:nnTF {#2} {#3}
1860   { \__cmd_peek_true_remove:Nw #1 }
1861   { \l__cmd_tmppb_tl }
1862   #3
1863 }
1864 \cs_new_protected:Npn \__cmd_peek_true_remove:Nw #1
1865 {
1866   \bool_if:NTF #1
1867   {
1868     \tex_afterassignment:D \l__cmd_tmppa_tl
1869     \cs_set_eq:NN \__cmd_tmp:w
1870   }
1871   { \l__cmd_tmppa_tl }
1872 }

```

(End definition for __cmd_peek_meaning:NTF and others.)

1.13 Messages

`\c__cmd_ignore_def_tl`

```
1873 \tl_const:Nn \c__cmd_ignore_def_tl
1874   { \\\ \ LaTeX-will-ignore-this-entire-definition. }
```

`_cmd_environment_or_command:` Two texts used in several messages.

```
1875 \cs_new:Npn \_cmd_environment_or_command:
1876   {
1877     \bool_if:NTF \l__cmd_environment_bool
1878       { environment ~ ' \l__cmd_environment_str ' }
1879     {
1880       command ~ '
1881       \exp_args:Nf \tl_trim_spaces:n
1882         { \exp_after:wN \token_to_str:N \l__cmd_fn_tl }
1883       ,
1884     }
1885   }
```

(End definition for `_cmd_environment_or_command:`.)

Some messages intended as errors when defining commands/environments.

```
1886 \msg_new:nnnn { cmd } { arg-after-body }
1887   { In-the-definition-of-#1,~b~(body)~argument~must~be~last. }
1888   {
1889     The~'body'~argument~type~is~followed~by~'#2'~in~the~argument~
1890     specification~of~the~#1.~This~is~not~allowed.
1891     \c__cmd_ignore_def_tl
1892   }
1893 \msg_new:nnnn { cmd } { bad-arg-spec }
1894   { Bad-argument~specification~'#2'~for~#1. }
1895   {
1896     The~argument~specification~provided~was~not~valid:~
1897     one~or~more~mandatory~pieces~of~information~were~missing.
1898     \c__cmd_ignore_def_tl
1899   }
1900 \msg_new:nnnn { cmd } { command-already-defined }
1901   { Command~'#1'~already~defined! }
1902   {
1903     You~have~used~#2~
1904     with~a~command~that~already~has~a~definition. \\\ \\\
1905     The~existing~definition~of~'#1'~will~not~be~altered.
1906   }
1907 \msg_new:nnnn { cmd } { command-not-yet-defined }
1908   { Command ~'#1'~not~yet~defined! }
1909   {
1910     You~have~used~#2~
1911     with~a~command~that~was~never~defined.
1912     \c__cmd_ignore_def_tl
1913   }
1914 \msg_new:nnnn { cmd } { environment-already-defined }
1915   { Environment~'#1'~already~defined! }
1916   {
```

```

1917     You-have-used~\NewDocumentEnvironment
1918     with-an-environment-that-already-has-a-definition. \\ \\
1919     The-existing-definition-of~'#1'~will-not-be-altered.
1920   }
1921   \msg_new:nnnn { cmd } { environment-not-yet-defined }
1922   { Environment~'#1'~not-yet-defined! }
1923   {
1924     You-have-used~\RenewDocumentEnvironment
1925     with-an-environment-that-was-never-defined.
1926     \c__cmd_ignore_def_tl
1927   }
1928   \msg_new:nnnn { cmd } { expandable-ending-optional }
1929   {
1930     Argument-specification~'#2'~for-expandable-command~'#1'~
1931     ends-with-optional-argument.
1932   }
1933   {
1934     Expandable-commands-must-have-a-final-mandatory-argument~
1935     (or-no-arguments-at-all).~You-cannot-have-a-terminal-optional~
1936     argument-with-expandable-commands.
1937   }
1938   \msg_new:nnnn { cmd } { inconsistent-long }
1939   { Inconsistent-long-arguments-for-expandable-command~'#1'. }
1940   {
1941     The-arguments-for-an-expandable-command-must-not-involve-short~
1942     arguments-after-long-arguments.~You-have-tried-to-mix-the-two-types.
1943   }
1944   \msg_new:nnnn { cmd } { invalid-command-arg }
1945   { Argument-type~'#2'~not-available-for~#1. }
1946   {
1947     The-letter~'#2'~can-only-be-used-in-environment-argument~
1948     specifications,~not~for~commands.
1949     \\ \\
1950     LaTeX-will-ignore-this-entire-definition.
1951   }
1952   \msg_new:nnnn { cmd } { invalid-expandable-argument-type }
1953   { Argument-type~'#2'~not-available-for-expandable-command~'#1'. }
1954   {
1955     The-letter~'#2'~specifies-an-argument-type-which-cannot-be-used~
1956     in-an-expandable-command.
1957     \c__cmd_ignore_def_tl
1958   }
1959   \msg_new:nnnn { cmd } { invalid-after-optional-expandably }
1960   {
1961     Argument-type~'#2'~not-available-after-optional-argument~
1962     for-expandable-command~'#1'.
1963   }
1964   {
1965     The-letter~'#2'~specifies-an-argument-type-which-cannot-be-used~
1966     in-an-expandable-command-after-an-optional-argument.
1967     \c__cmd_ignore_def_tl
1968   }
1969   \msg_new:nnnn { cmd } { non-trailing-obey-spaces }
1970   { Prefix~'!'~used-before-mandatory-argument~'#2'~of~#1. }

```

```

1971 {
1972   The~prefix~'!'~can~only~apply~to~trailing~optional~arguments.
1973   \c__cmd_ignore_def_tl
1974 }
1975 \msg_new:nnnn { cmd } { not-definable }
1976 { First~argument~of~'#2'~must~be~a~command. }
1977 {
1978   The~first~argument~of~'#2'~should~be~the~document~command~that~will~
1979   be~defined.~The~provided~argument~'#1'~is~a~character.~Perhaps~a~
1980   backslash~is~missing?
1981   \c__cmd_ignore_def_tl
1982 }
1983 \msg_new:nnnn { cmd } { not-one-token }
1984 { First~argument~of~'#2'~must~be~a~command. }
1985 {
1986   The~first~argument~of~'#2'~should~be~the~document~command~that~will~
1987   be~defined.~The~provided~argument~'#1'~contains~more~than~one~
1988   token.~Perhaps~a~backslash~is~missing?
1989   \c__cmd_ignore_def_tl
1990 }
1991 \msg_new:nnnn { cmd } { not-single-token }
1992 {
1993   Argument~delimiter~'#2'~for~the~#1~should~be~
1994   a~single~non-space~token.
1995 }
1996 {
1997   The~argument~specification~provided~was~not~valid:~in~a~place~
1998   where~a~single~token~is~required,~LaTeX~found~'#2'.
1999   \c__cmd_ignore_def_tl
2000 }
2001 \msg_new:nnnn { cmd } { forbidden-implicit-group-token }
2002 { Argument~delimiter~'#2'~for~the~#1~is~not~allowed. }
2003 {
2004   The~argument~specification~provided~was~not~valid:~the~implicit~
2005   #3-group-token~'#2'~is~not~allowed~as~an~argument~delimiter.
2006   \c__cmd_ignore_def_tl
2007 }
2008 \msg_new:nnnn { cmd } { processor-in-expandable }
2009 { Argument~processor~'>{#2}'~cannot~be~used~for~the~expandable~command~'#1'. }
2010 {
2011   The~argument~specification~for~#1~contains~a~processor~function:~
2012   this~is~only~supported~for~standard~robust~commands.
2013   \c__cmd_ignore_def_tl
2014 }
2015 \msg_new:nnnn { cmd } { too-many-arguments }
2016 { Too~many~arguments~in~argument~specification~'#2'~of~#1. }
2017 {
2018   The~argument~specification~provided~has~more~than~9~arguments.~
2019   This~cannot~be~implemented.
2020   \c__cmd_ignore_def_tl
2021 }
2022 \msg_new:nnnn { cmd } { two-markers }
2023 { Two~'#2'~apply~to~the~same~argument~in~argument~specification~of~#1. }
2024 {

```

```

2025     The~argument~specification~provided~has~two~markers~'#2'~applying~
2026     to~the~same~argument;~these~are~redundant.
2027   }
2028   \msg_new:nnnn { cmd } { unknown~argument~type }
2029   { Unknown~argument~type~'#2'~for~the~#1. }
2030   {
2031     The~letter~'#2'~does~not~specify~a~known~argument~type.
2032     \c__cmd_ignore_def_tl
2033   }
2034   \msg_new:nnnn { cmd } { xparse~argument~type }
2035   { Deprecated~argument~type~'#2'~for~the~#1~requires~xparse. }
2036   {
2037     The~letter~'#2'~specifies~a~known~argument~type~that~requires~
2038     the~xparse~package.
2039     \c__cmd_ignore_def_tl
2040   }

```

Errors when using commands/environments. The if-boolean message is always used in expandable errors. The loop-in-defaults and missing-required messages can be expandable or not expandable.

```

2041   \msg_new:nnn { cmd } { if-boolean }
2042   { Invalid~use~\iow_char:N\IfBooleanTF~{#1} }
2043   \msg_new:nnnn { cmd } { loop-in-defaults }
2044   { Defaults~of~#1~have~circular~dependency. }
2045   {
2046     The~default~values~of~two~or~more~arguments~of~the~#1~
2047     depend~on~each~other~in~a~way~that~cannot~be~resolved.
2048   }
2049   \msg_new:nnnn { cmd } { missing-required }
2050   { Missing~required~argument~for~#1. }
2051   {
2052     The~current~#1~expects~an~argument~starting~with~'#2'.~
2053     LaTeX~did~not~find~it,~and~will~insert~a~default~value~to~be~processed.
2054   }
2055   \msg_new:nnnn { cmd } { non-xparse }
2056   { \str_uppercase:n #1~not~defined~using~xparse. }
2057   {
2058     You~have~asked~for~the~argument~specification~for~the~#1,~
2059     but~this~was~not~defined~using~xparse.
2060   }
2061   \msg_new:nnnn { cmd } { split-excess-tokens }
2062   { Too~many~'#1'~tokens~when~trying~to~split~argument. }
2063   {
2064     LaTeX~was~asked~to~split~the~input~'#3'~
2065     at~each~occurrence~of~the~token~'#1',~up~to~a~maximum~of~#2~parts.~
2066     There~were~too~many~'#1'~tokens.
2067   }
2068   \msg_new:nnnn { cmd } { unknown }
2069   { Unknown~document~#1. }
2070   {
2071     You~have~asked~for~the~argument~specification~for~the~#1,~
2072     but~it~is~not~defined.
2073   }
2074   \msg_new:nnnn { cmd } { verbatim-newline }

```

```

2075 { Verbatim~argument~of~#1~ended~by~end~of~line. }
2076 {
2077   The~verbatim~argument~of~the~#1~cannot~contain~more~than~one~line,~
2078   but~the~end~
2079   of~the~current~line~has~been~reached.~You~may~have~forgotten~the~
2080   closing~delimiter.
2081   \\ \\
2082   LaTeX~will~ignore~'#2'.
2083 }
2084 \msg_new:nnnn { cmd } { verbatim-tokenized }
2085 { The~verbatim~#1~cannot~be~used~inside~an~argument. }
2086 {
2087   The~#1~takes~a~verbatim~argument.~
2088   It~may~not~appear~within~the~argument~of~another~function.~
2089   It~received~an~illegal~token \tl_if_empty:nF {#3} { ~'#3' } .
2090   \\ \\
2091   LaTeX~will~ignore~'#2'.
2092 }
2093 \msg_new:nnn { cmd } { define-command }
2094 {
2095   Defining~command~#1~
2096   with~sig.~'#2'~\msg_line_context:.
2097 }
2098 \msg_new:nnn { cmd } { define-environment }
2099 {
2100   Defining~environment~'#1'~
2101   with~sig.~'#2'~\msg_line_context:.
2102 }
2103 \msg_new:nnn { cmd } { redefine-command }
2104 {
2105   Redefining~command~#1~
2106   with~sig.~'#2'~\msg_line_context:.
2107 }
2108 \msg_new:nnn { cmd } { redefine-environment }
2109 {
2110   Redefining~environment~'#1'~
2111   with~sig.~'#2'~\msg_line_context:.
2112 }
2113 \msg_new:nnn { cmd } { optional-mandatory }
2114 {
2115   Since~the~mandatory~argument~'#1'~has~the~same~delimiter~'#2'~
2116   as~a~previous~optional~argument,~it~will~not~be~possible~to~
2117   omit~all~optional~arguments~when~calling~this~command.
2118 }
2119 \msg_new:nnn { cmd } { unsupported-let }
2120 {
2121   The~command~'#1'~was~undefined~but~not~the~associated~commands~
2122   '#1~code'~and/or~'#1~defaults'.~Maybe~you~tried~using~
2123   \iow_char:N\let.~This~may~lead~to~an~infinite~loop.
2124 }

```

1.14 User functions

The user functions are more or less just the internal functions renamed.

`\BooleanFalse` Design-space names for the Boolean values.

`\BooleanTrue` 2125 `\cs_new_eq:NN \BooleanFalse \c_false_bool`

2126 `\cs_new_eq:NN \BooleanTrue \c_true_bool`

(End definition for \BooleanFalse and \BooleanTrue.)

`\NewDocumentCommand` The user macros are pretty simple wrappers around the internal ones. There is however
`\RenewDocumentCommand` a check that the first argument is a single token, possibly surrounded by spaces (hence
`\ProvideDocumentCommand` the strange `\use:nnn`), and is definable.
`\DeclareDocumentCommand`

2127 `\cs_new_protected:Npn \NewDocumentCommand #1#2#3`

2128 `{`

2129 `__cmd_check_definable:nNT {#1} \NewDocumentCommand`

2130 `{`

2131 `\cs_if_exist:NTF #1`

2132 `{`

2133 `\msg_error:nnxx { cmd } { command-already-defined }`

2134 `{ \use:nnn \token_to_str:N #1 { } }`

2135 `{ \token_to_str:N \NewDocumentCommand }`

2136 `}`

2137 `{ __cmd_declare_cmd:Nnn #1 {#2} {#3} }`

2138 `}`

2139 `}`

2140 `\cs_new_protected:Npn \RenewDocumentCommand #1#2#3`

2141 `{`

2142 `__cmd_check_definable:nNT {#1} \RenewDocumentCommand`

2143 `{`

2144 `\cs_if_exist:NTF #1`

2145 `{ __cmd_declare_cmd:Nnn #1 {#2} {#3} }`

2146 `{`

2147 `\msg_error:nnxx { cmd } { command-not-yet-defined }`

2148 `{ \use:nnn \token_to_str:N #1 { } }`

2149 `{ \token_to_str:N \RenewDocumentCommand }`

2150 `}`

2151 `}`

2152 `}`

2153 `\cs_new_protected:Npn \ProvideDocumentCommand #1#2#3`

2154 `{`

2155 `__cmd_check_definable:nNT {#1} \ProvideDocumentCommand`

2156 `{ \cs_if_exist:NF #1 { __cmd_declare_cmd:Nnn #1 {#2} {#3} } }`

2157 `}`

2158 `\cs_new_protected:Npn \DeclareDocumentCommand #1#2#3`

2159 `{`

2160 `__cmd_check_definable:nNT {#1} \DeclareDocumentCommand`

2161 `{ __cmd_declare_cmd:Nnn #1 {#2} {#3} }`

2162 `}`

(End definition for \NewDocumentCommand and others.)

`\NewDocumentEnvironment` Very similar for environments.

`\RenewDocumentEnvironment` 2163 `\cs_new_protected:Npn \NewDocumentEnvironment #1#2#3#4`

`\ProvideDocumentEnvironment`

`\DeclareDocumentEnvironment`


```

2164 {
2165   \cs_if_exist:cTF {#1}
2166   { \msg_error:nnx { cmd } { environment-already-defined } {#1} }
2167   { \__cmd_declare_env:nnnn {#1} {#2} {#3} {#4} }
2168 }
2169 \cs_new_protected:Npn \RenewDocumentEnvironment #1#2#3#4
2170 {
2171   \cs_if_exist:cTF {#1}
2172   { \__cmd_declare_env:nnnn {#1} {#2} {#3} {#4} }
2173   { \msg_error:nnx { cmd } { environment-not-yet-defined } {#1} }
2174 }
2175 \cs_new_protected:Npn \ProvideDocumentEnvironment #1#2#3#4
2176 { \cs_if_exist:cF {#1} { \__cmd_declare_env:nnnn {#1} {#2} {#3} {#4} } }
2177 \cs_new_protected:Npn \DeclareDocumentEnvironment #1#2#3#4
2178 { \__cmd_declare_env:nnnn {#1} {#2} {#3} {#4} }

```

(End definition for `\NewDocumentEnvironment` and others.)

`\NewExpandableDocumentCommand`
`\RenewExpandableDocumentCommand`
`\ProvideExpandableDocumentCommand`
`\DeclareExpandableDocumentCommand`

The expandable versions are essentially the same as the basic functions. The strange `\use:nnn` is there in case #1 is surrounded with spaces, as can happen with usual document catcodes in `\RenewExpandableDocumentCommand { \! } ...`

```

2179 \cs_new_protected:Npn \NewExpandableDocumentCommand #1#2#3
2180 {
2181   \__cmd_check_definable:nNT {#1} \NewExpandableDocumentCommand
2182   {
2183     \cs_if_exist:NTF #1
2184     {
2185       \msg_error:nnxx { cmd } { command-already-defined }
2186       { \use:nnn \token_to_str:N #1 { } }
2187       { \token_to_str:N \NewExpandableDocumentCommand }
2188     }
2189     { \__cmd_declare_expandable_cmd:Nnn #1 {#2} {#3} }
2190   }
2191 }
2192 \cs_new_protected:Npn \RenewExpandableDocumentCommand #1#2#3
2193 {
2194   \__cmd_check_definable:nNT {#1} \RenewExpandableDocumentCommand
2195   {
2196     \cs_if_exist:NTF #1
2197     { \__cmd_declare_expandable_cmd:Nnn #1 {#2} {#3} }
2198     {
2199       \msg_error:nnxx { cmd } { command-not-yet-defined }
2200       { \use:nnn \token_to_str:N #1 { } }
2201       { \token_to_str:N \RenewExpandableDocumentCommand }
2202     }
2203   }
2204 }
2205 \cs_new_protected:Npn \ProvideExpandableDocumentCommand #1#2#3
2206 {
2207   \__cmd_check_definable:nNT {#1} \ProvideExpandableDocumentCommand
2208   {
2209     \cs_if_exist:NF #1
2210     { \__cmd_declare_expandable_cmd:Nnn #1 {#2} {#3} }
2211   }

```

```

2212 }
2213 \cs_new_protected:Npn \DeclareExpandableDocumentCommand #1#2#3
2214 {
2215   \__cmd_check_definable:nNT {#1} \DeclareExpandableDocumentCommand
2216   { \__cmd_declare_expandable_cmd:Nnn #1 {#2} {#3} }
2217 }

```

(End definition for `\NewExpandableDocumentCommand` and others.)

`\IfBooleanT` The logical `\true` and `\false` statements are just the normal `\c_true_bool` and `\c_false_bool` so `\bool_if:NNTF` is almost enough. However, this code-level function blows up badly when passed invalid input. We want `\IfBooleanTF` to accept a single (non-space) token equal to `\c_true_bool` or `\c_false_bool`, possibly surrounded by spaces. If the input is blank or multiple items, jump to the error and pick the false branch. If the input, ignoring spaces (we do this by omitting braces in the `\tl_if_single_token:nF` test), is not a single token then jump to the error as well. It is then safe to compare the token to the two booleans, picking the appropriate branch. If neither matches, we jump to the error as well.

```

2218 \cs_new:Npn \IfBooleanTF #1
2219 {
2220   \tl_if_single:nF {#1}
2221   { \prg_break:n { \use:n } }
2222   \tl_if_single_token:nF #1
2223   { \prg_break:n { \use:n } }
2224   \token_if_eq_meaning:NNT #1 \c_true_bool
2225   { \prg_break:n { \use_ii:nnn } }
2226   \token_if_eq_meaning:NNT #1 \c_false_bool
2227   { \prg_break:n { \use_iii:nnn } }
2228   \prg_break:n { \use:n }
2229   \prg_break_point:
2230   {
2231     \msg_expandable_error:nnn { cmd } { if-boolean } {#1}
2232     \use_ii:nn
2233   }
2234 }
2235 \cs_new:Npn \IfBooleanT #1#2 { \IfBooleanTF {#1} {#2} { } }
2236 \cs_new:Npn \IfBooleanF #1 { \IfBooleanTF {#1} { } }

```

(End definition for `\IfBooleanT`, `\IfBooleanF`, and `\IfBooleanTF`.)

`\IfNoValueT` Simple re-naming.

```

\IfNoValueF 2237 \cs_new_eq:NN \IfNoValueF \tl_if_novalue:nF
\IfNoValueTF 2238 \cs_new_eq:NN \IfNoValueT \tl_if_novalue:nT
2239 \cs_new_eq:NN \IfNoValueTF \tl_if_novalue:nTF

```

(End definition for `\IfNoValueT`, `\IfNoValueF`, and `\IfNoValueTF`.)

`\IfValueT` Inverted logic.

```

\IfValueF 2240 \cs_new:Npn \IfValueF { \tl_if_novalue:nT }
\IfValueTF 2241 \cs_new:Npn \IfValueT { \tl_if_novalue:nF }
2242 \cs_new:Npn \IfValueTF #1#2#3 { \tl_if_novalue:nTF {#1} {#3} {#2} }

```

(End definition for `\IfValueT`, `\IfValueF`, and `\IfValueTF`.)

`\ProcessedArgument` Processed arguments are returned using this name, which is reserved here although the definition will change.

2243 `\tl_new:N \ProcessedArgument`

(End definition for `\ProcessedArgument`.)

`\ReverseBoolean` Simple copies.

`\SplitArgument` 2244 `\cs_new_eq:NN \ReverseBoolean __cmd_bool_reverse:N`

`\SplitList` 2245 `\cs_new_eq:NN \SplitArgument __cmd_split_argument:nnn`

`\TrimSpaces` 2246 `\cs_new_eq:NN \SplitList __cmd_split_list:nn`

2247 `\cs_new_eq:NN \TrimSpaces __cmd_trim_spaces:n`

(End definition for `\ReverseBoolean` and others.)

`\ProcessList` To support `\SplitList`.

2248 `\cs_new_eq:NN \ProcessList \tl_map_function:nN`

(End definition for `\ProcessList`.)

`\GetDocumentCommandArgSpec` More simple mappings, with a check that the argument is a single control sequence or active character.

`\GetDocumentEnvironmentArgSpec`

`\ShowDocumentCommandArgSpec` 2249 `\cs_new_protected:Npn \GetDocumentCommandArgSpec #1`

`\ShowDocumentEnvironmentArgSpec`

2250 `{`
2251 `__cmd_check_definable:nNT {#1} \GetDocumentCommandArgSpec`
2252 `{ __cmd_get_arg_spec:N #1 }`

2253 `}`

2254 `\cs_new_eq:NN \GetDocumentEnvironmentArgSpec __cmd_get_arg_spec:n`

2255 `\cs_new_protected:Npn \ShowDocumentCommandArgSpec #1`

2256 `{`
2257 `__cmd_check_definable:nNT {#1} \ShowDocumentCommandArgSpec`
2258 `{ __cmd_show_arg_spec:N #1 }`

2259 `}`

2260 `\cs_new_eq:NN \ShowDocumentEnvironmentArgSpec __cmd_show_arg_spec:n`

(End definition for `\GetDocumentCommandArgSpec` and others.)

Finally as promised, restore `__kernel_chk_if_free_cs:N`:

2261 `\latexrelease\cs_gset_eq:NN __kernel_chk_if_free_cs:N __cmd_chk_if_free_cs:N`

2262 `\latexrelease\cs_undefine:N __cmd_chk_if_free_cs:N`

2263 `\latexrelease\`

2264 `\latexrelease\IncludeInRelease{0000/00/00}{\ltxcmd}%`

2265 `\latexrelease\{Document~command~parser}%`

2266 `\latexrelease\`

2267 `\latexrelease\EndModuleRelease`

2268 `\ExplSyntaxOff`

2269 `\latexrelease\@ifundefined{ExplSyntaxOff}{\{\latexrelease@postexpl}`

2270 `\latexrelease\catcode'\^~@=\@latexrelease@catcode@null\relax`

2271 `/2ekernel\latexrelease\`

We need to stop DocStrip treating `@@` in a special way at this point.

2272 `\@@=`

File h

lthooks.dtx

Contents

1 Introduction

Hooks are points in the code of commands or environments where it is possible to add processing code into existing commands. This can be done by different packages that do not know about each other and to allow for hopefully safe processing it is necessary to sort different chunks of code added by different packages into a suitable processing order.

This is done by the packages adding chunks of code (via `\AddToHook`) and labeling their code with some label by default using the package name as a label.

At `\begin{document}` all code for a hook is then sorted according to some rules (given by `\DeclareHookRule`) for fast execution without processing overhead. If the hook code is modified afterwards (or the rules are changed), a new version for fast processing is generated.

Some hooks are used already in the preamble of the document. If that happens then the hook is prepared for execution (and sorted) already at that point.

2 Package writer interface

The hook management system is offered as a set of CamelCase commands for traditional $\text{\LaTeX} 2_{\epsilon}$ packages (and for use in the document preamble if needed) as well as `expl3` commands for modern packages, that use the L3 programming layer of \LaTeX . Behind the scenes, a single set of data structures is accessed so that packages from both worlds can coexist and access hooks in other packages.

2.1 $\text{\LaTeX} 2_{\epsilon}$ interfaces

2.1.1 Declaring hooks

With a few exceptions, hooks have to be declared before they can be used. The exceptions are the generic hooks for commands, environments (i.e., executed at `\begin` and `\end`) and hooks run when loading files, e.g. before and after a package is loaded, etc. Their hook names depend on the command, environment or the file name and so declaring them beforehand is not practical.

`\NewHook`**`\NewHook {<hook>}`**

Creates a new *<hook>*. If this is a hook provided as part of a package it is suggested that the *<hook>* name is always structured as follows: *<package-name>/<hook-name>*. If necessary you can further subdivide the name by adding more / parts. If a hook name is already taken, an error is raised and the hook is not created.

The *<hook>* can be specified using the dot-syntax to denote the current package name. See section [2.1.5](#).

<code>\NewReversedHook</code>	<code>\NewReversedHook {<hook>}</code>
-------------------------------	--

Like `\NewHook` declares a new `<hook>`. the difference is that the code chunks for this hook are in reverse order by default (those added last are executed first). Any rules for the hook are applied after the default ordering. See sections 2.3 and 2.4 for further details.

The `<hook>` can be specified using the dot-syntax to denote the current package name. See section 2.1.5.

<code>\NewMirroredHookPair</code>	<code>\NewMirroredHookPair {<hook-1>} {<hook-2>}</code>
-----------------------------------	---

A shorthand for `\NewHook{<hook-1>}\NewReversedHook{<hook-2>}`.

The `<hooks>` can be specified using the dot-syntax to denote the current package name. See section 2.1.5.

2.1.2 Special declarations for hooks

The declarations here should normally not be used. They are available to provide support for special use cases mainly involving generic command hooks.

<code>\DisableHook</code>	<code>\DisableHook {<hook>}</code>
---------------------------	--

After this declaration the `<hook>` is no longer usable: Any attempt to add further code to it will result in an error and any use, e.g., via `\UseHook`, will simply do nothing.

This is intended to be used with generic command hooks (see `lthcmdhooks-doc`) as depending on the definition of the command such generic hooks may be unusable. If that is known, a package developer can disable such hooks up front.

The `<hook>` can be specified using the dot-syntax to denote the current package name. See section 2.1.5.

<code>\ProvideHook</code>	<code>\ProvideHook {<hook>}</code>
---------------------------	--

Like `\NewHook` but does nothing if the hook was previously declared with `\NewHook`. This declaration should only be used in special situations, e.g., when command of another package need to be altered and it is not clear if for that command a generic hook was already explicitly declared before.

Normally `\NewHook` should be used instead.

<code>\ProvideReversedHook</code>	<code>\ProvideReversedHook {<hook>}</code>
-----------------------------------	--

Like `\NewReversedHook` but does nothing if the hook was previously declared as a reversed hook.

<code>\ProvideMirroredHookPair</code>	<code>\ProvideMirroredHookPair {<hook-1>} {<hook-2>}</code>
---------------------------------------	---

A shorthand for `\ProvideHook{<hook-1>}\ProvideReversedHook{<hook-2>}`.

2.1.3 Using hooks in code

<code>\UseHook</code>	<code>\UseHook {<hook>}</code>
-----------------------	--------------------------------------

Execute the hook code inside a command or environment.

Before `\begin{document}` the fast execution code for a hook is not set up, so in order to use a hook there it is explicitly initialized first. As that involves assignments using a hook at those times is not 100% the same as using it after `\begin{document}`.

The `<hook>` *cannot* be specified using the dot-syntax. A leading `.` is treated literally.

\UseOneTimeHook

\UseOneTimeHook {<hook>}

Some hooks are only used (and can be only used) in one place, for example, those in `\begin{document}` or `\end{document}`. Once we have passed that point adding to the hook through a defined `\<addto-cmd>` command (e.g., `\AddToHook` or `\AtBeginDocument`, etc.) would have no effect (as would the use of such a command inside the hook code itself). It is therefore customary to redefine `\<addto-cmd>` to simply process its argument, i.e., essentially make it behave like `\@firstofone`.

`\UseOneTimeHook` does that: it records that the hook has been consumed and any further attempt to add to it will result in executing the code to be added immediately.

FMi: Maybe add an error version as well?

The <hook> cannot be specified using the dot-syntax. A leading . is treated literally. See section 2.1.5 for details.

2.1.4 Updating code for hooks

\AddToHook

\AddToHook {<hook>}[<label>]{<code>}

Adds <code> to the <hook> labeled by <label>. When the optional argument <label> is not provided, the <default label> is used (see section 2.1.5). If `\AddToHook` is used in a package/class, the <default label> is the package/class name, otherwise it is **top-level** (the **top-level** label is treated differently: see section 2.1.6).

If there already exists code under the <label> then the new <code> is appended to the existing one (even if this is a reversed hook). If you want to replace existing code under the <label>, first apply `\RemoveFromHook`.

The hook doesn't have to exist for code to be added to it. However, if it is not declared, then obviously the added <code> will never be executed. This allows for hooks to work regardless of package loading order and enables packages to add to hooks from other packages without worrying whether they are actually used in the current document. See section 2.1.8.

The <hook> and <label> can be specified using the dot-syntax to denote the current package name. See section 2.1.5.

\RemoveFromHook

\RemoveFromHook {<hook>}[<label>]

Removes any code labeled by <label> from the <hook>. When the optional argument <label> is not provided, the <default label> is used (see section 2.1.5).

If the code for that <label> wasn't yet added to the <hook>, an order is set so that when some code attempts to add that label, the removal order takes action and the code is not added.

If the optional <label> argument is *, then all code chunks are removed. This is rather dangerous as it drops code from other packages one may not know about and should therefore not be used by packages but only in document preambles!

The <hook> and <label> can be specified using the dot-syntax to denote the current package name. See section 2.1.5.

In contrast to the `\voids` relationship between two labels in a `\DeclareHookRule` this is a destructive operation as the labeled code is removed from the hook data structure, whereas the relationship setting can be undone by providing a different relationship later.

A useful application for this declaration inside the document body is when one wants to temporarily add code to hooks and later remove it again, e.g.,

```
\AddToHook{env/quote/before}{\small}
\begin{quote}
  A quote set in a smaller typeface
\end{quote}
...
\RemoveFromHook{env/quote/before}
... now back to normal for further quotes
```

Note that you can't cancel the setting with

```
\AddToHook{env/quote/before}{}
```

because that only “adds” a further empty chunk of code to the hook. Adding `\normalsize` would work but that means the hook then contained `\small\normalsize` which means two font size changes for no good reason.

The above is only needed if one wants to typeset several quotes in a smaller typeface. If the hook is only needed once then `\AddToHookNext` is simpler, because it resets itself after one use.

`\AddToHookNext`

```
\AddToHookNext {<hook>}{<code>}
```

Adds `<code>` to the next invocation of the `<hook>`. The code is executed after the normal hook code has finished and it is executed only once, i.e. it is deleted after it was used.

Using the declaration is a global operation, i.e., the code is not lost, even if the declaration is used inside a group and the next invocation happens after the group. If the declaration is used several times before the hook is executed then all code is executed in the order in which it was declared.³

It is possible to nest declarations using the same hook (or different hooks), e.g.,

```
\AddToHookNext{<hook>}{<code-1>\AddToHookNext{<hook>}{<code-2>}}
```

will execute `<code-1>` next time the `<hook>` is used and at that point puts `<code-2>` into the `<hook>` so that it gets executed on following time the hook is run.

A hook doesn't have to exist for code to be added to it. This allows for hooks to work regardless of package loading order. See section 2.1.8.

The `<hook>` can be specified using the dot-syntax to denote the current package name. See section 2.1.5.

2.1.5 Hook names and default labels

It is best practice to use `\AddToHook` in packages or classes *without specifying a <label>* because then the package or class name is automatically used, which is helpful if rules are needed, and avoids mistyping the `<label>`.

Using an explicit `<label>` is only necessary in very specific situations, e.g., if you want to add several chunks of code into a single hook and have them placed in different parts of the hook (by providing some rules).

The other case is when you develop a larger package with several sub-packages. In that case you may want to use the same `<label>` throughout the sub-packages in order to avoid that the labels change if you internally reorganize your code.

³There is no mechanism to reorder such code chunks (or delete them).

Except for `\UseHook`, `\UseOneTimeHook` and `\IfHookEmptyTF` (and their `expl3` interfaces `\hook_use:n`, `\hook_use_once:n` and `\hook_if_empty:nTF`, all $\langle hook \rangle$ and $\langle label \rangle$ arguments are processed in the same way: first, spaces are trimmed around the argument, then it is fully expanded until only character tokens remain. If the full expansion of the $\langle hook \rangle$ or $\langle label \rangle$ contains a non-expandable non-character token, a low-level `TeX` error is raised (namely, the $\langle hook \rangle$ is expanded using `TeX`'s `\csname...\endcsname`, as such, Unicode characters are allowed in $\langle hook \rangle$ and $\langle label \rangle$ arguments). The arguments of `\UseHook`, `\UseOneTimeHook`, and `\IfHookEmptyTF` are processed much in the same way except that spaces are not trimmed around the argument, for better performance.

It is not enforced, but highly recommended that the hooks defined by a package, and the $\langle labels \rangle$ used to add code to other hooks contain the package name to easily identify the source of the code chunk and to prevent clashes. This should be the standard practice, so this hook management code provides a shortcut to refer to the current package in the name of a $\langle hook \rangle$ and in a $\langle label \rangle$. If the $\langle hook \rangle$ name or the $\langle label \rangle$ consist just of a single dot (`.`), or starts with a dot followed by a slash (`./`) then the dot denotes the $\langle default label \rangle$ (usually the current package or class name—see `\SetDefaultHookLabel`). A `“.`” or `“./”` anywhere else in a $\langle hook \rangle$ or in $\langle label \rangle$ is treated literally and is not replaced.

For example, inside the package `mypackage.sty`, the default label is `mypackage`, so the instructions:

```
\NewHook    {./hook}
\AddToHook  {./hook}[.]{code}      % Same as \AddToHook{./hook}{code}
\AddToHook  {./hook}[./sub]{code}
\DeclareHookRule{begindocument}{.}{before}{babel}
\AddToHook  {file/after/foo.tex}{code}
```

are equivalent to:

```
\NewHook    {mypackage/hook}
\AddToHook  {mypackage/hook}[mypackage]{code}
\AddToHook  {mypackage/hook}[mypackage/sub]{code}
\DeclareHookRule{begindocument}{mypackage}{before}{babel}
\AddToHook  {file/after/foo.tex}{code}                % unchanged
```

The $\langle default label \rangle$ is automatically set equal to the name of the current package or class at the time the package is loaded. If the hook command is used outside of a package, or the current file wasn't loaded with `\usepackage` or `\documentclass`, then the `top-level` is used as the $\langle default label \rangle$. This may have exceptions—see `\PushDefaultHookLabel`.

This syntax is available in all $\langle label \rangle$ arguments and most $\langle hook \rangle$ arguments, both in the `LaTeX 2ε` interface, and the `LaTeX 3` interface described in section 2.2.

Note, however, that the replacement of `.` by the $\langle default label \rangle$ takes place when the hook command is executed, so actions that are somehow executed after the package ends will have the wrong $\langle default label \rangle$ if the dot-syntax is used. For that reason, this syntax is not available in `\UseHook` (and `\hook_use:n`) because the hook is most of the time used outside of the package file in which it was defined. This syntax is also not available in the hook conditionals `\IfHookEmptyTF` (and `\hook_if_empty:nTF`), because these conditionals are used in some performance-critical parts of the hook management code, and because they are usually used to refer to other package's hooks, so the dot-syntax doesn't make much sense.

In some cases, for example in large packages, one may want to separate it in logical parts, but still use the main package name as $\langle label \rangle$, then the $\langle default label \rangle$ can be set using `\SetDefaultHookLabel` or `\PushDefaultHookLabel{...}\PopDefaultHookLabel`.

<code>\PushDefaultHookLabel</code>	<code>\PushDefaultHookLabel {$\langle default label \rangle$}</code>
<code>\PopDefaultHookLabel</code>	<code>$\langle code \rangle$</code>

`\PopDefaultHookLabel`

`\PushDefaultHookLabel` sets the current $\langle default label \rangle$ to be used in $\langle label \rangle$ arguments, or when replacing a leading “.” (see above). `\PopDefaultHookLabel` reverts the $\langle default label \rangle$ to its previous value.

Inside a package or class, the $\langle default label \rangle$ is equal to the package or class name, unless explicitly changed. Everywhere else, the $\langle default label \rangle$ is **top-level** (see section 2.1.6) unless explicitly changed.

The effect of `\PushDefaultHookLabel` holds until the next `\PopDefaultHookLabel`. `\usepackage` (and `\RequirePackage` and `\documentclass`) internally use

```
\PushDefaultHookLabel{ $\langle package name \rangle$ }
 $\langle package code \rangle$ 
\PopDefaultHookLabel
```

to set the $\langle default label \rangle$ for the package or class file. Inside the $\langle package code \rangle$ the $\langle default label \rangle$ can also be changed with `\SetDefaultHookLabel`. `\input` and other file input-related commands from the L^AT_EX kernel do not use `\PushDefaultHookLabel`, so code within files loaded by these commands does *not* get a dedicated $\langle label \rangle$! (that is, the $\langle default label \rangle$ is the current active one when the file was loaded.)

Packages that provide their own package-like interfaces (TikZ’s `\usetikzlibrary`, for example) can use `\PushDefaultHookLabel` and `\PopDefaultHookLabel` to set dedicated labels and emulate `\usepackage`-like hook behaviour within those contexts.

The **top-level** label is treated differently, and is reserved to the user document, so it is not allowed to change the $\langle default label \rangle$ to **top-level**.

<code>\SetDefaultHookLabel</code>	<code>\SetDefaultHookLabel {$\langle default label \rangle$}</code>
-----------------------------------	--

Similarly to `\PushDefaultHookLabel`, sets the current $\langle default label \rangle$ to be used in $\langle label \rangle$ arguments, or when replacing a leading “.”. The effect holds until the label is changed again or until the next `\PopDefaultHookLabel`. The difference between `\PushDefaultHookLabel` and `\SetDefaultHookLabel` is that the latter does not save the current $\langle default label \rangle$.

This command is useful when a large package is composed of several smaller packages, but all should have the same $\langle label \rangle$, so `\SetDefaultHookLabel` can be used at the beginning of each package file to set the correct label.

`\SetDefaultHookLabel` is not allowed in the main document, where the $\langle default label \rangle$ is **top-level** and there is no `\PopDefaultHookLabel` to end its effect. It is also not allowed to change the $\langle default label \rangle$ to **top-level**.

2.1.6 The top-level label

The **top-level** label, assigned to code added from the main document, is different from other labels. Code added to hooks (usually `\AtBeginDocument`) in the preamble is almost always to change something defined by a package, so it should go at the very end of the hook.

Therefore, code added in the `top-level` is always executed at the end of the hook, regardless of where it was declared. If the hook is reversed (see `\NewReversedHook`), the `top-level` chunk is executed at the very beginning instead.

Rules regarding `top-level` have no effect: if a user wants to have a specific set of rules for a code chunk, they should use a different label to said code chunk, and provide a rule for that label instead.

The `top-level` label is exclusive for the user, so trying to add code with that label from a package results in an error.

2.1.7 Defining relations between hook code

The default assumption is that code added to hooks by different packages are independent and the order in which they are executed is irrelevant. While this is true in many cases it is obviously false in others.

Before the hook management system was introduced packages had to take elaborate precaution to determine if some other package got loaded as well (before or after) and find some ways to alter its behavior accordingly. In addition it was often the user's responsibility to load packages in the right order so that code added to hooks got added in the right order and some cases even altering the loading order wouldn't resolve the conflicts.

With the new hook management system it is now possible to define rules (i.e., relationships) between code chunks added by different packages and explicitly describe in which order they should be processed.

<code>\DeclareHookRule</code>	<code>\DeclareHookRule {<hook>}{<label1>}{<relation>}{<label2>}</code>
-------------------------------	--

Defines a relation between `<label1>` and `<label2>` for a given `<hook>`. If `<hook>` is `??` this defines a default relation for all hooks that use the two labels, i.e., that have chunks of code labeled with `<label1>` and `<label2>`. Rules specific to a given hook take precedence over default rules that use `??` as the `<hook>`.

Currently, the supported relations are the following:

before or **<** Code for `<label1>` comes before code for `<label2>`.

after or **>** Code for `<label1>` comes after code for `<label2>`.

incompatible-warning Only code for either `<label1>` or `<label2>` can appear for that hook (a way to say that two packages—or parts of them—are incompatible). A warning is raised if both labels appear in the same hook.

incompatible-error Like **incompatible-warning** but instead of a warning a L^AT_EX error is raised, and the code for both labels are dropped from that hook until the conflict is resolved.

voids Code for `<label1>` overwrites code for `<label2>`. More precisely, code for `<label2>` is dropped for that hook. This can be used, for example if one package is a superset in functionality of another one and therefore wants to undo code in some hook and replace it with its own version.

unrelated The order of code for `<label1>` and `<label2>` is irrelevant. This rule is there to undo an incorrect rule specified earlier.

There can only be a single relation between two labels for a given hook, i.e., a later `\DeclareHookrule` overwrites any previous declaration.

The `<hook>` and `<label>` can be specified using the dot-syntax to denote the current package name. See section 2.1.5.

<code>\ClearHookRule</code>	<code>\ClearHookRule{<hook>}{<label1>}{<label2>}</code>
-----------------------------	---

Syntactic sugar for saying that `<label1>` and `<label2>` are unrelated for the given `<hook>`.

<code>\DeclareDefaultHookRule</code>	<code>\DeclareDefaultHookRule{<label1>}{<relation>}{<label2>}</code>
--------------------------------------	--

This sets up a relation between `<label1>` and `<label2>` for all hooks unless overwritten by a specific rule for a hook. Useful for cases where one package has a specific relation to some other package, e.g., is **incompatible** or always needs a special ordering **before** or **after**. (Technically it is just a shorthand for using `\DeclareHookRule` with `??` as the hook name.)

Declaring default rules is only supported in the document preamble.⁴

The `<label>` can be specified using the dot-syntax to denote the current package name. See section 2.1.5.

2.1.8 Querying hooks

Simpler data types, like token lists, have three possible states; they can:

- exist and be empty;

⁴Trying to do so, e.g., via `\DeclareHookRule` with `??` has bad side-effects and is not supported (though not explicitly caught for performance reasons).

- exist and be non-empty; and
- not exist (in which case emptiness doesn’t apply);

Hooks are a bit more complicated: a hook may exist or not, and either way it may or may not be empty. This means that even a hook that doesn’t exist may be non-empty and it can also be disabled.

This seemingly strange state may happen when, for example, package *A* defines hook `A/foo`, and package *B* adds some code to that hook. However, a document may load package *B* before package *A*, or may not load package *A* at all. In both cases some code is added to hook `A/foo` without that hook being defined yet, thus that hook is said to be non-empty, whereas it doesn’t exist. Therefore, querying the existence of a hook doesn’t imply its emptiness, neither does the other way around.

Given that code or rules can be added to a hook even if it doesn’t physically exist yet, means that a querying its existence has no real use case (in contrast to other variables that can only be update if they have already been declared). For that reason only the test for emptiness has a public interface.

A hook is said to be empty when no code was added to it, either to its permanent code pool, or to its “next” token list. The hook doesn’t need to be declared to have code added to its code pool. A hook is said to exist when it was declared with `\NewHook` or some variant thereof. Generic hooks such as `file` and `env` hooks are automatically declared when code is added to them.

<code>\IfHookEmptyTF</code> ★	<code>\IfHookEmptyTF {<hook>} {<true code>} {<false code>}</code>
-------------------------------	---

Tests if the *<hook>* is empty (*i.e.*, no code was added to it using either `\AddToHook` or `\AddToHookNext`) or such code was removed again (via `\RemoveFromHook`), and branches to either *<true code>* or *<false code>* depending on the result.

The *<hook>* *cannot* be specified using the dot-syntax. A leading `.` is treated literally.

2.1.9 Displaying hook code

If one has to adjust the code execution in a hook using a hook rule it is helpful to get some information about the code associated with a hook, its current order and the existing rules.

<code>\ShowHook</code>	<code>\ShowHook {<hook>}</code>
<code>\LogHook</code>	Displays information about the <i><hook></i> such as

- the code chunks (and their labels) added to it,
- any rules set up to order them,
- the computed order in which the chunks are executed,
- any code executed on the next invocation only.

`\LogHook` prints the information to the `.log` file, and `\ShowHook` prints them to the terminal/command window and starts `TEX`’s prompt (only in `\errorstopmode`) to wait for user action.

The *<hook>* can be specified using the dot-syntax to denote the current package name. See section [2.1.5](#).

Suppose a hook `example-hook` whose output of `\ShowHook{example-hook}` is:

```

1  -> The hook 'example-hook':
2  > Code chunks:
3  >   foo -> [code from package 'foo']
4  >   bar -> [from package 'bar']
5  >   baz -> [package 'baz' is here]
6  > Document-level (top-level) code (executed last):
7  >   -> [code from 'top-level']
8  > Extra code for next invocation:
9  >   -> [one-time code]
10 > Rules:
11 >   foo|baz with relation >
12 >   baz|bar with default relation <
13 > Execution order (after applying rules):
14 >   baz, foo, bar.
```

In the listing above, lines 3 to 5 show the three code chunks added to the hook and their respective labels in the format

`<label> -> <code>`

Line 7 shows the code chunk added by the user in the main document (labeled `top-level`) in the format

Document-level (top-level) code (executed `<first|last>`):
 -> `<top-level code>`

This code will be either the first or last code executed by the hook (`last` if the hook is normal, `first` if it is reversed). This chunk is not affected by rules and does not take part in sorting.

Line 9 shows the code chunk for the next execution of the hook in the format

-> `<next-code>`

This code will be used and disappear at the next `\UseHook{example-hook}`, in contrast to the chunks mentioned earlier, which can only be removed from that hook by doing `\RemoveFromHook{<label>}[example-hook]`.

Lines 11 and 12 show the rules declared that affect this hook in the format

`<label-1>|<label-2> with <default?> relation <relation>`

which means that the `<relation>` applies to `<label-1>` and `<label-2>`, in that order, as detailed in `\DeclareHookRule`. If the relation is `default` it means that that rule applies to `<label-1>` and `<label-2>` in *all* hooks, (unless overridden by a non-default relation).

Finally, line 14 lists the labels in the hook after sorting; that is, in the order they will be executed when the hook is used.

2.1.10 Debugging hook code

`\DebugHooksOn`
`\DebugHooksOff`

`\DebugHooksOn`

Turn the debugging of hook code on or off. This displays most changes made to the hook data structures. The output is rather coarse and not really intended for normal use.

2.2 L3 programming layer (expl3) interfaces

This is a quick summary of the L^AT_EX3 programming interfaces for use with packages written in `expl3`. In contrast to the L^AT_EX 2_ε interfaces they always use mandatory arguments only, e.g., you always have to specify the $\langle label \rangle$ for a code chunk. We therefore suggest to use the declarations discussed in the previous section even in `expl3` packages, but the choice is yours.

<code>\hook_new:n</code>	<code>\hook_new:n {\langle hook \rangle}</code>
<code>\hook_new_reversed:n</code>	<code>\hook_new_reversed:n {\langle hook \rangle}</code>
<code>\hook_new_pair:nn</code>	<code>\hook_new_pair:nn {\langle hook-1 \rangle} {\langle hook-2 \rangle}</code>

Creates a new $\langle hook \rangle$ with normal or reverse ordering of code chunks. `\hook_new_pair:nn` creates a pair of such hooks with $\{\langle hook-2 \rangle\}$ being a reversed hook. If a hook name is already taken, an error is raised and the hook is not created.

The $\langle hook \rangle$ can be specified using the dot-syntax to denote the current package name. See section 2.1.5.

<code>\hook_disable:n</code>	<code>\hook_disable:n {\langle hook \rangle}</code>
------------------------------	---

Marks $\{\langle hook \rangle\}$ as disabled. Any further attempt to add code to it or declare it, will result in an error and any call to `\hook_use:n` will simply do nothing.

This declaration is intended for use with generic hooks that are known not to work (see `ltxcmdhooks-doc`) if they receive code.

The $\langle hook \rangle$ can be specified using the dot-syntax to denote the current package name. See section 2.1.5.

<code>\hook_provide:n</code>	<code>\hook_provide:n {\langle hook \rangle}</code>
------------------------------	---

Like `\hook_new:n` but does nothing if the hook was previously declared with `\hook_new:n`. This declaration should only be used in special situations, e.g., when a command of another package needs to be altered and it is not clear if for that command a generic `cmd` hook was already explicitly declared before.

Normally `\hook_new:n` should be used instead.

<code>\hook_provide_reversed:n</code>	<code>\hook_provide_reversed:n {\langle hook \rangle}</code>
---------------------------------------	--

Like `\hook_new_reversed:n` but does nothing if the hook was previously declared as a reversed hook.

<code>\hook_provide_pair:nn</code>	<code>\hook_provide_pair:nn {\langle hook-1 \rangle} {\langle hook-2 \rangle}</code>
------------------------------------	--

A shorthand for `\hook_provide:n{\langle hook-1 \rangle}\hook_provide_reversed:n{\langle hook-2 \rangle}`.

<code>\hook_use:n</code>	<code>\hook_use:n {\langle hook \rangle}</code>
--------------------------	---

Executes the $\{\langle hook \rangle\}$ code followed (if set up) by the code for next invocation only, then empties that next invocation code.

The $\langle hook \rangle$ *cannot* be specified using the dot-syntax. A leading `.` is treated literally.

<code>\hook_use_once:n</code>	<code>\hook_use_once:n {\langle hook \rangle}</code>
-------------------------------	--

Changes the $\{\langle hook \rangle\}$ status so that from now on any addition to the hook code is executed immediately. Then execute any $\{\langle hook \rangle\}$ code already set up.

The $\langle hook \rangle$ *cannot* be specified using the dot-syntax. A leading `.` is treated literally.

<hr/> <hr/> <code>\hook_gput_code:nnn</code>	<code>\hook_gput_code:nnn {<hook>} {<label>} {<code>}</code> <p>Adds a chunk of <code><code></code> to the <code><hook></code> labeled <code><label></code>. If the label already exists the <code><code></code> is appended to the already existing code.</p> <p>If code is added to an external <code><hook></code> (of the kernel or another package) then the convention is to use the package name as the <code><label></code> not some internal module name or some other arbitrary string.</p> <p>The <code><hook></code> and <code><label></code> can be specified using the dot-syntax to denote the current package name. See section 2.1.5.</p>
<hr/> <hr/> <code>\hook_gput_next_code:nn</code>	<code>\hook_gput_next_code:nn {<hook>} {<code>}</code> <p>Adds a chunk of <code><code></code> for use only in the next invocation of the <code><hook></code>. Once used it is gone.</p> <p>This is simpler than <code>\hook_gput_code:nnn</code>, the code is simply appended to the hook in the order of declaration at the very end, i.e., after all standard code for the hook got executed.</p> <p>Thus if one needs to undo what the standard does one has to do that as part of <code><code></code>.</p> <p>The <code><hook></code> can be specified using the dot-syntax to denote the current package name. See section 2.1.5.</p>
<hr/> <hr/> <code>\hook_gremove_code:nn</code>	<code>\hook_gremove_code:nn {<hook>} {<label>}</code> <p>Removes any code for <code><hook></code> labeled <code><label></code>.</p> <p>If the code for that <code><label></code> wasn't yet added to the <code><hook></code>, an order is set so that when some code attempts to add that label, the removal order takes action and the code is not added.</p> <p>If the second argument is <code>*</code>, then all code chunks are removed. This is rather dangerous as it drops code from other packages one may not know about, so think twice before using that!</p> <p>The <code><hook></code> and <code><label></code> can be specified using the dot-syntax to denote the current package name. See section 2.1.5.</p>
<hr/> <hr/> <code>\hook_gset_rule:nnnn</code>	<code>\hook_gset_rule:nnnn {<hook>} {<label1>} {<relation>} {<label2>}</code> <p>Relate <code><label1></code> with <code><label2></code> when used in <code><hook></code>. See <code>\DeclareHookRule</code> for the allowed <code><relation></code>s. If <code><hook></code> is <code>??</code> a default rule is specified.</p> <p>The <code><hook></code> and <code><label></code> can be specified using the dot-syntax to denote the current package name. See section 2.1.5. The dot-syntax is parsed in both <code><label></code> arguments, but it usually makes sense to be used in only one of them.</p>
<hr/> <hr/> <code>\hook_if_empty_p:n *</code> <hr/> <hr/> <code>\hook_if_empty:nTF *</code>	<code>\hook_if_empty:nTF {<hook>} {<true code>} {<false code>}</code> <p>Tests if the <code><hook></code> is empty (<i>i.e.</i>, no code was added to it using either <code>\AddToHook</code> or <code>\AddToHookNext</code>), and branches to either <code><true code></code> or <code><false code></code> depending on the result.</p> <p>The <code><hook></code> <i>cannot</i> be specified using the dot-syntax. A leading <code>.</code> is treated literally.</p>

<code>\hook_show:n</code>	<code>\hook_show:n {<hook>}</code>
<code>\hook_log:n</code>	

Displays information about the $\langle hook \rangle$ such as

- the code chunks (and their labels) added to it,
- any rules set up to order them,
- the computed order in which the chunks are executed,
- any code executed on the next invocation only.

`\hook_log:n` prints the information to the `.log` file, and `\hook_show:n` prints them to the terminal/command window and starts TeX's prompt (only if `\errorstopmode`) to wait for user action.

The $\langle hook \rangle$ can be specified using the dot-syntax to denote the current package name. See section 2.1.5.

<code>\hook_debug_on:</code>	<code>\hook_debug_on:</code>
<code>\hook_debug_off:</code>	

Turns the debugging of hook code on or off. This displays changes to the hook data.

2.3 On the order of hook code execution

Chunks of code for a $\langle hook \rangle$ under different labels are supposed to be independent if there are no special rules set up that define a relation between the chunks. This means that you can't make assumptions about the order of execution!

Suppose you have the following declarations:

```
\NewHook{myhook}
\AddToHook{myhook}[packageA]{\typeout{A}}
\AddToHook{myhook}[packageB]{\typeout{B}}
\AddToHook{myhook}[packageC]{\typeout{C}}
```

then executing the hook with `\UseHook` will produce the typeout A B C in that order. In other words, the execution order is computed to be `packageA`, `packageB`, `packageC` which you can verify with `\ShowHook{myhook}`:

```
-> The hook 'myhook':
> Code chunks:
>   packageA -> \typeout {A}
>   packageB -> \typeout {B}
>   packageC -> \typeout {C}
> Document-level (top-level) code (executed last):
>   ---
> Extra code for next invocation:
>   ---
> Rules:
>   ---
> Execution order:
>   packageA, packageB, packageC.
```

The reason is that the code chunks are internally saved in a property list and the initial order of such a property list is the order in which key-value pairs got added. However, that is only true if nothing other than adding happens!

Suppose, for example, you want to replace the code chunk for `packageA`, e.g.,

```
\RemoveFromHook{myhook}[packageA]
\AddToHook{myhook}[packageA]{\typeout{A alt}}
```

then your order becomes `packageB`, `packageC`, `packageA` because the label got removed from the property list and then re-added (at its end).

While that may not be too surprising, the execution order is also sometimes altered if you add a redundant rule, e.g. if you specify

```
\DeclareHookRule{myhook}{packageA}{before}{packageB}
```

instead of the previous lines we get

```
-> The hook 'myhook':
> Code chunks:
>   packageA -> \typeout {A}
>   packageB -> \typeout {B}
>   packageC -> \typeout {C}
> Document-level (top-level) code (executed last):
>   ---
> Extra code for next invocation:
>   ---
> Rules:
>   packageB|packageA with relation >
> Execution order (after applying rules):
>   packageA, packageC, packageB.
```

As you can see the code chunks are still in the same order, but in the execution order for the labels `packageB` and `packageC` have swapped places. The reason is that, with the rule there are two orders that satisfy it, and the algorithm for sorting happened to pick a different one compared to the case without rules (where it doesn't run at all as there is nothing to resolve). Incidentally, if we had instead specified the redundant rule

```
\DeclareHookRule{myhook}{packageB}{before}{packageC}
```

the execution order would not have changed.

In summary: it is not possible to rely on the order of execution unless there are rules that partially or fully define the order (in which you can rely on them being fulfilled).

2.4 The use of “reversed” hooks

You may have wondered why you can declare a “reversed” hook with `\NewReversedHook` and what that does exactly.

In short: the execution order of a reversed hook (without any rules!) is exactly reversed to the order you would have gotten for a hook declared with `\NewHook`.

This is helpful if you have a pair of hooks where you expect to see code added that involves grouping, e.g., starting an environment in the first and closing that environment in the second hook. To give a somewhat contrived example⁵, suppose there is a package adding the following:

⁵there are simpler ways to achieve the same effect.

```

\AddToHook{env/quote/before}[package-1]{\begin{itshape}}
\AddToHook{env/quote/after} [package-1]{\end{itshape}}

```

As a result, all quotes will be in italics. Now suppose further that another `package-too` makes the quotes also in blue and therefore adds:

```

\usepackage{color}
\AddToHook{env/quote/before}[package-too]{\begin{color}{blue}}
\AddToHook{env/quote/after} [package-too]{\end{color}}

```

Now if the `env/quote/after` hook would be a normal hook we would get the same execution order in both hooks, namely:

```
package-1, package-too
```

(or vice versa) and as a result, would get:

```

\begin{itshape}\begin{color}{blue} ...
\end{itshape}\end{color}

```

and an error message that `\begin{color}` ended by `\end{itshape}`. With `env/quote/after` declared as a reversed hook the execution order is reversed and so all environments are closed in the correct sequence and `\ShowHook` would give us the following output:

```

-> The hook 'env/quote/after':
> Code chunks:
>   package-1 -> \end {itshape}
>   package-too -> \end {color}
> Document-level (top-level) code (executed first):
>   ---
> Extra code for next invocation:
>   ---
> Rules:
>   ---
> Execution order (after reversal):
>   package-too, package-1.

```

The reversal of the execution order happens before applying any rules, so if you alter the order you will probably have to alter it in both hooks, not just in one, but that depends on the use case.

2.5 Difference between “normal” and “one-time” hooks

When executing a hook a developer has the choice of using either `\UseHook` or `\UseOneTimeHook` (or their `expl3` equivalents `\hook_use:n` and `\hook_use_once:n`). This choice affects how `\AddToHook` is handled after the hook has been executed for the first time.

With normal hooks adding code via `\AddToHook` means that the code chunk is added to the hook data structure and then used each time `\UseHook` is called.

With one-time hooks it this is handled slightly differently: After `\UseOneTimeHook` has been called, any further attempts to add code to the hook via `\AddToHook` will simply execute the `<code>` immediately.

This has some consequences one needs to be aware of:

- If $\langle code \rangle$ is added to a normal hook after the hook was executed and it is never executed again for one or the other reason, then this new $\langle code \rangle$ will never be executed.
- In contrast if that happens with a one-time hook the $\langle code \rangle$ is executed immediately.

In particular this means that construct such as

```
\AddToHook{myhook}
{  $\langle code-1 \rangle$  \AddToHook{myhook}{ $\langle code-2 \rangle$ }  $\langle code-3 \rangle$  }
```

works for one-time hooks⁶ (all three code chunks are executed one after another), but it makes little sense with a normal hook, because with a normal hook the first time `\UseHook{myhook}` is executed it would

- execute $\langle code-1 \rangle$,
- then execute `\AddToHook{myhook}{code-2}` which adds the code chunk $\langle code-2 \rangle$ to the hook for use on the next invocation,
- and finally execute $\langle code-3 \rangle$.

The second time `\UseHook` is called it would execute the above and in addition $\langle code-2 \rangle$ as that was added as a code chunk to the hook in the meantime. So each time the hook is used another copy of $\langle code-2 \rangle$ is added and so that code chunk is executed $\langle \# \text{ of invocations} \rangle - 1$ times.

2.6 Private L^AT_EX kernel hooks

There are a few places where it is absolutely essential for L^AT_EX to function correctly that code is executed in a precisely defined order. Even that could have been implemented with the hook management (by adding various rules to ensure the appropriate ordering with respect to other code added by packages). However, this makes every document unnecessary slow, because there has to be sorting even though the result is predetermined. Furthermore it forces package writers to unnecessarily add such rules if they add further code to the hook (or break L^AT_EX).

For that reason such code is not using the hook management, but instead private kernel commands directly before or after a public hook with the following naming convention: `\@kernel@before@hook` or `\@kernel@after@hook`. For example, in `\enddocument` you find

```
\UseHook{enddocument}%
\@kernel@after@enddocument
```

which means first the user/package-accessible `enddocument` hook is executed and then the internal kernel hook. As their name indicates these kernel commands should not be altered by third-party packages, so please refrain from that in the interest of stability and instead use the public hook next to it.⁷

⁶This is sometimes used with `\AtBeginDocument` which is why it is supported.

⁷As with everything in T_EX there is no enforcement of this rule, and by looking at the code it is easy to find out how the kernel adds to them. The main reason of this section is therefore to say “please don’t do that, this is unconfigurable code!”

2.7 Legacy L^AT_EX 2_ε interfaces

L^AT_EX 2_ε offered a small number of hooks together with commands to add to them. They are listed here and are retained for backwards compatibility.

With the new hook management several additional hooks have been added to L^AT_EX and more will follow. See the next section for what is already available.

<code>\AtBeginDocument</code>	<code>\AtBeginDocument [<i>label</i>] {<i>code</i>}</code>
-------------------------------	--

If used without the optional argument *label*, it works essentially like before, i.e., it is adding *code* to the hook `begindocument` (which is executed inside `\begin{document}`). However, all code added this way is labeled with the label `top-level` (see section 2.1.6) if done outside of a package or class or with the package/class name if called inside such a file (see section 2.1.5).

This way one can add further code to the hook using `\AddToHook` or `\AtBeginDocument` using a different label and explicitly order the code chunks as necessary, e.g., run some code before or after another package's code. When using the optional argument the call is equivalent to running `\AddToHook {begindocument} [label] {code}`.

`\AtBeginDocument` is a wrapper around the `begindocument` hook (see section 2.8.4), which is a one-time hook. As such, after the `begindocument` hook is executed at `\begin{document}` any attempt to add *code* to this hook with `\AtBeginDocument` or with `\AddToHook` will cause that *code* to execute immediately instead. See section 2.5 for more on one-time hooks.

For important packages with known order requirement we may over time add rules to the kernel (or to those packages) so that they work regardless of the loading-order in the document.

<code>\AtEndDocument</code>	<code>\AtEndDocument [<i>label</i>] {<i>code</i>}</code>
-----------------------------	--

Like `\AtBeginDocument` but for the `enddocument` hook.

There is also `\AtBeginDvi` which is discussed in conjunction with the shipout hooks.

The few hooks that existed previously in L^AT_EX 2_ε used internally commands such as `@begindocumenthook` and packages sometimes augmented them directly rather than working through `\AtBeginDocument`. For that reason there is currently support for this, that is, if the system detects that such an internal legacy hook command contains code it adds it to the new hook system under the label `legacy` so that it doesn't get lost.

However, over time the remaining cases of direct usage need updating because in one of the future release of L^AT_EX we will turn this legacy support off, as it does unnecessary slow down the processing.

2.8 L^AT_EX 2_ε commands and environments augmented by hooks

intro to be written

2.8.1 Generic hooks for all environments

Every environment *env* has now four associated hooks coming with it:

env/⟨env⟩/before This hook is executed as part of `\begin` as the very first action, in particular prior to starting the environment group. Its scope is therefore not restricted by the environment.

env/⟨env⟩/begin This hook is executed as part of `\begin` directly in front of the code specific to the environment start (e.g., the second argument of `\newenvironment`). Its scope is the environment body.

env/⟨env⟩/end This hook is executed as part of `\end` directly in front of the code specific to the end of the environment (e.g., the third argument of `\newenvironment`).

env/⟨env⟩/after This hook is executed as part of `\end` after the code specific to the environment end and after the environment group has ended. Its scope is therefore not restricted by the environment.

The hook is implemented as a reversed hook so if two packages add code to `env/⟨env⟩/before` and to `env/⟨env⟩/after` they can add surrounding environments and the order of closing them happens in the right sequence.

Generic environment hooks are never one-time hooks even with environments that are supposed to appear only once in a document.⁸ In contrast to other hooks there is also no need to declare them using `\NewHook`.

The hooks are only executed if `\begin{⟨env⟩}` and `\end{⟨env⟩}` is used. If the environment code is executed via low-level calls to `\⟨env⟩` and `\end⟨env⟩` (e.g., to avoid the environment grouping) they are not available. If you want them available in code using this method, you would need to add them yourself, i.e., write something like

```
\UseHook{env/quote/before}\quote
...
\endquote\UseHook{env/quote/after}
```

to add the outer hooks, etc.

`\BeforeBeginEnvironment`

`\BeforeBeginEnvironment [⟨label⟩] {⟨code⟩}`

This declaration adds to the `env/⟨env⟩/before` hook using the `⟨label⟩`. If `⟨label⟩` is not given, the `⟨default label⟩` is used (see section 2.1.5).

`\AtBeginEnvironment`

`\AtBeginEnvironment [⟨label⟩] {⟨code⟩}`

Like `\BeforeBeginEnvironment` but adds to the `env/⟨env⟩/begin` hook.

`\AtEndEnvironment`

`\AtEndEnvironment [⟨label⟩] {⟨code⟩}`

Like `\BeforeBeginEnvironment` but adds to the `env/⟨env⟩/end` hook.

`\AfterEndEnvironment`

`\AfterEndEnvironment [⟨label⟩] {⟨code⟩}`

Like `\BeforeBeginEnvironment` but adds to the `env/⟨env⟩/after` hook.

⁸Thus if one adds code to such hooks after the environment has been processed, it will only be executed if the environment appears again and if that doesn't happen the code will never get executed.

2.8.2 Generic hooks for commands

Similar to environments there are now (at least in theory) two generic hooks available for any \LaTeX command. These are

cmd/⟨name⟩/before This hook is executed at the very start of the command execution.

cmd/⟨name⟩/after This hook is executed at the very end of the command body. It is implemented as a reversed hook.

In practice there are restrictions and especially the **after** hook works only with a subset of commands. Details about these restrictions are documented in `ltxcmdhooks-doc.pdf` or with code in `ltxcmdhooks-code.pdf`.

2.8.3 Generic hooks provided by file loading operations

There are several hooks added to \LaTeX 's process of loading file via its high-level interfaces such as `\input`, `\include`, `\usepackage`, `\RequirePackage`, etc. These are documented in `ltxfilehook-doc.pdf` or with code in `ltxfilehook-code.pdf`.

2.8.4 Hooks provided by `\begin{document}`

Until 2020 `\begin{document}` offered exactly one hook that one could add to using `\AtBeginDocument`. Experiences over the years have shown that this single hook in one place was not enough and as part of adding the general hook management system a number of additional hooks have been added at this point. The places for these hooks have been chosen to provide the same support as offered by external packages, such as `etoolbox` and others that augmented `\document` to gain better control.

Supported are now the following hooks (all of them one-time hooks):

begindocument/before This hook is executed at the very start of `\document`, one can think of it as a hook for code at the end of the preamble section and this is how it is used by `etoolbox`'s `\AtEndPreamble`.

This is a one-time hook, so after it is executed, all further attempts to add code to it will execute such code immediately (see section 2.5).

begindocument This hook is added to when using `\AtBeginDocument` and it is executed after the `.aux` file as be read in and most initialization are done, so they can be altered and inspected by the hook code. It is followed by a small number of further initializations that shouldn't be altered and are therefore coming later.

The hook should not be used to add material for typesetting as we are still in \LaTeX 's initialization phase and not in the document body. If such material needs to be added to the document body use the next hook instead.

This is a one-time hook, so after it is executed, all further attempts to add code to it will execute such code immediately (see section 2.5).

begindocument/end This hook is executed at the end of the `\document` code in other words at the beginning of the document body. The only command that follows it is `\ignorespaces`.

This is a one-time hook, so after it is executed, all further attempts to add code to it will execute such code immediately (see section 2.5).

The generic hooks executed by `\begin` also exist, i.e., `env/document/before` and `env/document/begin`, but with this special environment it is better use the dedicated one-time hooks above.

2.8.5 Hooks provided by `\end{document}`

L^AT_EX 2_ε always provided `\AtEndDocument` to add code to the execution of `\end{document}` just in front of the code that is normally executed there. While this was a big improvement over the situation in L^AT_EX 2.09 it was not flexible enough for a number of use cases and so packages, such as `etoolbox`, `atveryend` and others patched `\enddocument` to add additional points where code could be hooked into.

Patching using packages is always problematical as leads to conflicts (code availability, ordering of patches, incompatible patches, etc.). For this reason a number of additional hooks have been added to the `\enddocument` code to allow packages to add code in various places in a controlled way without the need for overwriting or patching the core code.

Supported are now the following hooks (all of them one-time hooks):

`\enddocument` The hook associated with `\AtEndDocument`. It is immediately called at the beginning of `\enddocument`.

When this hook is executed there may be still unprocessed material (e.g., floats on the deferlist) and the hook may add further material to be typeset. After it, `\clearpage` is called to ensure that all such material gets typeset. If there is nothing waiting the `\clearpage` has no effect.

This is a one-time hook, so after it is executed, all further attempts to add code to it will execute such code immediately (see section 2.5).

`\enddocument/afterlastpage` As the name indicates this hook should not receive code that generates material for further pages. It is the right place to do some final housekeeping and possibly write out some information to the `.aux` file (which is still open at this point to receive data, but since there will be no more pages you need to write to it using `\immediate\write`). It is also the correct place to set up any testing code to be run when the `.aux` file is re-read in the next step.

After this hook has been executed the `.aux` file is closed for writing and then read back in to do some tests (e.g., looking for missing references or duplicated labels, etc.).

This is a one-time hook, so after it is executed, all further attempts to add code to it will execute such code immediately (see section 2.5).

`\enddocument/afteraux` At this point, the `.aux` file has been reprocessed and so this is a possible place for final checks and display of information to the user. However, for the latter you might prefer the next hook, so that your information is displayed after the (possibly longish) list of files if that got requested via `\listfiles`.

This is a one-time hook, so after it is executed, all further attempts to add code to it will execute such code immediately (see section 2.5).

`\enddocument/info` This hook is meant to receive code that write final information messages to the terminal. It follows immediately after the previous hook (so both could have been combined, but then packages adding further code would always need to also supply an explicit rule to specify where it should go).

This hook already contains some code added by the kernel (under the labels `kernel/filelist` and `kernel/warnings`), namely the list of files when `\listfiles` has been used and the warnings for duplicate labels, missing references, font substitutions etc.

This is a one-time hook, so after it is executed, all further attempts to add code to it will execute such code immediately (see section 2.5).

enddocument/end Finally, this hook is executed just in front of the final call to `\@@end`.

This is a one-time hook, so after it is executed, all further attempts to add code to it will execute such code immediately (see section 2.5). is it even possible to add code after this one?

There is also the hook `shipout/lastpage`. This hook is executed as part of the last `\shipout` in the document to allow package to add final `\special`'s to that page. Where this hook is executed in relation to those from the above list can vary from document to document. Furthermore to determine correctly which of the `\shipouts` is the last one, \LaTeX needs to be run several times, so initially it might get executed on the wrong page. See section 2.8.6 for where to find the details.

It is also possible to use the generic `env/document/end` hook which is executed by `\end`, i.e., just in front of the first hook above. Note however that the other generic `\end` environment hook, i.e., `env/document/after` will never get executed, because by that time \LaTeX has finished the document processing.

2.8.6 Hooks provided by `\shipout` operations

There are several hooks and mechanisms added to \LaTeX 's process of generating pages. These are documented in `ltshipout-doc.pdf` or with code in `ltshipout-code.pdf`.

2.8.7 Hooks provided in NFSS commands

In languages that need to support for more than one script in parallel (and thus several sets of fonts), e.g., Latin and Japanese fonts, NFSS font commands, such as `\sffamily`, need to switch both the Latin family to “Sans Serif” and in addition alter a second set of fonts.

To support this several NFSS have hooks in which such support can be added.

rmfamily After `\rmfamily` has done its initial checks and prepared a any font series update this hook is executed and only afterwards `\selectfont`.

sffamily Like the `rmfamily` hook but for the `\sffamily` command.

ttfamily Like the `rmfamily` hook but for the `\ttfamily` command.

normalfont The `\normalfont` command resets font encoding family series and shape to their document defaults. It then executes this hook and finally calls `\selectfont`.

expand@font@defaults The internal `\expand@font@defaults` command expands and saves the current defaults for the meta families (rm/sf/tt) and the meta series (bf/md). If the NFSS machinery has been augmented, e.g., for Chinese or Japanese fonts, then further defaults may need to be set at this point. This can be done in this hook which is executed at the end of this macro.

bfseries/defaults, bfseries If the `\bfdefault` was explicitly changed by the user its new value is used to set the bf series defaults for the meta families (rm/sf/tt) when `\bfseries` is called. In the **bfseries/defaults** hook further adjustments can be made in this case. This hook is only executed if such a change is detected. In contrast the **bfseries** hook is always executed just before `\selectfont` is called to change to the new series.

mdseries/defaults, mdseries These two hooks are like the previous ones but used in `\mdseries` command.

3 The Implementation

```

1 <@@=hook>
2 <*2ekernel | latexrelease>
3 \ExplSyntaxOn
4 <latexrelease>\NewModuleRelease{2020/10/01}{lthooks}
5 <latexrelease>                                {The~hook~management~system}

```

3.1 Debugging

```

\g__hook_debug_bool Holds the current debugging state.
6 \bool_new:N \g__hook_debug_bool

(End definition for \g__hook_debug_bool.)

\hook_debug_on: Turns debugging on and off by redefining \__hook_debug:n.
\hook_debug_off:
\__hook_debug:n
\__hook_debug_gset:
7 \cs_new_eq:NN \__hook_debug:n \use_none:n
8 \cs_new_protected:Npn \hook_debug_on:
9 {
10   \bool_gset_true:N \g__hook_debug_bool
11   \__hook_debug_gset:
12 }
13 \cs_new_protected:Npn \hook_debug_off:
14 {
15   \bool_gset_false:N \g__hook_debug_bool
16   \__hook_debug_gset:
17 }
18 \cs_new_protected:Npn \__hook_debug_gset:
19 {
20   \cs_gset_protected:Npx \__hook_debug:n ##1
21   { \bool_if:NT \g__hook_debug_bool {##1} }
22 }

(End definition for \hook_debug_on: and others. These functions are documented on page 177.)

```

3.2 Borrowing from internals of other kernel modules

```

\__hook_str_compare:nn Private copy of \__str_if_eq:nn
23 \cs_new_eq:NN \__hook_str_compare:nn \__str_if_eq:nn

(End definition for \__hook_str_compare:nn.)

```

3.3 Declarations

<code>\l__hook_tmpa_bool</code>	Scratch boolean used throughout the package. <code>24 \bool_new:N \l__hook_tmpa_bool</code> <i>(End definition for \l__hook_tmpa_bool.)</i>
<code>\l__hook_return_tl</code> <code>\l__hook_tmpa_tl</code> <code>\l__hook_tmpb_tl</code>	Scratch variables used throughout the package. <code>25 \tl_new:N \l__hook_return_tl</code> <code>26 \tl_new:N \l__hook_tmpa_tl</code> <code>27 \tl_new:N \l__hook_tmpb_tl</code> <i>(End definition for \l__hook_return_tl, \l__hook_tmpa_tl, and \l__hook_tmpb_tl.)</i>
<code>\g__hook_all_seq</code>	In a few places we need a list of all hook names ever defined so we keep track if them in this sequence. <code>28 \seq_new:N \g__hook_all_seq</code> <i>(End definition for \g__hook_all_seq.)</i>
<code>\g__hook_removal_list_prop</code>	A token list to hold delayed removals. <code>29 \tl_new:N \g__hook_removal_list_tl</code> <i>(End definition for \g__hook_removal_list_prop.)</i>
<code>\l__hook_cur_hook_tl</code>	Stores the name of the hook currently being sorted. <code>30 \tl_new:N \l__hook_cur_hook_tl</code> <i>(End definition for \l__hook_cur_hook_tl.)</i>
<code>\l__hook_work_prop</code>	A property list holding a copy of the <code>\g__hook_⟨hook⟩_code_prop</code> of the hook being sorted to work on, so that changes don't act destructively on the hook data structure. <code>31 \prop_new:N \l__hook_work_prop</code> <i>(End definition for \l__hook_work_prop.)</i>
<code>\g__hook_execute_immediately_prop</code>	List of hooks that from now on should not longer receive code. <code>32 \prop_new:N \g__hook_execute_immediately_prop</code> <i>(End definition for \g__hook_execute_immediately_prop.)</i>
<code>\g__hook_used_prop</code>	All hooks that receive code (for use in debugging display). <code>33 \prop_new:N \g__hook_used_prop</code> <i>(End definition for \g__hook_used_prop.)</i>
<code>\g__hook_hook_curr_name_tl</code> <code>\g__hook_name_stack_seq</code>	Default label used for hook commands, and a stack to keep track of packages within packages. <code>34 \tl_new:N \g__hook_hook_curr_name_tl</code> <code>35 \seq_new:N \g__hook_name_stack_seq</code> <i>(End definition for \g__hook_hook_curr_name_tl and \g__hook_name_stack_seq.)</i>
<code>__hook_tmp:w</code>	Temporary macro for generic usage. <code>36 \cs_new_eq:NN __hook_tmp:w ?</code>

```

(End definition for \_hook_tmp:w.)

\tl_gremove_once:Nx Some variants of expl3 functions.
\tl_show:x          FMi: should probably be moved to expl3
\tl_log:x

37 \cs_generate_variant:Nn \tl_gremove_once:Nn { Nx }
38 \cs_generate_variant:Nn \tl_show:n { x }
39 \cs_generate_variant:Nn \tl_log:n { x }

(End definition for \tl_gremove_once:Nx, \tl_show:x, and \tl_log:x.)

\s__hook_mark Scan mark used for delimited arguments.
40 \scan_new:N \s__hook_mark

(End definition for \s__hook_mark.)

\_hook_tl_set:Nn Private copies of a few expl3 functions. l3debug will only add debugging to the public
\_hook_tl_set:Nx names, not to these copies, so we don't have to use \debug_suspend: and \debug_-
\_hook_tl_set:cn resume: everywhere.
\_hook_tl_set:cx Functions like \_hook_tl_set:Nn have to be redefined, rather than copied because
in expl3 they use \_kernel_tl_(g)set:Nx, which is also patched by l3debug.

41 \cs_new_protected:Npn \_hook_tl_set:Nn #1#2
42 { \cs_set_nopar:Npx #1 { \_kernel_exp_not:w {#2} } }
43 \cs_new_protected:Npn \_hook_tl_set:Nx #1#2
44 { \cs_set_nopar:Npx #1 {#2} }
45 \cs_generate_variant:Nn \_hook_tl_set:Nn { c }
46 \cs_generate_variant:Nn \_hook_tl_set:Nx { c }

(End definition for \_hook_tl_set:Nn.)

\_hook_tl_gset:Nn Same as above.
\_hook_tl_gset:No 47 \cs_new_protected:Npn \_hook_tl_gset:Nn #1#2
\_hook_tl_gset:Nx 48 { \cs_gset_nopar:Npx #1 { \_kernel_exp_not:w {#2} } }
\_hook_tl_gset:cn 49 \cs_new_protected:Npn \_hook_tl_gset:No #1#2
\_hook_tl_gset:co 50 { \cs_gset_nopar:Npx #1 { \_kernel_exp_not:w \exp_after:wN {#2} } }
\_hook_tl_gset:cx 51 \cs_new_protected:Npn \_hook_tl_gset:Nx #1#2
52 { \cs_gset_nopar:Npx #1 {#2} }
53 \cs_generate_variant:Nn \_hook_tl_gset:Nn { c }
54 \cs_generate_variant:Nn \_hook_tl_gset:No { c }
55 \cs_generate_variant:Nn \_hook_tl_gset:Nx { c }

(End definition for \_hook_tl_gset:Nn.)

\_hook_tl_gput_right:Nn Same as above.
\_hook_tl_gput_right:No 56 \cs_new_protected:Npn \_hook_tl_gput_right:Nn #1#2
\_hook_tl_gput_right:cn 57 { \_hook_tl_gset:Nx #1 { \_kernel_exp_not:w \exp_after:wN { #1 #2 } } }
58 \cs_generate_variant:Nn \_hook_tl_gput_right:Nn { No, cn }

(End definition for \_hook_tl_gput_right:Nn.)

\_hook_tl_gput_left:Nn Same as above.
\_hook_tl_gput_left:No 59 \cs_new_protected:Npn \_hook_tl_gput_left:Nn #1#2
60 {
61 \_hook_tl_gset:Nx #1
62 { \_kernel_exp_not:w {#2} \_kernel_exp_not:w \exp_after:wN {#1} }
63 }
64 \cs_generate_variant:Nn \_hook_tl_gput_left:Nn { No }

```

(End definition for `_hook_tl_gput_left:Nn`.)

`_hook_tl_gset_eq:NN` Same as above.

```
65 \cs_new_eq:NN \_hook_tl_gset_eq:NN \tl_gset_eq:NN
```

(End definition for `_hook_tl_gset_eq:NN`.)

`_hook_tl_gclear:N` Same as above.

```
\_hook_tl_gclear:c
66 \cs_new_protected:Npn \_hook_tl_gclear:N #1
67 { \_hook_tl_gset_eq:NN #1 \c_empty_tl }
68 \cs_generate_variant:Nn \_hook_tl_gclear:N { c }
```

(End definition for `_hook_tl_gclear:N`.)

3.4 Providing new hooks

3.4.1 The data structures of a hook

`\g_@@_<hook>_code_prop` Hooks have a name (called *<hook>* in the description below) and for each hook we have to provide a number of data structures. These are

`\@@_<hook>`

`\@@_next_<hook>` `\g__hook_<hook>_code_prop` A property list holding the code for the hook in separate chunks. The keys are by default the package names that add code to the hook, but it is possible for packages to define other keys.

`\g__hook_<hook>_rule_<label1>|<label2>_tl` A token list holding the relation between *<label1>* and *<label2>* in the *<hook>*. The *<labels>* are lexically (reverse) sorted to ensure that two labels always point to the same token list. For global rules, the *<hook>* name is `??`.

`_hook_<hook>` The code that is actually executed when the hook is called in the document is stored in this token list. It is constructed from the code chunks applying the information. This token list is named like that so that in case of an error inside the hook, the reported token list in the error is shorter, and to make it simpler to normalize hook names in `_hook_make_name:n`.

`\g__hook_<hook>_reversed_tl` Some hooks are “reversed”. This token list stores a `-` for such hook so that it can be identified. The `-` character is used because *<reversed>*1 is `+1` for normal hooks and `-1` for reversed ones.

`\g__hook_<hook>_declared_tl` This token list serves as marker for the hook being officially declared. Its existence is tested to raise an error in case another declaration is attempted.

`_hook_toplevel_<hook>` This token list stores the code inserted in the hook from the user’s document, in the `top-level` label. This label is special, and doesn’t participate in sorting. Instead, all code is appended to it and executed after (or before, if the hook is reversed) the normal hook code, but before the `next` code chunk.

`__hook_next_⟨hook⟩` Finally there is extra code (normally empty) that is used on the next invocation of the hook (and then deleted). This can be used to define some special behavior for a single occasion from within the document. This token list follows the same naming scheme than the main `__hook_⟨hook⟩` token list. It is called `__hook_next_⟨hook⟩` rather than `__hook_next_⟨hook⟩` because otherwise a hook whose name is `next_⟨hook⟩` would clash with the next code-token list of the hook called `⟨hook⟩`.

3.4.2 On the existence of hooks

A hook may be in different states of existence. Here we give an overview of internal commands to set up hooks and explain how the different states are distinguished. The actual implementation then follows in the next sections.

One problem we have to solve is, that we need to be able to add code to hooks (e.g., with `\AddToHook`) even if that code has not been declared yet. For example, one package needs to write into a hook of another package, but that package may not get loaded or only loaded later. Another problem most hooks require declaration but this is not the case for the generic hooks.

We therefore distinguish the following states for a hook and they are managed with four different tests: structure existence (`__hook_if_structure_exist:nTF`), creation (`__hook_if_usable:nTF`), declaration (`__hook_if_declared:nTF`) and disabled or not (`__hook_if_disabled:nTF`)

not existing Nothing is known about the hook so far. This state can be detected with `__hook_if_structure_exist:nTF` (which uses the false branch).

In this state the hook can be declared, disabled, rules can be defined or code could be added to it, but it is not possible to use the hook (with `\UseHook`).

basic data structure set up A hook is in this state when its basic data structure has been set up (using `__hook_init_structure:n`). The data structure setup happens automatically when commands such as `\AddToHook` are used and the hook is at that point in state “not existing”.

In this state the four tests give the following results:

```
\__hook_if_structure_exist:nTF returns true.
      \__hook_if_usable:nTF returns false.
      \__hook_if_declared:nTF returns false.
      \__hook_if_disabled:nTF returns false.
```

The allowed actions are the same as in the “not existing” state.

declared A hook is in this state it is not disabled and was explicitly declared (e.g., with `\NewHook`). In this case the four tests give the following results:

```
\__hook_if_structure_exist:nTF returns true.
      \__hook_if_usable:nTF returns true.
      \__hook_if_declared:nTF returns true.
      \__hook_if_disabled:nTF returns false.
```

usable A hook is in this state if it is not disabled, was not explicitly declared but nevertheless is allowed to be used (with `\UseHook` or `\hook_use:n`). This state is only possible for generic hooks as they do not need to be declared. Therefore such hooks move directly from state “not existing” to “usable” the moment a declaration such as `\AddToHook` wants to add to the hook data structure. In this state the tests give the following results:

```
\__hook_if_structure_exist:nTF returns true.
    \__hook_if_usable:nTF returns true.
    \__hook_if_declared:nTF returns false.
    \__hook_if_disabled:nTF returns false.
```

disabled A hook in any state is moved to this state when `\DisableHook` is used. This changes the tests to give the following results:

```
\__hook_if_structure_exist:nTF unchanged.
    \__hook_if_usable:nTF returns false.
    \__hook_if_declared:nTF returns true.
    \__hook_if_disabled:nTF returns true.
```

The structure test is unchanged (if the hook was unknown before it is false, otherwise true). The usable test returns false so that any `\UseHook` will bypass the hook from now on. The declared test returns true so that any further `\NewHook` generates an error and the disabled test returns true so that `\AddToHook` can return an error.

FMi: maybe it should do this only after begin document?

3.4.3 Setting hooks up

`\hook_new:n` The `\hook_new:n` declaration declares a new hook and expects the hook $\langle name \rangle$ as its argument, e.g., `begindocument`.

```
69 \cs_new_protected:Npn \hook_new:n #1
70   { \__hook_normalize_hook_args:Nn \__hook_new:n {#1} }
71 \cs_new_protected:Npn \__hook_new:n #1
72   {
```

We check if the hook was already *explicitly* declared with `\hook_new:n`, and if it already exists we complain, otherwise set the “created” flag for the hook so that it errors next time `\hook_new:n` is used.

```
73   \__hook_if_declared:nTF {#1}
74   { \msg_error:nnn { hooks } { exists } {#1} }
75   {
76     \tl_new:c { g__hook_#1_declared_tl }
77     \__hook_make_usable:n {#1}
78   }
79 }
```

(End definition for `\hook_new:n` and `__hook_new:n`. This function is documented on page 175.)

`__hook_make_usable:n` This initializes all hook data structures for the hook but if used on its own doesn't mark the hook as declared (as `\hook_new:n` does, so a later `\hook_new:n` on that hook will not result in an error. This command is internally used by `\hook_gput_code:n` when adding code to a generic hook.

```
80 \cs_new_protected:Npn \__hook_make_usable:n #1
81 {
```

Now we check if the hook's data structure can be safely created without `expl3` raising errors, then we add the hook name to the list of all hooks and allocate the necessary data structures for the new hook, otherwise just do nothing.

```
82 \tl_if_exist:cF { __hook~#1 }
83 {
84 \seq_gput_right:Nn \g__hook_all_seq {#1}
```

This is only used by the actual code of the current hook, so declare it normally:

```
85 \tl_new:c { __hook~#1 }
```

Now ensure that the base data structure for the hook exists:

```
86 \__hook_init_structure:n {#1}
```

The `\g__hook_<hook>_labels_clist` holds the sorted list of labels (once it got sorted). This is used only for debugging.

```
87 \clist_new:c { g__hook_#1_labels_clist }
```

Some hooks should reverse the default order of code chunks. To signal this we have a token list which is empty for normal hooks and contains a `-` for reversed hooks.

```
88 \tl_new:c { g__hook_#1_reversed_tl }
```

The above is all in L3 convention, but we also provide an interface to legacy $\text{\LaTeX 2}_{\epsilon}$ hooks of the form `\@...hook`, e.g., `\@begindocumenthook`. There have been a few of them and they have been added to using `\g@addto@macro`. If there exists such a macro matching the name of the new hook, i.e., `\@<hook-name>hook` and it is not empty then we add its contents as a code chunk under the label `legacy`.

Warning: this support will vanish in future releases!

```
89 \__hook_include_legacy_code_chunk:n {#1}
90 }
91 }
```

(End definition for `__hook_make_usable:n`.)

`__hook_init_structure:n` This function declares the basic data structures for a hook without explicitly declaring the hook itself. This is needed to allow adding to undeclared hooks. Here it is unnecessary to check whether all variables exist, since all three are declared at the same time (either all of them exist, or none).

It creates the hook code pool (`\g__hook_<hook>_code_prop`) and the top-level and `next` token lists. A hook is initialized with `__hook_init_structure:n` the first time anything is added to it. Initializing a hook just with `__hook_init_structure:n` will not make it usable with `\hook_use:n`.

```
92 \cs_new_protected:Npn \__hook_init_structure:n #1
93 {
94 \__hook_if_structure_exist:nF {#1}
95 {
96 \prop_new:c { g__hook_#1_code_prop }
97 \tl_new:c { __hook_toplevel~#1 }
```

```

98         \tl_new:c { __hook_next~#1 }
99     }
100 }

```

(End definition for `_hook_init_structure:n`.)

`\hook_new_reversed:n` Declare a new hook. The default ordering of code chunks is reversed, signaled by setting the token list to a minus sign.

```

101 \cs_new_protected:Npn \hook_new_reversed:n #1
102 { \__hook_normalize_hook_args:Nn \__hook_new_reversed:n {#1} }
103 \cs_new_protected:Npn \__hook_new_reversed:n #1
104 {
105     \__hook_new:n {#1}

```

If the hook already exists the above will generate an error message, so the next line should be executed (but it is — too bad).

```

106     \tl_gset:cn { g__hook_#1_reversed_tl } { - }
107 }

```

(End definition for `\hook_new_reversed:n` and `__hook_new_reversed:n`. This function is documented on page 175.)

`\hook_new_pair:nn` A shorthand for declaring a normal and a (matching) reversed hook in one go.

```

108 \cs_new_protected:Npn \hook_new_pair:nn #1#2
109 { \hook_new:n {#1} \hook_new_reversed:n {#2} }

```

(End definition for `\hook_new_pair:nn`. This function is documented on page 175.)

`__hook_include_legacy_code_chunk:n` The L^AT_EX legacy concept for hooks uses with hooks the following naming scheme in the code: `\@...hook`.

If this macro is not empty we add it under the label `legacy` to the current hook and then empty it globally. This way packages or classes directly manipulating commands such as `\@begindocumenthook` still get their hook data added.

Warning: this support will vanish in future releases!

```

110 \cs_new_protected:Npn \__hook_include_legacy_code_chunk:n #1
111 {

```

If the macro doesn't exist (which is the usual case) then nothing needs to be done.

```

112     \tl_if_exist:cT { @#1hook }

```

Of course if the legacy hook exists but is empty, there is no need to add anything under `legacy` the legacy label.

```

113     {
114         \tl_if_empty:cF { @#1hook }
115         {
116             \exp_args:Nnnv \__hook_hook_gput_code_do:nnn {#1}
117                 { legacy } { @#1hook }

```

Once added to the hook, we need to clear it otherwise it might get added again later if the hook data gets updated.

```

118         \__hook_tl_gclear:c { @#1hook }
119     }
120 }
121 }

```

(End definition for `__hook_include_legacy_code_chunk:n`.)

3.4.4 Disabling and providing hooks

`\hook_disable:n` Disables a hook by creating its `\g__hook_⟨hook⟩_declared_tl` so that the hook errors when used with `\hook_new:n`, then it undefines `__hook_⟨hook⟩` so that it may not be executed.

`__hook_if_disabled_p:n` This does not clear any code that may be already stored in the hook’s structure, but doesn’t allow adding more code. `__hook_if_disabled:nTF` uses that specific combination to check if the hook is disabled.

```

122 <latexrelease>\IncludeInRelease{2021/06/01}%
123 <latexrelease>          {\hook_disable:n}{Disable~hooks}

124 \cs_new_protected:Npn \hook_disable:n #1
125   { \__hook_normalize_hook_args:Nn \__hook_disable:n {#1} }
126 \cs_new_protected:Npn \__hook_disable:n #1
127   {
128     \tl_gclear_new:c { g__hook_#1_declared_tl }
129     \cs_undefine:c { __hook~#1 }
130   }
131 \prg_new_conditional:Npnn \__hook_if_disabled:n #1 { p, T, F, TF }
132   {
133     \bool_lazy_and:nnTF
134       { \tl_if_exist_p:c { g__hook_#1_declared_tl } }
135       { ! \tl_if_exist_p:c { __hook~#1 } }
136       { \prg_return_true: }
137       { \prg_return_false: }
138   }
139 <latexrelease>\EndIncludeInRelease

140 <latexrelease>\IncludeInRelease{2020/10/01}
141 <latexrelease>          {\hook_disable:n}{Disable~hooks}
142 <latexrelease>
143 <latexrelease>\cs_new_protected:Npn \hook_disable:n #1 {}
144 <latexrelease>
145 <latexrelease>\EndIncludeInRelease

```

(End definition for `\hook_disable:n`, `__hook_disable:n`, and `__hook_if_disabled:nTF`. This function is documented on page 175.)

`\hook_provide:n` The `\hook_provide:n` declaration declares a new hook if it wasn’t declared already, in which case it only checks that the already existing hook is not a reversed hook. The `\hook_provide_reversed:n` does the same for reversed hooks. `begindocument`.

`__hook_provide:n`

```

146 <latexrelease>\IncludeInRelease{2021/06/01}%
147 <latexrelease>          {\hook_provide:n}{Providing~hooks}

148 \cs_new_protected:Npn \hook_provide:n #1
149   { \__hook_normalize_hook_args:Nn \__hook_provide:nn {#1} { } }
150 \cs_new_protected:Npn \hook_provide_reversed:n #1
151   { \__hook_normalize_hook_args:Nn \__hook_provide:nn {#1} { - } }

152 \cs_new_protected:Npn \__hook_provide:nn #1 #2
153   {

```

If the hook to be provided was disabled we warn (for now — this may change).

```

154   \__hook_if_disabled:nTF {#1}
155     { \msg_warning:nnn { hooks } { provide-disabled } {#1} }

```

Otherwise we check if it was already declared.

```

156     {
157         \__hook_if_declared:nTF {#1}
158         {

```

Issue an error if we try to provide a a hook that is reversed and the already existing one is not (or vice versa).

```

159             \str_if_eq:eeF { \tl_use:c { g__hook_#1_reversed_tl } } {#2}
160             { \msg_error:nnn { hooks } { provide-error } {#1} }
161         }

```

If it wasn't declared, we declared as a normal or reversed hook as appropriate.

```

162     {
163         \tl_new:c { g__hook_#1_declared_tl }
164         \__hook_make_usable:n {#1}
165         \tl_gset:cn { g__hook_#1_reversed_tl } {#2}
166     }
167 }
168 }

```

(End definition for `\hook_provide:n`, `\hook_provide_reversed:n`, and `__hook_provide:n`. These functions are documented on page 175.)

`\hook_provide_pair:nn` A shorthand for providing a normal and a (matching) reversed hook in one go.

```

169 \cs_new_protected:Npn \hook_provide_pair:nn #1#2
170 { \hook_provide:n {#1} \hook_provide_reversed:n {#2} }

```

(End definition for `\hook_provide_pair:nn`. This function is documented on page 175.)

```

171 <latexrelease>\EndIncludeInRelease
172 <latexrelease>\IncludeInRelease{2020/10/01}
173 <latexrelease>          {\hook_provide:n}{Providing~hooks}
174 <latexrelease>
175 <latexrelease>\cs_new_protected:Npn \hook_provide_reversed:n #1 {}
176 <latexrelease>\cs_new_protected:Npn \hook_provide:n #1 {}
177 <latexrelease>\cs_new_protected:Npn \hook_provide_pair:nn #1#2 {}
178 <latexrelease>
179 <latexrelease>\EndIncludeInRelease

```

3.5 Parsing a label

`__hook_parse_label_default:n` This macro checks if a label was given (not `\c_novalue_tl`), and if so, tries to parse the label looking for a leading `.` to replace by `__hook_currname_or_default:`.

```

180 \cs_new:Npn \__hook_parse_label_default:n #1
181 {
182     \tl_if_novalue:nTF {#1}
183     { \__hook_currname_or_default: }
184     { \tl_trim_spaces_apply:nN {#1} \__hook_parse_dot_label:n }
185 }

```

(End definition for `__hook_parse_label_default:n`.)

`__hook_parse_dot_label:n` Start by checking if the label is empty, which raises an error, and uses the fallback value.
`__hook_parse_dot_label:w` If not, split the label at a `.`, if any, and check if no tokens are before the `.`, or if the
`__hook_parse_dot_label_cleanup:w` only character is a `.`. If these requirements are fulfilled, the leading `.` is replaced with
`__hook_parse_dot_label_aux:w` `__hook_currname_or_default:.` Otherwise the label is returned unchanged.

```

186 \cs_new:Npn \__hook_parse_dot_label:n #1
187 {
188   \tl_if_empty:NTF {#1}
189   {
190     \msg_expandable_error:nn { hooks } { empty-label }
191     \__hook_currname_or_default:
192   }
193   {
194     \str_if_eq:nnTF {#1} { . }
195     { \__hook_currname_or_default: }
196     { \__hook_parse_dot_label:w #1 ./ \s__hook_mark }
197   }
198 }
199 \cs_new:Npn \__hook_parse_dot_label:w #1 ./ #2 \s__hook_mark
200 {
201   \tl_if_empty:NTF {#1}
202   { \__hook_parse_dot_label_aux:w #2 \s__hook_mark }
203   {
204     \tl_if_empty:NTF {#2}
205     { \__hook_make_name:n {#1} }
206     { \__hook_parse_dot_label_cleanup:w #1 ./ #2 \s__hook_mark }
207   }
208 }
209 \cs_new:Npn \__hook_parse_dot_label_cleanup:w #1 ./ \s__hook_mark {#1}
210 \cs_new:Npn \__hook_parse_dot_label_aux:w #1 ./ \s__hook_mark
211 { \__hook_currname_or_default: / \__hook_make_name:n {#1} }

```

(End definition for __hook_parse_dot_label:n and others.)

`__hook_currname_or_default:` Uses `\g__hook_hook_curr_name_tl` if it is set, otherwise tries `\@currname`. If neither is set, raises an error and uses the fallback value `label-missing`.

```

212 \cs_new:Npn \__hook_currname_or_default:
213 {
214   \tl_if_empty:NTF \g__hook_hook_curr_name_tl
215   {
216     \tl_if_empty:NTF \@currname
217     {
218       \msg_expandable_error:nnn { kernel } { should-not-happen }
219       { Empty~default~label. }
220       \__hook_make_name:n { label-missing }
221     }
222     { \@currname }
223   }
224   { \g__hook_hook_curr_name_tl }
225 }

```

(End definition for __hook_currname_or_default:.)

`__hook_make_name:n` Provides a standard sanitization of a hook's name. It uses `\cs:w` to build a control
`__hook_make_name:w` sequence out of the hook name, then uses `\cs_to_str:N` to get the string representation

of that, without the escape character. `\cs:w`-based expansion is used instead of `e`-based because Unicode characters don't behave well inside `\expanded`. The macro adds the `_hook_` prefix to the hook name to reuse the hook's code token list to build the csnames and avoid leaving “public” control sequences defined (as `\relax`) in TeX's memory.

```

226 \cs_new:Npn \__hook_make_name:n #1
227 {
228   \exp_after:wN \exp_after:wN \exp_after:wN \__hook_make_name:w
229   \exp_after:wN \token_to_str:N \cs:w \_hook~ #1 \cs_end:
230 }
231 \exp_last_unbraced:NNNN
232 \cs_new:Npn \__hook_make_name:w #1 \tl_to_str:n { \_hook~ } { }
```

(End definition for `__hook_make_name:n` and `__hook_make_name:w`.)

Standard route for normalising hook and label arguments. The main macro does the entire operation within a group so that csnames made by `__hook_make_name:n` are wiped off before continuing. This means that this function cannot be used for `\hook_`-`use:n`!

```

233 \cs_new_protected:Npn \__hook_normalize_hook_args_aux:Nn #1 #2
234 {
235   \group_begin:
236   \use:e
237   {
238     \group_end:
239     \exp_not:N #1 #2
240   }
241 }
242 \cs_new_protected:Npn \__hook_normalize_hook_args:Nn #1 #2
243 {
244   \__hook_normalize_hook_args_aux:Nn #1
245   { { \__hook_parse_label_default:n {#2} } }
246 }
247 \cs_new_protected:Npn \__hook_normalize_hook_args:Nnn #1 #2 #3
248 {
249   \__hook_normalize_hook_args_aux:Nn #1
250   {
251     { \__hook_parse_label_default:n {#2} }
252     { \__hook_parse_label_default:n {#3} }
253   }
254 }
255 \cs_new_protected:Npn \__hook_normalize_hook_rule_args:Nnnnn #1 #2 #3 #4 #5
256 {
257   \__hook_normalize_hook_args_aux:Nn #1
258   {
259     { \__hook_parse_label_default:n {#2} }
260     { \__hook_parse_label_default:n {#3} }
261     { \tl_trim_spaces:n {#4} }
262     { \__hook_parse_label_default:n {#5} }
263   }
264 }
```

(End definition for `__hook_normalize_hook_args:Nn` and others.)

3.6 Adding or removing hook code

```

\hook_gput_code:nnn With \hook_gput_code:nnn{<hook>}{<label>}{<code>} a chunk of <code> is added to an
\__hook_gput_code:nnn existing <hook> labeled with <label>.
\__hook_gput_code:nxv
\__hook_hook_gput_code_do:nnn
265 \cs_new_protected:Npn \hook_gput_code:nnn #1 #2
266 { \__hook_normalize_hook_args:Nnn \__hook_gput_code:nnn {#1} {#2} }
267 \cs_new_protected:Npn \__hook_gput_code:nnn #1 #2 #3
268 {

```

First check if the hook was used as a one-time hook:

```

269 \prop_if_in:NnTF \g__hook_execute_immediately_prop {#1}
270 {#3}
271 {

```

Then check if the current *<hook>/<label>* pair was marked for removal, in which case `__hook_unmark_removal:nn` is used to remove that mark (once). This may happen when a package removes code from another package which was not yet loaded: the removal order is stored, and at this stage it is executed by not adding to the hook.

```

272 \__hook_if_marked_removal:nnTF {#1} {#2}
273 { \__hook_unmark_removal:nn {#1} {#2} }
274 {

```

If no removal is queued, we are free to add. Start by checking if the hook exists.

```

275 \__hook_if_usable:nTF {#1}

```

If so we simply add (or append) the new code to the property list holding different chunks for the hook. At `\begin{document}` this is then sorted into a token list for fast execution.

```

276 {
277 \__hook_hook_gput_code_do:nnn {#1} {#2} {#3}

```

However, if there is an update within the document we need to alter this execution code which is done by `__hook_update_hook_code:n`. In the preamble this does nothing.

```

278 \__hook_update_hook_code:n {#1}
279 }

```

If the hook does not exist, however, before giving up try to declare it as a generic hook, if its name matches one of the valid patterns.

```

280 {
281 \__hook_if_disabled:nTF {#1}
282 { \msg_error:nnn { hooks } { hook-disabled } {#1} }
283 { \__hook_try_declaring_generic_hook:nnn {#1} {#2} {#3} }
284 }
285 }
286 }
287 }

```

```

288 \cs_generate_variant:Nn \__hook_gput_code:nnn { nxv }

```

This macro will unconditionally add a chunk of code to the given hook.

```

289 \cs_new_protected:Npn \__hook_hook_gput_code_do:nnn #1 #2 #3
290 {

```

However, first some debugging info if debugging is enabled:

```

291 \__hook_debug:n{\iow_term:x{****~ Add~ to~
292 \__hook_if_usable:nF {#1} { undeclared~ }
293 hook~ #1~ (#2)
294 \on@line\space <-- \tl_to_str:n{#3}} }

```

Then try to get the code chunk labeled #2 from the hook. If there's code already there, then append #3 to that, otherwise just put #3. If the current label is `top-level`, the code is added to a dedicated token list `__hook_toplevel_\hook` that goes at the end of the hook (or at the beginning, for a reversed hook), just before `__hook_next_\hook`.

```

295 \str_if_eq:nnTF {#2} { top-level }
296 {
297   \str_if_eq:eeTF { top-level } { \__hook_currname_or_default: }
298   {

```

If the hook's basic structure does not exist, we need to declare it with `__hook_init_structure:n`.

```

299   \__hook_init_structure:n {#1}
300   \__hook_tl_gput_right:cn { __hook_toplevel~#1 } {#3}
301 }
302 { \msg_error:nnn { hooks } { misused-top-level } {#1} }
303 }
304 {
305   \prop_get:cnNTF { g__hook_#1_code_prop } {#2} \l__hook_return_tl
306   {
307     \prop_gput:cno { g__hook_#1_code_prop } {#2}
308     { \l__hook_return_tl #3 }
309   }
310   { \prop_gput:cnn { g__hook_#1_code_prop } {#2} {#3} }
311 }
312 }

```

(End definition for `\hook_gput_code:nnn`, `__hook_gput_code:nnn`, and `__hook_hook_gput_code_do:nnn`. This function is documented on page 176.)

`__hook_gput_undeclared_hook:nnn`

Often it may happen that a package *A* defines a hook `foo`, but package *B*, that adds code to that hook, is loaded before *A*. In such case we need to add code to the hook before its declared.

```

313 \cs_new_protected:Npn \__hook_gput_undeclared_hook:nnn #1 #2 #3
314 {
315   \__hook_init_structure:n {#1}
316   \__hook_hook_gput_code_do:nnn {#1} {#2} {#3}
317 }

```

(End definition for `__hook_gput_undeclared_hook:nnn`.)

`__hook_try_declaring_generic_hook:nnn`

These entry-level macros just pass the arguments along to the common `__hook_try_declaring_generic_hook:nNNnn` with the right functions to execute when some action is to be taken.

The wrapper `__hook_try_declaring_generic_hook:nnn` then defers `\hook_gput_code:nnn` if the generic hook was declared, or to `__hook_gput_undeclared_hook:nnn` otherwise (the hook was tested for existence before, so at this point if it isn't generic, it doesn't exist).

The wrapper `__hook_try_declaring_generic_next_hook:nn` for next-execution hooks does the same: it defers the code to `\hook_gput_next_code:nn` if the generic hook was declared, or to `__hook_gput_next_do:nn` otherwise.

```

318 \cs_new_protected:Npn \__hook_try_declaring_generic_hook:nnn #1
319 {
320   \__hook_try_declaring_generic_hook:nNNnn {#1}

```

```

321     \hook_gput_code:nnn \__hook_gput_undeclared_hook:nnn
322   }
323 \cs_new_protected:Npn \__hook_try_declaring_generic_next_hook:nn #1
324   {
325     \__hook_try_declaring_generic_hook:nNNnn {#1}
326     \hook_gput_next_code:nn \__hook_gput_next_do:nn
327   }

```

(End definition for __hook_try_declaring_generic_hook:nnn and
__hook_try_declaring_generic_next_hook:nn.)

__hook_try_declaring_generic_hook:nNNnn
hook_try_declaring_generic_hook_split:nNNnn

__hook_try_declaring_generic_hook:nNNnn now splits the hook name at the first / (if any) and first checks if it is a file-specific hook (they require some normalization) using __hook_if_file_hook:wTF. If not then check it is one of a predefined set for generic names. We also split off the second component to see if we have to make a reversed hook. In either case the function returns *<true>* for a generic hook and *<false>* in other cases.

```

328 \cs_new_protected:Npn \__hook_try_declaring_generic_hook:nNNnn #1
329   {
330     \__hook_if_file_hook:wTF #1 / / \s__hook_mark
331     {
332       \exp_args:Ne \__hook_try_declaring_generic_hook_split:nNNnn
333       { \exp_args:Ne \__hook_file_hook_normalize:n {#1} }
334     }
335     { \__hook_try_declaring_generic_hook_split:nNNnn {#1} }
336   }
337 \cs_new_protected:Npn \__hook_try_declaring_generic_hook_split:nNNnn #1 #2 #3
338   {
339     \__hook_try_declaring_generic_hook:wnTF #1 / / / \scan_stop: {#1}
340     { #2 }
341     { #3 } {#1}
342   }

```

(End definition for __hook_try_declaring_generic_hook:nNNnn and
__hook_try_declaring_generic_hook_split:nNNnn.)

__hook_try_declaring_generic_hook:wnTF

```

343 <latexrelease>\IncludeInRelease{2021/06/01}%
344 <latexrelease>          {\__hook_try_declaring_generic_hook:wn}{Support~cmd~hooks}
345 \prg_new_protected_conditional:Npnn \__hook_try_declaring_generic_hook:wn
346   #1 / #2 / #3 / #4 \scan_stop: #5 { TF }
347   {
348     \tl_if_empty:nTF {#2}
349     { \prg_return_false: }
350     {
351       \prop_if_in:NnTF \c__hook_generics_prop {#1}
352       {
353         \__hook_if_usable:nF {#5}
354       }

```

If the hook doesn't exist yet we check if it is a cmd hook and if so we attempt patching the command in addition to declaring the hook.

For some commands this will not be possible, in which case __hook_patch_cmd_or_delay:Nnn (defined in ltcmdhooks) will generate an appropriate error message.

```

355     \str_if_eq:nnT {#1} { cmd }
356     { \__hook_try_put_cmd_hook:n {#5} }

```

Declare the hook always even if it can't really be used (error message generated elsewhere).

Here we use `__hook_make_usable:n`, so that a `\hook_new:n` is still possible later.

```

357         \__hook_make_usable:n {#5}
358     }
359     \prop_if_in:NnTF \c__hook_generics_reversed_ii_prop {#2}
360     { \tl_gset:cn { g__hook_#5_reversed_tl } { - } }
361     {
362         \prop_if_in:NnT \c__hook_generics_reversed_iii_prop {#3}
363         { \tl_gset:cn { g__hook_#5_reversed_tl } { - } }
364     }
365     \prg_return_true:
366 }
367 { \prg_return_false: }
368 }
369 }
370 \<latexrelease>\EndIncludeInRelease

371 \<latexrelease>\IncludeInRelease{2020/10/01}%
372 \<latexrelease>         { \__hook_try_declaring_generic_hook:wn } { Support~cmd~hooks }
373 \<latexrelease>
374 \<latexrelease> \prg_new_protected_conditional:Npnn \__hook_try_declaring_generic_hook:wn
375 \<latexrelease>     #1 / #2 / #3 / #4 \scan_stop: #5 { TF }
376 \<latexrelease> {
377     \tl_if_empty:nTF {#2}
378     { \prg_return_false: }
379     {
380         \prop_if_in:NnTF \c__hook_generics_prop {#1}
381         {
382             \__hook_if_declared:nF {#5} { \hook_new:n {#5} }
383             \prop_if_in:NnTF \c__hook_generics_reversed_ii_prop {#2}
384             { \tl_gset:cn { g__hook_#5_reversed_tl } { - } }
385             {
386                 \prop_if_in:NnT \c__hook_generics_reversed_iii_prop {#3}
387                 { \tl_gset:cn { g__hook_#5_reversed_tl } { - } }
388             }
389             \prg_return_true:
390         }
391         { \prg_return_false: }
392     }
393 }
394 \<latexrelease> }
395 \<latexrelease>\EndIncludeInRelease

```

(End definition for `__hook_try_declaring_generic_hook:wnTF`.)

`__hook_if_file_hook_p:w` `__hook_if_file_hook:wTF` checks if the argument is a valid file-specific hook (not, for example, `file/before`, but `file/before/foo.tex`). If it is a file-specific hook, then it executes the `<true>` branch, otherwise `<false>`.

A file-specific hook is `file/<position>/<name>`. If any of these parts don't exist, it is a general file hook or not a file hook at all, so the conditional evaluates to `<false>`. Otherwise, it checks that the first part is `file` and that the `<position>` is in the `\c__hook_generics_file_prop`.

A property list is used here to avoid having to worry with catcodes, because `expl3`'s file name parsing turns all characters into catcode-12 tokens, which might differ from hand-input letters.

```

396 \prg_new_conditional:Npnn \__hook_if_file_hook:w
397   #1 / #2 / #3 \s__hook_mark { TF }
398   {
399     \str_if_eq:nnTF {#1} { file }
400     {
401       \bool_lazy_or:nnTF
402         { \tl_if_empty_p:n {#3} }
403         { \str_if_eq_p:nn {#3} { / } }
404         { \prg_return_false: }
405         {
406           \prop_if_in:NnTF \c__hook_generics_file_prop {#2}
407             { \prg_return_true: }
408             { \prg_return_false: }
409         }
410     }
411     { \prg_return_false: }
412   }

```

(End definition for `__hook_if_file_hook:wTF`.)

`__hook_file_hook_normalize:n` When a file-specific hook is found, before being declared it is lightly normalized by `__hook_file_hook_normalize:n`. The current implementation just replaces two consecutive slashes (//) by a single one, to cope with simple cases where the user did something like `\def\input@path{./mypath/}`, in which case a hook would have to be `\AddToHook{file/after/./mypath//file.tex}`.

```

413 \cs_new:Npn \__hook_file_hook_normalize:n #1
414   { \__hook_strip_double_slash:n {#1} }
415 \cs_new:Npn \__hook_strip_double_slash:n #1
416   { \__hook_strip_double_slash:w #1 // \s__hook_mark }

```

This function is always called after testing if the argument is a file hook with `__hook_if_file_hook:wTF`, so we can assume it has three parts (it is either `file/before/...` or `file/after/...`), so we use `#1/#2/#3 //` instead of just `#1 //` to prevent losing a slash if the file name is empty.

```

417 \cs_new:Npn \__hook_strip_double_slash:w #1/#2/#3 // #4 \s__hook_mark
418   {
419     \tl_if_empty:nTF {#4}
420       { #1/#2/#3 }
421       { \__hook_strip_double_slash:w #1/#2/#3 / #4 \s__hook_mark }
422   }

```

(End definition for `__hook_file_hook_normalize:n`, `__hook_strip_double_slash:n`, and `__hook_strip_double_slash:w`.)

`\c__hook_generics_prop` Property list holding the generic names. We don't provide any user interface to this as this is meant to be static.

cmd The generic hooks used for commands.

env The generic hooks used in `\begin` and `\end`.

file, package, class, include The generic hooks used when loading a file

```

423 \prop_const_from_keyval:Nn \c__hook_generics_prop
424 {cmd=,env=,file=,package=,class=,include=}

```

(End definition for \c__hook_generics_prop.)

```

\c__hook_generics_reversed_ii_prop
\c__hook_generics_reversed_iii_prop
\c__hook_generics_file_prop

```

Some of the generic hooks are supposed to use reverse ordering, these are the following (only the second or third sub-component is checked):

```

425 \prop_const_from_keyval:Nn \c__hook_generics_reversed_ii_prop {after=,end=}
426 \prop_const_from_keyval:Nn \c__hook_generics_reversed_iii_prop {after=}
427 \prop_const_from_keyval:Nn \c__hook_generics_file_prop {before=,after=}

```

(End definition for \c__hook_generics_reversed_ii_prop, \c__hook_generics_reversed_iii_prop, and \c__hook_generics_file_prop.)

```

\hook_gremove_code:nn
\__hook_gremove_code:nn

```

With \hook_gremove_code:nn{<hook>}{<label>} any code for <hook> stored under <label> is removed.

```

428 \cs_new_protected:Npn \hook_gremove_code:nn #1 #2
429 { \__hook_normalize_hook_args:Nnn \__hook_gremove_code:nn {#1} {#2} }
430 \cs_new_protected:Npn \__hook_gremove_code:nn #1 #2
431 {

```

First check that the hook code pool exists. __hook_if_usable:nTF isn't used here because it should be possible to remove code from a hook before its defined (see section 2.1.8).

```

432 \__hook_if_structure_exist:nTF {#1}
433 {

```

Then remove the chunk and run __hook_update_hook_code:n so that the execution token list reflects the change if we are after \begin{document}.

If all code is to be removed, clear the code pool \g__hook_<hook>_code_prop, the top-level code __hook_toplevel_<hook>, and the next-execution code __hook_next_<hook>.

```

434 \str_if_eq:nnTF {#2} {*}
435 {
436 \prop_gclear:c { g__hook_#1_code_prop }
437 \__hook_tl_gclear:c { __hook_toplevel~#1 }
438 \__hook_tl_gclear:c { __hook_next~#1 }
439 }
440 {

```

If the label is top-level then clear the token list, as all code there is under the same label. Marked removal is not implemented for top-level because it is hard to reliably know that no code was added to __hook_toplevel_<hook> (granted that an empty code could be interpreted as that, but then it differs in behaviour from other labels, in which an empty chunk is still valid for removal). Besides, it doesn't make much (if any) sense for packages to remove top-level code. So here the chunk is just cleared unconditionally.

```

441 \str_if_eq:nnTF {#2} { top-level }
442 { \__hook_tl_gclear:c { __hook_toplevel~#1 } }
443 {

```

Otherwise check if the label being removed exists in the code pool. If it does, just call __hook_gremove_code_do:nn to do the removal, otherwise mark it to be removed.

```

444 \prop_get:cnNTF { g__hook_#1_code_prop } {#2} \l__hook_return_tl
445 { \__hook_gremove_code_do:nn }
446 { \__hook_mark_removal:nn }

```

```

447         {#1} {#2}
448     }
449 }

```

Finally update the code, if the hook exists.

```

450     \__hook_if_usable:nT {#1}
451     { \__hook_update_hook_code:n {#1} }
452 }

```

If the code pool for this hook doesn't exist it means that nothing tried to add to it before, so we just queue this removal order for later.

```

453     { \__hook_mark_removal:nn {#1} {#2} }
454 }

```

Remove code for a given label.

```

455 \cs_new_protected:Npn \__hook_gremove_code_do:nn #1 #2
456 { \prop_gremove:cn { g__hook_#1_code_prop } {#2} }

```

(End definition for `\hook_gremove_code:nn`, `__hook_gremove_code:nn`, and `__hook_gremove_code_do:nn`. This function is documented on page 176.)

`__hook_mark_removal:nn` Marks $\langle label \rangle$ (#2) to be removed from $\langle hook \rangle$ (#1). The number of removals should be fairly small, and `\tl_gremove_once:Nx` is fairly efficient even for longer token lists, so we use a single global token list, rather than one for each hook.

A hand-crafted token list is used here because property lists don't hold repeated items, so multiple usages of `__hook_mark_removal:nn` would be cancelled by a single `__hook_unmark_removal:nn`.

```

457 \cs_new_protected:Npn \__hook_mark_removal:nn #1 #2
458 {
459     \tl_gput_right:Nx \g__hook_removal_list_tl
460     { \__hook_removal_tl:nn {#1} {#2} }
461 }

```

(End definition for `__hook_mark_removal:nn`.)

`__hook_unmark_removal:nn` Unmarks $\langle label \rangle$ (#2) to be removed from $\langle hook \rangle$ (#1). `\tl_gremove_once:Nx` is used rather than `\tl_gremove_all:Nx` so that two additions are needed to cancel two marked removals, rather than only one.

```

462 \cs_new_protected:Npn \__hook_unmark_removal:nn #1 #2
463 {
464     \tl_gremove_once:Nx \g__hook_removal_list_tl
465     { \__hook_removal_tl:nn {#1} {#2} }
466 }

```

(End definition for `__hook_unmark_removal:nn`.)

`__hook_if_marked_removal:nnTF` Checks if the `\g__hook_removal_list_tl` contains the current $\langle label \rangle$ (#2) and $\langle hook \rangle$ (#1).

```

467 \prg_new_protected_conditional:Npnn \__hook_if_marked_removal:nn #1 #2 { TF }
468 {
469     \exp_args:NNx \tl_if_in:NnTF \g__hook_removal_list_tl
470     { \__hook_removal_tl:nn {#1} {#2} }
471     { \prg_return_true: } { \prg_return_false: }
472 }

```

(End definition for `_hook_if_marked_removal:nnTF`.)

`_hook_removal_tl:nn` Builds a token list with #1 and #2 which can only be matched by #1 and #2. The `&`₄ anchors a removal, so that #1 can't be mistaken by #2 and vice versa, and the two `$`₃ delimit the two arguments

```
473 \cs_new:Npn \_hook_removal_tl:nn #1 #2
474   { & \tl_to_str:n {#2} $ \tl_to_str:n {#1} $ }
```

(End definition for `_hook_removal_tl:nn`.)

`\g__hook_??_code_prop` Initially these variables simply used an empty “label” name (not two question marks).
`_hook~??` This was a bit unfortunate, because then l3doc complains about `__` in the middle of a
`\g__hook_??_reversed_tl` command name when trying to typeset the documentation. However using a “normal” name such as `default` has the disadvantage of that being not really distinguishable from a real hook name. I now have settled for `??` which needs some gymnastics to get it into the `csname`, but since this is used a lot, the code should be fast, so this is not done with `c` expansion in the code later on.

`_hook_` isn't used, but it has to be defined to trick the code into thinking that `??` is actually a hook.

```
475 \prop_new:c {g__hook_??_code_prop}
476 \prop_new:c {_hook~??}
```

Default rules are always given in normal ordering (never in reversed ordering). If such a rule is applied to a reversed hook it behaves as if the rule is reversed (e.g., **after** becomes **before**) because those rules are applied first and then the order is reversed.

```
477 \tl_new:c {g__hook_??_reversed_tl}
```

(End definition for `\g__hook_??_code_prop`, `_hook~??`, and `\g__hook_??_reversed_tl`.)

3.7 Setting rules for hooks code

`\hook_gset_rule:nnnn` With `\hook_gset_rule:nnnn{<hook>}{<label1>}{<relation>}{<label2>}` a relation is defined between the two code labels for the given `<hook>`. The special hook `??` stands for *any* hook, which sets a default rule (to be used if no other relation between the two hooks exist).

```
478 \cs_new_protected:Npn \hook_gset_rule:nnnn #1#2#3#4
479   {
480     \_hook_normalize_hook_rule_args:Nnnnn \_hook_gset_rule:nnnn
481       {#1} {#2} {#3} {#4}
482   }
```

```
483 \cs_new_protected:Npn \_hook_gset_rule:nnnn #1#2#3#4
484   {
```

First we ensure the basic data structure of the hook exists:

```
485   \_hook_init_structure:n {#1}
```

Then we clear any previous relationship between both labels.

```
486   \_hook_rule_gclear:nnn {#1} {#2} {#4}
```

Then we call the function to handle the given rule. Throw an error if the rule is invalid.

```

487 \cs_if_exist_use:CTF { __hook_rule_#3_gset:nnn }
488 {
489     {#1} {#2} {#4}
490     \__hook_update_hook_code:n {#1}
491 }
492 { \msg_error:nnnnnn { hooks } { unknown-rule }
493     {#1} {#2} {#3} {#4} }
494 }

```

(End definition for `\hook_gset_rule:nnnn` and `__hook_gset_rule:nnnn`. This function is documented on page 176.)

```

\__hook_rule_before_gset:nnn
\__hook_rule_after_gset:nnn
\__hook_rule_<_gset:nnn
\__hook_rule_>_gset:nnn

```

Then we add the new rule. We need to normalize the rules here to allow for faster processing later. Given a pair of labels l_A and l_B , the rule $l_A > l_B$ is the same as $l_B < l_A$ only presented differently. But by normalizing the forms of the rule to a single representation, say, $l_B < l_A$, reduces the time spent looking for the rules later considerably.

Here we do that normalization by using `\(pdf)strcmp` to lexically sort labels l_A and l_B to a fixed order. This order is then enforced every time these two labels are used together.

Here we use `__hook_label_pair:nn {<hook>} {l_A} {l_B}` to build a string $l_B|l_A$ with a fixed order, and use `__hook_label_ordered:nnTF` to apply the correct rule to the pair of labels, depending if it was sorted or not.

```

495 \cs_new_protected:Npn \__hook_rule_before_gset:nnn #1#2#3
496 {
497     \__hook_tl_gset:cx { g__hook_#1_rule_ \__hook_label_pair:nn {#2} {#3} _tl }
498     { \__hook_label_ordered:nnTF {#2} {#3} { < } { > } }
499 }
500 \cs_new_eq:cN { __hook_rule_<_gset:nnn } \__hook_rule_before_gset:nnn
501 \cs_new_protected:Npn \__hook_rule_after_gset:nnn #1#2#3
502 {
503     \__hook_tl_gset:cx { g__hook_#1_rule_ \__hook_label_pair:nn {#3} {#2} _tl }
504     { \__hook_label_ordered:nnTF {#3} {#2} { < } { > } }
505 }
506 \cs_new_eq:cN { __hook_rule_>_gset:nnn } \__hook_rule_after_gset:nnn

```

(End definition for `__hook_rule_before_gset:nnn` and others.)

```
\__hook_rule_voids_gset:nnn
```

This rule removes (clears, actually) the code from label #3 if label #2 is in the hook #1.

```

507 \cs_new_protected:Npn \__hook_rule_voids_gset:nnn #1#2#3
508 {
509     \__hook_tl_gset:cx { g__hook_#1_rule_ \__hook_label_pair:nn {#2} {#3} _tl }
510     { \__hook_label_ordered:nnTF {#2} {#3} { -> } { <- } }
511 }

```

(End definition for `__hook_rule_voids_gset:nnn`.)

```

\__hook_rule_incompatible-error_gset:nnn
\__hook_rule_incompatible-warning_gset:nnn

```

These relations make an error/warning if labels #2 and #3 appear together in hook #1.

```

512 \cs_new_protected:cpn { __hook_rule_incompatible-error_gset:nnn } #1#2#3
513 { \__hook_tl_gset:cn { g__hook_#1_rule_ \__hook_label_pair:nn {#2} {#3} _tl }
514     { xE } }
515 \cs_new_protected:cpn { __hook_rule_incompatible-warning_gset:nnn } #1#2#3
516 { \__hook_tl_gset:cn { g__hook_#1_rule_ \__hook_label_pair:nn {#2} {#3} _tl }
517     { xW } }

```

(End definition for `_hook_rule_incompatible-error_gset:nnn` and
`_hook_rule_incompatible-warning_gset:nnn`.)

`_hook_rule_unrelated_gset:nnn` Undo a setting. `_hook_rule_unrelated_gset:nnn` doesn't need to do anything, since we use `_hook_rule_gclear:nnn` before setting any rule.

```
518 \cs_new_protected:Npn \_hook_rule_unrelated_gset:nnn #1#2#3 { }
519 \cs_new_protected:Npn \_hook_rule_gclear:nnn #1#2#3
520   { \cs_undefine:c { g\_hook\_#1\_rule\_ \_hook\_label\_pair:nn {#2} {#3} \_tl } }
```

(End definition for `_hook_rule_unrelated_gset:nnn` and `_hook_rule_gclear:nnn`.)

`_hook_label_pair:nn` Ensure that the lexically greater label comes first.

```
521 \cs_new:Npn \_hook\_label\_pair:nn #1#2
522   {
523     \if_case:w \_hook\_str\_compare:nn {#1} {#2} \exp\_stop\_f:
524       #1 | #1 % 0
525     \or:   #1 | #2 % +1
526     \else: #2 | #1 % -1
527     \fi:
528   }
```

(End definition for `_hook_label_pair:nn`.)

`_hook_label_ordered_p:nn` Check that labels #1 and #2 are in the correct order (as returned by `_hook_label_pair:nn`) and if so return true, else return false.

`_hook_label_ordered:nnTF`

```
529 \prg_new_conditional:Npnn \_hook\_label\_ordered:nn #1#2 { TF }
530   {
531     \if_int_compare:w \_hook\_str\_compare:nn {#1} {#2} > 0 \exp\_stop\_f:
532     \prg_return_true:
533     \else:
534     \prg_return_false:
535     \fi:
536   }
```

(End definition for `_hook_label_ordered:nnTF`.)

`_hook_if_label_case:nnnnn` To avoid doing the string comparison twice in `_hook_initialize_single:NNn` (once with `\str_if_eq:nn` and again with `_hook_label_ordered:nn`), we use a three-way branching macro that will compare #1 and #2 and expand to `\use_i:nnn` if they are equal, `\use_ii:nn` if #1 is lexically greater, and `\use_iii:nn` otherwise.

```
537 \cs_new:Npn \_hook\_if\_label\_case:nnnnn #1#2
538   {
539     \cs:w use\_
540     \if_case:w \_hook\_str\_compare:nn {#1} {#2}
541       i \or: ii \else: iii \fi: :nnn
542     \cs_end:
543   }
```

(End definition for `_hook_if_label_case:nnnnn`.)

`_hook_update_hook_code:n` Before `\begin{document}` this does nothing, in the body it reinitializes the hook code using the altered data.

```
544 \cs_new_eq:NN \_hook\_update\_hook\_code:n \use_none:n
```

(End definition for `_hook_update_hook_code:n`.)

`_hook_initialize_all:` Initialize all known hooks (at `\begin{document}`), i.e., update the fast execution token lists to hold the necessary code in the right order.

```
545 \cs_new_protected:Npn \_hook_initialize_all: {
```

First we change `_hook_update_hook_code:n` which so far was a no-op to now initialize one hook. This way any later updates to the hook will run that code and also update the execution token list.

```
546 \cs_gset_eq:NN \_hook_update_hook_code:n \_hook_initialize_hook_code:n
```

Now we loop over all hooks that have been defined and update each of them.

```
547 \_hook_debug:n { \prop_gc_clear:N \g__hook_used_prop }
548 \seq_map_inline:Nn \g__hook_all_seq
549 {
550 \_hook_update_hook_code:n {##1}
551 }
```

If we are debugging we show results hook by hook for all hooks that have data.

```
552 \_hook_debug:n
553 { \iow_term:x{^^JAll~ initialized~ (non-empty)~ hooks:}
554 \prop_map_inline:Nn \g__hook_used_prop
555 { \iow_term:x{^^J~ ##1~ ->~
556 \exp_not:v {\_hook~##1}~ }
557 }
558 }
```

After all hooks are initialized we change the “use” to just call the hook code and not initialize it (as it was done in the preamble.

```
559 \cs_gset_eq:NN \hook_use:n \_hook_use_initialized:n
560 \cs_gset_eq:NN \_hook_preamble_hook:n \use_none:n
561 }
```

(End definition for `_hook_initialize_all:.`)

`_hook_initialize_hook_code:n` Initializing or reinitializing the fast execution hook code. In the preamble this is selectively done in case a hook gets used and at `\begin{document}` this is done for all hooks and afterwards only if the hook code changes.

```
562 \cs_new_protected:Npn \_hook_initialize_hook_code:n #1
563 {
564 \_hook_debug:n{ \iow_term:x{^^JUupdate~ code~ for~ hook~
565 '##1' \on@line :^^J} }
```

This does the sorting and the updates. First thing we do is to check if a legacy hook macro exists and if so we add it to the hook under the label `legacy`. This might make the hook non-empty so we have to do this before the then following test.

```
566 \_hook_include_legacy_code_chunk:n {##1}
```

If there aren't any code chunks for the current hook, there is no point in even starting the sorting routine so we make a quick test for that and in that case just update `_hook_⟨hook⟩` to hold the `top-level` and `next` code chunks. If there are code chunks we call `_hook_initialize_single:NNn` and pass to it ready made csnames as they are needed several times inside. This way we save a bit on processing time if we do that up front.

```
567 \_hook_if_usable:nT {##1}
568 {
569 \prop_if_empty:cTF {g__hook_#1_code_prop}
```

```

570     {
571       \__hook_tl_gset:co { __hook~#1 }
572       {
573         \cs:w __hook_toplevel~#1 \exp_after:wN \cs_end:
574         \cs:w __hook_next~#1 \cs_end:
575       }
576     }
577   {

```

By default the algorithm sorts the code chunks and then saves the result in a token list for fast execution by adding the code one after another using `\tl_gput_right:NV`. When we sort code for a reversed hook, all we have to do is to add the code chunks in the opposite order into the token list. So all we have to do in preparation is to change two definitions used later on.

```

578     \__hook_if_reversed:nTF {#1}
579     { \cs_set_eq:NN \__hook_tl_gput:Nn \__hook_tl_gput_left:Nn
580       \cs_set_eq:NN \__hook_clist_gput:NV \clist_gput_left:NV }
581     { \cs_set_eq:NN \__hook_tl_gput:Nn \__hook_tl_gput_right:Nn
582       \cs_set_eq:NN \__hook_clist_gput:NV \clist_gput_right:NV }

```

When sorting, some relations (namely voids) need to act destructively on the code property lists to remove code that shouldn't appear in the sorted hook token list, so we temporarily save the old code property list so that it can be restored later.

```

583     \prop_set_eq:Nc \l__hook_work_prop { g__hook_#1_code_prop }
584     \__hook_initialize_single:ccn
585     { __hook~#1 } { g__hook_#1_labels_clist } {#1}

```

For debug display we want to keep track of those hooks that actually got code added to them, so we record that in plist. We use a plist to ensure that we record each hook name only once, i.e., we are only interested in storing the keys and the value is arbitrary.

```

586     \__hook_debug:n{ \exp_args:NNx \prop_gput:Nnn
587                     \g__hook_used_prop {#1}{ } }
588   }
589 }
590 }

```

(End definition for `__hook_initialize_hook_code:n`.)

`__hook_tl_csname:n` It is faster to pass a single token and expand it when necessary than to pass a bunch of character tokens around.

FMi: note to myself: verify

```

591 \cs_new:Npn \__hook_tl_csname:n #1 { l__hook_label_#1_tl }
592 \cs_new:Npn \__hook_seq_csname:n #1 { l__hook_label_#1_seq }

```

(End definition for `__hook_tl_csname:n` and `__hook_seq_csname:n`.)

`\l__hook_labels_seq` For the sorting I am basically implementing Knuth's algorithm for topological sorting as given in TAOCP volume 1 pages 263–266. For this algorithm we need a number of local variables:

`\l__hook_labels_int`
`\l__hook_front_tl`
`\l__hook_rear_tl`
`\l__hook_label_0_tl`

- List of labels used in the current hook to label code chunks:

```

593 \seq_new:N \l__hook_labels_seq

```


- Number of labels used in the current hook. In Knuth’s algorithm this is called N :

```
594 \int_new:N \l__hook_labels_int
```

- The sorted code list to be build is managed using two pointers one to the front of the queue and one to the rear. We model this using token list pointers. Knuth calls them F and R :

```
595 \tl_new:N \l__hook_front_tl
596 \tl_new:N \l__hook_rear_tl
```

- The data for the start of the queue is kept in this token list, it corresponds to what Don calls `QLINK[0]` but since we aren’t manipulating individual words in memory it is slightly differently done:

```
597 \tl_new:c { \__hook_tl_csname:n { 0 } }
```

(End definition for `\l__hook_labels_seq` and others.)

```
\__hook_initialize_single:NNn
\__hook_initialize_single:ccn
```

`__hook_initialize_single:NNn` implements the sorting of the code chunks for a hook and saves the result in the token list for fast execution (#4). The arguments are $\langle hook-code-plist \rangle$, $\langle hook-code-tl \rangle$, $\langle hook-top-level-code-tl \rangle$, $\langle hook-next-code-tl \rangle$, $\langle hook-ordered-labels-clist \rangle$ and $\langle hook-name \rangle$ (the latter is only used for debugging—the $\langle hook-rule-plist \rangle$ is accessed using the $\langle hook-name \rangle$).

The additional complexity compared to Don’s algorithm is that we do not use simple positive integers but have arbitrary alphanumeric labels. As usual Don’s data structures are chosen in a way that one can omit a lot of tests and I have mimicked that as far as possible. The result is a restriction I do not test for at the moment: a label can’t be equal to the number 0!

FMi: Needs checking for, just in case ... maybe

```
598 \cs_new_protected:Npn \__hook_initialize_single:NNn #1#2#3
599 {
```

Step T1: Initialize the data structure ...

```
600 \seq_clear:N \l__hook_labels_seq
601 \int_zero:N \l__hook_labels_int
```

Store the name of the hook:

```
602 \tl_set:Nn \l__hook_cur_hook_tl {#3}
```

We loop over the property list holding the code and record all labels listed there. Only rules for those labels are of interest to us. While we are at it we count them (which gives us the N in Knuth’s algorithm. The prefix `label_` is added to the variables to ensure that labels named `front`, `rear`, `labels`, or `return` don’t interact with our code.

```
603 \prop_map_inline:Nn \l__hook_work_prop
604 {
605   \int_incr:N \l__hook_labels_int
606   \seq_put_right:Nn \l__hook_labels_seq {##1}
607   \__hook_tl_set:cn { \__hook_tl_csname:n {##1} } { 0 }
608   \seq_clear_new:c { \__hook_seq_csname:n {##1} }
609 }
```

Steps T2 and T3: Sort the relevant rules into the data structure...

This loop constitutes a square matrix of the labels in `\l__hook_work_prop` in the vertical and the horizontal directions. However since the rule $l_A \langle rel \rangle l_B$ is the same as $l_B \langle rel \rangle^{-1} l_A$ we can cut the loop short at the diagonal of the matrix (*i.e.*, when both labels are equal), saving a good amount of time. The way the rules were set up (see the implementation of `__hook_rule_before_gset:nnn` above) ensures that we have no rule in the ignored side of the matrix, and all rules are seen. The rules are applied in `__hook_apply_label_pair:nnn`, which takes the properly-ordered pair of labels as argument.

```

610 \prop_map_inline:Nn \l__hook_work_prop
611 {
612   \prop_map_inline:Nn \l__hook_work_prop
613   {
614     \__hook_if_label_case:nnnnn {##1} {####1}
615     { \prop_map_break: }
616     { \__hook_apply_label_pair:nnn {##1} {####1} }
617     { \__hook_apply_label_pair:nnn {####1} {##1} }
618     {#3}
619   }
620 }
```

Take a breath and take a look at the data structures that have been set up:

```

621 \__hook_debug:n { \__hook_debug_label_data:N \l__hook_work_prop }
```

Step T4:

```

622 \tl_set:Nn \l__hook_rear_tl { 0 }
623 \tl_set:cn { \__hook_tl_csname:n { 0 } } { 0 }
624 \seq_map_inline:Nn \l__hook_labels_seq
625 {
626   \int_compare:nNnT { \cs:w \__hook_tl_csname:n {##1} \cs_end: } = 0
627   {
628     \tl_set:cn { \__hook_tl_csname:n { \l__hook_rear_tl } } {##1}
629     \tl_set:Nn \l__hook_rear_tl {##1}
630   }
631 }
632 \tl_set_eq:Nc \l__hook_front_tl { \__hook_tl_csname:n { 0 } }
633 \__hook_tl_gclear:N #1
634 \clist_gclear:N #2
```

The whole loop combines steps T5–T7:

```

635 \bool_while_do:nn { ! \str_if_eq_p:Vn \l__hook_front_tl { 0 } }
636 {
```

This part is step T5:

```

637   \int_decr:N \l__hook_labels_int
638   \prop_get:NVN \l__hook_work_prop \l__hook_front_tl \l__hook_return_tl
639   \exp_args:NNV \__hook_tl_gput:Nn #1 \l__hook_return_tl
640   \__hook_clist_gput:NV #2 \l__hook_front_tl
641   \__hook_debug:n{ \iow_term:x{Handled~ code~ for~ \l__hook_front_tl} }
```

This is step T6 except that we don't use a pointer P to move through the successors, but instead use `##1` of the mapping function.

```

642   \seq_map_inline:cn { \__hook_seq_csname:n { \l__hook_front_tl } }
643   {
```

```

644         \tl_set:cx { \__hook_tl_csname:n {##1} }
645         { \int_eval:n
646           { \cs:w \__hook_tl_csname:n {##1} \cs_end: - 1 }
647         }
648       \int_compare:nNnT
649         { \cs:w \__hook_tl_csname:n {##1} \cs_end: } = 0
650       {
651         \tl_set:cn { \__hook_tl_csname:n { \l__hook_rear_tl } } {##1}
652         \tl_set:Nn \l__hook_rear_tl {##1}
653       }
654     }

```

and step T7:

```

655     \tl_set_eq:Nc \l__hook_front_tl
656     { \__hook_tl_csname:n { \l__hook_front_tl } }

```

This is step T8: If we haven't moved the code for all labels (i.e., if `\l__hook_labels_int` is still greater than zero) we have a loop and our partial order can't be flattened out.

```

657   }
658   \int_compare:nNnF \l__hook_labels_int = 0
659   {
660     \iow_term:x{=====}
661     \iow_term:x{Error:~ label~ rules~ are~ incompatible:}

```

FMi: improve output on a rainy day

This is not really the information one needs in the error case but will do for now ...

```

662     \__hook_debug_label_data:N \l__hook_work_prop
663     \iow_term:x{=====}
664   }

```

After we have added all hook code to #1 we finish it off with adding extra code for the `top-level` (#2) and for one time execution (#3). These should normally be empty. The `top-level` code is added with `__hook_tl_gput:Nn` as that might change for a reversed hook (then `top-level` is the very first code chunk added). The `next` code is always added last.

```

665     \exp_args:NNo \__hook_tl_gput:Nn #1 { \cs:w __hook_toplevel~#3 \cs_end: }
666     \__hook_tl_gput_right:No #1 { \cs:w __hook_next~#3 \cs_end: }
667   }
668   \cs_generate_variant:Nn \__hook_initialize_single:NNn { cc }

```

(End definition for `__hook_initialize_single:NNn`.)

```

\__hook_tl_gput:Nn
\__hook_clist_gput:NV

```

These append either on the right (normal hook) or on the left (reversed hook). This is setup up in `__hook_initialize_hook_code:n`, elsewhere their behavior is undefined.

```

669 \cs_new:Npn \__hook_tl_gput:Nn { \ERROR }
670 \cs_new:Npn \__hook_clist_gput:NV { \ERROR }

```

(End definition for `__hook_tl_gput:Nn` and `__hook_clist_gput:NV`.)

```

\__hook_apply_label_pair:nnn
\__hook_label_if_exist_apply:nnnF

```

This is the payload of steps T2 and T3 executed in the loop described above. This macro assumes #1 and #2 are ordered, which means that any rule pertaining the pair #1 and #2 is `\g__hook_⟨hook⟩_rule_#1|#2_tl`, and not `\g__hook_⟨hook⟩_rule_#2|#1_tl`. This also saves a great deal of time since we only need to check the order of the labels once.

The arguments here are $\langle label1 \rangle$, $\langle label2 \rangle$, $\langle hook \rangle$, and $\langle hook-code-plist \rangle$. We are about to apply the next rule and enter it into the data structure. `__hook_apply_label_pair:nnn` will just call `__hook_label_if_exist_apply:nnnF` for the $\langle hook \rangle$, and if no rule is found, also try the $\langle hook \rangle$ name ?? denoting a default hook rule.

`__hook_label_if_exist_apply:nnnF` will check if the rule exists for the given hook, and if so call `__hook_apply_rule:nnn`.

```
671 \cs_new_protected:Npn \__hook_apply_label_pair:nnn #1#2#3
672 {
```

Extra complication: as we use default rules and local hook specific rules we first have to check if there is a local rule and if that exist use it. Otherwise check if there is a default rule and use that.

```
673   \__hook_label_if_exist_apply:nnnF {#1} {#2} {#3}
674   {
```

If there is no hook-specific rule we check for a default one and use that if it exists.

```
675       \__hook_label_if_exist_apply:nnnF {#1} {#2} { ?? } { }
676   }
677 }
```

```
678 \cs_new_protected:Npn \__hook_label_if_exist_apply:nnnF #1#2#3
679 {
680   \if_cs_exist:w g__hook_ #3 _rule_ #1 | #2 _tl \cs_end:
```

What to do precisely depends on the type of rule we have encountered. If it is a `before` rule it will be handled by the algorithm but other types need to be managed differently. All this is done in `__hook_apply_rule:nnnN`.

```
681   \__hook_apply_rule:nnn {#1} {#2} {#3}
682   \exp_after:wN \use_none:n
683   \else:
684     \use:nn
685   \fi:
686 }
```

(End definition for `__hook_apply_label_pair:nnn` and `__hook_label_if_exist_apply:nnnF`.)

`__hook_apply_rule:nnn` This is the code executed in steps T2 and T3 while looping through the matrix This is part of step T3. We are about to apply the next rule and enter it into the data structure. The arguments are $\langle label1 \rangle$, $\langle label2 \rangle$, $\langle hook-name \rangle$, and $\langle hook-code-plist \rangle$.

```
687 \cs_new_protected:Npn \__hook_apply_rule:nnn #1#2#3
688 {
689   \cs:w __hook_apply_
690   \cs:w g__hook_#3_reversed_tl \cs_end: rule_
691   \cs:w g__hook_ #3 _rule_ #1 | #2 _tl \cs_end: :nnn \cs_end:
692   {#1} {#2} {#3}
693 }
```

(End definition for `__hook_apply_rule:nnn`.)

`__hook_apply_rule_<:nnn` The most common cases are `<` and `>` so we handle that first. They are relations \prec and \succ in TAOCP, and they dictate sorting.

```
\__hook_apply_rule_>:nnn
694 \cs_new_protected:cpn { __hook_apply_rule_<:nnn } #1#2#3
695 {
696   \__hook_debug:n { \__hook_msg_pair_found:nnn {#1} {#2} {#3} }
697   \tl_set:cx { \__hook_tl_csname:n {#2} }
```

```

698     { \int_eval:n{ \cs:w \__hook_tl_csname:n {#2} \cs_end: + 1 } }
699     \seq_put_right:cn{ \__hook_seq_csname:n {#1} }{#2}
700   }
701 \cs_new_protected:cpn { \__hook_apply_rule_>:nnn } #1#2#3
702 {
703   \__hook_debug:n { \__hook_msg_pair_found:nnn {#1} {#2} {#3} }
704   \tl_set:cx { \__hook_tl_csname:n {#1} }
705   { \int_eval:n{ \cs:w \__hook_tl_csname:n {#1} \cs_end: + 1 } }
706   \seq_put_right:cn{ \__hook_seq_csname:n {#2} }{#1}
707 }

```

(End definition for __hook_apply_rule_<:nnn and __hook_apply_rule_>:nnn.)

__hook_apply_rule_xE:nnn These relations make two labels incompatible within a hook. xE makes raises an error if
 __hook_apply_rule_xW:nnn the labels are found in the same hook, and xW makes it a warning.

```

708 \cs_new_protected:cpn { \__hook_apply_rule_xE:nnn } #1#2#3
709 {
710   \__hook_debug:n { \__hook_msg_pair_found:nnn {#1} {#2} {#3} }
711   \msg_error:nnnnnn { hooks } { labels-incompatible }
712   {#1} {#2} {#3} { 1 }
713   \use:c { \__hook_apply_rule_>:nnn } {#1} {#2} {#3}
714   \use:c { \__hook_apply_rule_<:nnn } {#1} {#2} {#3}
715 }
716 \cs_new_protected:cpn { \__hook_apply_rule_xW:nnn } #1#2#3
717 {
718   \__hook_debug:n { \__hook_msg_pair_found:nnn {#1} {#2} {#3} }
719   \msg_warning:nnnnnn { hooks } { labels-incompatible }
720   {#1} {#2} {#3} { 0 }
721 }

```

(End definition for __hook_apply_rule_xE:nnn and __hook_apply_rule_xW:nnn.)

__hook_apply_rule_>:nnn If we see -> we have to drop code for label #3 and carry on. We could do a little better
 __hook_apply_rule_<:nnn and drop everything for that label since it doesn't matter where we sort in the empty
 code. However that would complicate the algorithm a lot with little gain.⁹ So we still
 unnecessarily try to sort it in and depending on the rules that might result in a loop that
 is otherwise resolved. If that turns out to be a real issue, we can improve the code.

Here the code is removed from \l__hook_cur_hook_tl rather than #3 because the
 latter may be ??, and the default hook doesn't store any code. Removing from \l__-
 hook_cur_hook_tl makes default rules -> and <- work properly.

```

722 \cs_new_protected:cpn { \__hook_apply_rule_>:nnn } #1#2#3
723 {
724   \__hook_debug:n
725   {
726     \__hook_msg_pair_found:nnn {#1} {#2} {#3}
727     \iow_term:x{--->~ Drop~ '#2'~ code~ from~
728       \iow_char:N \ \ g__hook_ \l__hook_cur_hook_tl _code_prop ~
729       because~ of~ '#1' }
730   }
731   \prop_put:Nnn \l__hook_work_prop {#2} { }
732 }

```

⁹This also has the advantage that the result of the sorting doesn't change which might otherwise
 (for unrelated chunks) if we aren't careful.

```

733 \cs_new_protected:cpn { __hook_apply_rule_<:nnn } #1#2#3
734 {
735   \__hook_debug:n
736   {
737     \__hook_msg_pair_found:nnn {#1} {#2} {#3}
738     \iow_term:x{--->~ Drop~ '#1'~ code~ from~
739       \iow_char:N \ g__hook_ \l__hook_cur_hook_tl _code_prop ~
740       because~ of~ '#2' }
741   }
742   \prop_put:Nnn \l__hook_work_prop {#1} { }
743 }

```

(End definition for __hook_apply_rule_>:nnn and __hook_apply_rule_<:nnn.)

__hook_apply_rule_<:nnn Reversed rules.

```

\__hook_apply_rule_>:nnn 744 \cs_new_eq:cc { __hook_apply_rule_<:nnn } { __hook_apply_rule_>:nnn }
\__hook_apply_rule_<:nnn 745 \cs_new_eq:cc { __hook_apply_rule_>:nnn } { __hook_apply_rule_<:nnn }
\__hook_apply_rule_>:nnn 746 \cs_new_eq:cc { __hook_apply_rule_<:nnn } { __hook_apply_rule_>:nnn }
\__hook_apply_rule_x:nnn 747 \cs_new_eq:cc { __hook_apply_rule_>:nnn } { __hook_apply_rule_>:nnn }
748 \cs_new_eq:cc { __hook_apply_rule_xE:nnn } { __hook_apply_rule_xE:nnn }
749 \cs_new_eq:cc { __hook_apply_rule_xW:nnn } { __hook_apply_rule_xW:nnn }

```

(End definition for __hook_apply_rule_<:nnn and others.)

__hook_msg_pair_found:nnn A macro to avoid moving this many tokens around.

```

750 \cs_new_protected:Npn \__hook_msg_pair_found:nnn #1#2#3
751 {
752   \iow_term:x{~ \str_if_eq:nnTF {#3} {??} {default} {~normal} ~
753     rule~ \__hook_label_pair:nn {#1} {#2}:~
754     \use:c { g__hook_#3_rule_ \__hook_label_pair:nn {#1} {#2} _tl } ~
755     found}
756 }

```

(End definition for __hook_msg_pair_found:nnn.)

__hook_debug_label_data:N

```

757 \cs_new_protected:Npn \__hook_debug_label_data:N #1 {
758   \iow_term:x{Code~ labels~ for~ sorting:}
759   \iow_term:x{~ \seq_use:Nnnn\l__hook_labels_seq {~and~}{,~}{~and~} }
760   \iow_term:x{^^J Data~ structure~ for~ label~ rules:}
761   \prop_map_inline:Nn #1
762   {
763     \iow_term:x{~ ##1~ == \tl_use:c{ \__hook_tl_csname:n {##1} }~ ->~
764       \seq_use:cnnn{ \__hook_seq_csname:n {##1} }{-->~}{-->~}{-->~}
765     }
766   }
767   \iow_term:x{}
768 }

```

(End definition for __hook_debug_label_data:N.)

\hook_show:n

\hook_log:n

__hook_log_line:x

__hook_log_line_indent:x

__hook_log:nN

This writes out information about the hook given in its argument onto the .log file and the terminal, if \show_hook:n is used. Internally both share the same structure, except that at the end, \hook_show:n triggers T_EX's prompt.

```

769 \cs_new_protected:Npn \hook_log:n #1

```

```

770 {
771   \cs_set_eq:NN \__hook_log_cmd:x \iow_log:x
772   \__hook_normalize_hook_args:Nn \__hook_log:nN {#1} \tl_log:x
773 }
774 \cs_new_protected:Npn \hook_show:n #1
775 {
776   \cs_set_eq:NN \__hook_log_cmd:x \iow_term:x
777   \__hook_normalize_hook_args:Nn \__hook_log:nN {#1} \tl_show:x
778 }
779 \cs_new_protected:Npn \__hook_log_line:x #1
780 { \__hook_log_cmd:x { >~#1 } }
781 \cs_new_protected:Npn \__hook_log_line_indent:x #1
782 { \__hook_log_cmd:x { >~\@spaces #1 } }
783 \cs_new_protected:Npn \__hook_log:nN #1 #2
784 {
785   \__hook_preamble_hook:n {#1}
786   \__hook_log_cmd:x { ^^J ->~The~hook~'#1': }
787   \__hook_if_usable:nF {#1}
788   { \__hook_log_line:x { The~hook~is~not~declared. } }
789   \__hook_if_disabled:nT {#1}
790   { \__hook_log_line:x { The~hook~is~disabled. } }
791   \hook_if_empty:nTF {#1}
792   { #2 { The~hook~is~empty } }
793   {
794     \__hook_log_line:x { Code~chunks: }
795     \prop_if_empty:cTF { g__hook_#1_code_prop }
796     { \__hook_log_line_indent:x { --- } }
797     {
798       \prop_map_inline:cn { g__hook_#1_code_prop }
799       { \__hook_log_line_indent:x { ##1~-->~\tl_to_str:n {##2} } }
800     }
801   }

```

If there is code in the top-level token list, print it:

```

801   \__hook_log_line:x
802   {
803     Document-level~(top-level)~code
804     \__hook_if_usable:nT {#1}
805     { ~(executed~\__hook_if_reversed:nTF {#1} {first} {last} ) } :
806   }
807   \__hook_log_line_indent:x
808   {
809     \tl_if_empty:cTF { __hook_toplevel~#1 }
810     { --- }
811     { -> ~ \exp_args:Nv \tl_to_str:n { __hook_toplevel~#1 } }
812   }
813   \__hook_log_line:x { Extra~code~for~next~invocation: }
814   \__hook_log_line_indent:x
815   {
816     \tl_if_empty:cTF { __hook_next~#1 }
817     { --- }

```

If the token list is not empty we want to display it but without the first tokens (the code to clear itself) so we call a helper command to get rid of them.

```

818         { ->~ \exp_args:Nv \__hook_log_next_code:n { __hook_next~#1 } }
819     }

```

Loop through the rules in a hook and for every rule found, print it. If no rule is there, print ---. The boolean \l__hook_tmpa_bool here indicates if the hook has no rules.

```

820     \__hook_log_line:x { Rules: }
821     \bool_set_true:N \l__hook_tmpa_bool
822     \__hook_list_rules:nn {#1}
823     {
824         \bool_set_false:N \l__hook_tmpa_bool
825         \__hook_log_line_indent:x
826         {
827             ##2~ with~
828             \str_if_eq:nnT {##3} {??} { default~ }
829             relation~ ##1
830         }
831     }
832     \bool_if:NT \l__hook_tmpa_bool
833     { \__hook_log_line_indent:x { --- } }

```

When the hook is declared (that is, the sorting algorithm is applied to that hook) and not empty

```

834     \bool_lazy_and:nnTF
835     { \__hook_if_usable_p:n {#1} }
836     { ! \hook_if_empty_p:n {#1} }
837     {
838         \__hook_log_line:x
839         {
840             Execution~order
841             \bool_if:NTF \l__hook_tmpa_bool
842             { \__hook_if_reversed:nT {#1} { ~(after~reversal) } }
843             { ~(after~
844                 \__hook_if_reversed:nT {#1} { reversal~and~
845                 applying~rules)
846             } :
847         }
848         #2 % \tl_show:n
849         {
850             \@spaces
851             \clist_if_empty:cTF { g__hook_#1_labels_clist }
852             { --- }
853             { \clist_use:cn {g__hook_#1_labels_clist} { ,~ } }
854         }
855     }
856     {
857         \__hook_log_line:x { Execution~order: }
858         #2
859         {
860             \@spaces Not~set~because~the~hook~ \__hook_if_usable:nTF {#1}
861             { code~pool~is~empty }
862             { is~\__hook_if_disabled:nTF {#1} {disabled} {undeclared} }
863         }
864     }
865 }

```



```
866 }
```

To display the code for next invocation only (i.e., from `\AddToHookNext` we have to remove the first two tokens at the front which are `\tl_gclear:N` and the token list to clear.

```
867 \cs_new:Npn \__hook_log_next_code:n #1
868 { \exp_args:No \tl_to_str:n { \use_none:nn #1 } }
```

```
\__hook_log_next_code:n
```

(End definition for `\hook_show:n` and others. These functions are documented on page 177.)

```
\__hook_list_rules:nn
\__hook_list_one_rule:nnn
\__hook_list_if_rule_exists:nnnF
```

This macro takes a *<hook>* and an *<inline function>* and loops through each pair of *<labels>* in the *<hook>*, and if there is a relation between this pair of *<labels>*, the *<inline function>* is executed with `#1 = <relation>`, `#2 = <label1>|<label2>`, and `#3 = <hook>` (the latter may be the argument `#1` to `__hook_list_rules:nn`, or `??` if it is a default rule).

```
869 \cs_new_protected:Npn \__hook_list_rules:nn #1 #2
870 {
871   \cs_set_protected:Npn \__hook_tmp:w ##1 ##2 ##3 {#2}
872   \prop_map_inline:cn { g__hook_#1_code_prop }
873   {
874     \prop_map_inline:cn { g__hook_#1_code_prop }
875     {
876       \__hook_if_label_case:nnnnn {##1} {####1}
877       { \prop_map_break: }
878       { \__hook_list_one_rule:nnn {##1} {####1} }
879       { \__hook_list_one_rule:nnn {####1} {##1} }
880       {#1}
881     }
882   }
883 }
```

These two are quite similar to `__hook_apply_label_pair:nnn` and `__hook_label_if_exist_apply:nnnF`, respectively, but rather than applying the rule, they pass it to the *<inline function>*.

```
884 \cs_new_protected:Npn \__hook_list_one_rule:nnn #1#2#3
885 {
886   \__hook_list_if_rule_exists:nnnF {#1} {#2} {#3}
887   { \__hook_list_if_rule_exists:nnnF {#1} {#2} { ?? } { } }
888 }
889 \cs_new_protected:Npn \__hook_list_if_rule_exists:nnnF #1#2#3
890 {
891   \if_cs_exist:w g__hook_ #3_rule_ #1 | #2_tl \cs_end:
892   \exp_args:Nv \__hook_tmp:w
893   { g__hook_ #3_rule_ #1 | #2_tl } { #1 | #2 } {#3}
894   \exp_after:wN \use_none:nn
895   \fi:
896   \use:n
897 }
```

(End definition for `__hook_list_rules:nn`, `__hook_list_one_rule:nnn`, and `__hook_list_if_rule_exists:nnnF`.)

```
\__hook_debug_print_rules:n
```

A shorthand for debugging that prints similar to `\prop_show:N`.

```
898 \cs_new_protected:Npn \__hook_debug_print_rules:n #1
899 {
```

```

900 \iow_term:n { The~hook~#1~contains~the~rules: }
901 \cs_set_protected:Npn \__hook_tmp:w ##1
902 {
903   \__hook_list_rules:nn {#1}
904   {
905     \iow_term:x
906     {
907       > ##1 {####2} ##1 => ##1 {####1}
908       \str_if_eq:nnT {####3} {??} { ~(default) }
909     }
910   }
911 }
912 \exp_args:No \__hook_tmp:w { \use:nn { ~ } { ~ } }
913 }

```

(End definition for __hook_debug_print_rules:n.)

3.8 Specifying code for next invocation

`\hook_gput_next_code:nn`

```

914 \cs_new_protected:Npn \hook_gput_next_code:nn #1
915 { \__hook_normalize_hook_args:Nn \__hook_gput_next_code:nn {#1} }

```

(End definition for \hook_gput_next_code:nn. This function is documented on page 176.)

```

\__hook_gput_next_code:nn
\__hook_gput_next_do:nn
\__hook_gput_next_do:Nnn
\__hook_clear_next:n
916 \cs_new_protected:Npn \__hook_gput_next_code:nn #1 #2
917 {
918   \__hook_if_disabled:nTF {#1}
919   { \msg_error:nnn { hooks } { hook-disabled } {#1} }
920   {
921     \__hook_init_structure:n {#1}
922     \__hook_if_usable:nTF {#1}
923     { \__hook_gput_next_do:nn {#1} {#2} }
924     { \__hook_try_declaring_generic_next_hook:nn {#1} {#2} }
925   }
926 }
927 \cs_new_protected:Npn \__hook_gput_next_do:nn #1
928 {
929   \exp_args:Nc \__hook_gput_next_do:Nnn
930   { __hook_next~#1 } {#1}
931 }

```

First check if the “next code” token list is empty: if so we need to add a `\tl_gclear:c` to clear it, so the code lasts for one usage only. The token list is cleared early so that nested usages don’t get lost. `\tl_gclear:c` is used instead of `\tl_gclear:N` in case the hook is used in an expansion-only context, so the token list doesn’t expand before `\tl_gclear:N`: that would make an infinite loop. Also in case the main code token list is empty, the hook code has to be updated to add the next execution token list.

```

932 \cs_new_protected:Npn \__hook_gput_next_do:Nnn #1 #2
933 {
934   \tl_if_empty:cT { __hook~#2 }
935   { \__hook_update_hook_code:n {#2} }

```

```

936 \tl_if_empty:NT #1
937 { \__hook_tl_gset:Nn #1 { \__hook_clear_next:n {#2} } }
938 \__hook_tl_gput_right:Nn #1
939 }
940 \cs_new_protected:Npn \__hook_clear_next:n #1
941 { \cs_gset_eq:cN { \__hook_next~#1 } \c_empty_tl }

```

(End definition for __hook_gput_next_code:nn and others.)

3.9 Using the hook

`\hook_use:n` `\hook_use:n` as defined here is used in the preamble, where hooks aren't initialized by default. `__hook_use_initialized:n` is also defined, which is the non-`\protected` version for use within the document. Their definition is identical, except for the `__hook_preamble_hook:n` (which wouldn't hurt in the expandable version, but it would be an unnecessary extra expansion).

`__hook_use_initialized:n` holds the expandable definition while in the preamble. `__hook_preamble_hook:n` initializes the hook in the preamble, and is redefined to `\use_none:n` at `\begin{document}`.

Both versions do the same internally: check if the hook exist as given, and if so use it as quickly as possible. If it doesn't exist, the a call to `__hook_use:wn` checks for file hooks.

At `\begin{document}`, all hooks are initialized, and any change in them causes an update, so `\hook_use:n` can be made expandable. This one is better not protected so that it can expand into nothing if containing no code. Also important in case of generic hooks that we do not generate a `\relax` as a side effect of checking for a csname. In contrast to the \TeX low-level `\csname ... \endcsname` construct `\tl_if_exist:c` is careful to avoid this.

```

942 \cs_new_protected:Npn \hook_use:n #1
943 {
944   \tl_if_exist:cTF { \__hook~#1 }
945   {
946     \__hook_preamble_hook:n {#1}
947     \cs:w \__hook~#1 \cs_end:
948   }
949   { \__hook_use:wn #1 / \s__hook_mark {#1} }
950 }
951 \cs_new:Npn \__hook_use_initialized:n #1
952 {
953   \if_cs_exist:w \__hook~#1 \cs_end:
954   \else:
955     \__hook_use_undefined:w
956   \fi:
957   \cs:w \__hook~#1 \__hook_use_end:
958 }
959 \cs_new:Npn \__hook_use_undefined:w #1 #2 \__hook~#3 \__hook_use_end:
960 {
961   #1 % fi
962   \__hook_use:wn #3 / \s__hook_mark {#3}
963 }
964 \cs_new_protected:Npn \__hook_preamble_hook:n #1
965 { \__hook_initialize_hook_code:n {#1} }
966 \cs_new_eq:NN \__hook_use_end: \cs_end:

```

(End definition for `\hook_use:n` and others. This function is documented on page 175.)

`__hook_use:wn` `__hook_use:wn` does a quick check to test if the current hook is a file hook: those need a special treatment. If it is not, the hook does not exist. If it is, then `__hook_try_file_hook:n` is called, and checks that the current hook is a file-specific hook using `__hook_if_file_hook:wTF`. If it's not, then it's a generic file/ hook and is used if it exist.

If it is a file-specific hook, it passes through the same normalization as during declaration, and then it is used if defined. `__hook_if_usable_use:n` checks if the hook exist, and calls `__hook_preamble_hook:n` if so, then uses the hook.

```

967 \cs_new:Npn \__hook_use:wn #1 / #2 \s__hook_mark #3
968 {
969   \str_if_eq:nnTF {#1} { file }
970     { \__hook_try_file_hook:n {#3} }
971     { } % Hook doesn't exist
972 }
973 \cs_new_protected:Npn \__hook_try_file_hook:n #1
974 {
975   \__hook_if_file_hook:wTF #1 / / \s__hook_mark
976   {
977     \exp_args:Ne \__hook_if_usable_use:n
978       { \exp_args:Ne \__hook_file_hook_normalize:n {#1} }
979   }
980   { \__hook_if_usable_use:n {#1} } % file/ generic hook (e.g. file/before)
981 }
982 \cs_new_protected:Npn \__hook_if_usable_use:n #1
983 {
984   \tl_if_exist:cT { __hook~#1 }
985   {
986     \__hook_preamble_hook:n {#1}
987     \cs:w __hook~#1 \cs_end:
988   }
989 }

```

(End definition for `__hook_use:wn`, `__hook_try_file_hook:n`, and `__hook_if_usable_use:n`.)

`\hook_use_once:n` For hooks that can and should be used only once we have a special use command that remembers the hook name in `\g__hook_execute_immediately_prop`. This has the effect that any further code added to the hook is executed immediately rather than stored in the hook.

The code needs some gymnastics to prevent space trimming from the hook name, since `\hook_use:n` and `\hook_use_once:n` are documented to not trim spaces.

```

990 \cs_new_protected:Npn \hook_use_once:n #1
991 {
992   \tl_if_exist:cT { __hook~#1 }
993   {
994     \tl_set:Nn \l__hook_return_tl {#1}
995     \__hook_normalize_hook_args:Nn \__hook_use_once_store:n
996       { \l__hook_return_tl }
997     \hook_use:n {#1}
998   }
999 }

```

```

1000 \cs_new_protected:Npn \__hook_use_once_store:n #1
1001   { \prop_gput:Nnn \g__hook_execute_immediately_prop {#1} { } }

```

(End definition for `\hook_use_once:n`. This function is documented on page 175.)

3.10 Querying a hook

Simpler data types, like token lists, have three possible states; they can exist and be empty, exist and be non-empty, and they may not exist, in which case emptiness doesn't apply (though `\tl_if_empty:N` returns false in this case).

Hooks are a bit more complicated: they have several other states as discussed in 3.4.2. A hook may exist or not, and either way it may or may not be empty (even a hook that doesn't exist may be non-empty) or may be disabled.

A hook is said to be empty when no code was added to it, either to its permanent code pool, or to its “next” token list. The hook doesn't need to be declared to have code added to its code pool (it may happen that a package *A* defines a hook `foo`, but it's loaded after package *B*, which adds some code to that hook. In this case it is important that the code added by package *B* is remembered until package *A* is loaded).

All other states can only be queried with internal tests as the different states are irrelevant for package code.

`\hook_if_empty_p:n` Test if a hook is empty (that is, no code was added to that hook). A $\langle hook \rangle$ being empty means that all three of its `\g__hook_⟨hook⟩_code_prop`, its `__hook_toplevel_⟨hook⟩` and its `__hook_next_⟨hook⟩` are empty.

```

1002 \prg_new_conditional:Npnn \hook_if_empty:n #1 { p , T , F , TF }
1003   {
1004     \__hook_if_structure_exist:nTF {#1}
1005     {
1006       \bool_lazy_and:nnTF
1007         { \prop_if_empty_p:c { g__hook_#1_code_prop } }
1008         {
1009           \bool_lazy_and_p:nn
1010             { \tl_if_empty_p:c { __hook_toplevel~#1 } }
1011             { \tl_if_empty_p:c { __hook_next~#1 } }
1012         }
1013       { \prg_return_true: }
1014       { \prg_return_false: }
1015     }
1016     { \prg_return_true: }
1017   }

```

(End definition for `\hook_if_empty:nTF`. This function is documented on page 176.)

`__hook_if_usable_p:n` A hook is usable if the token list that stores the sorted code for that hook, `__hook_⟨hook⟩`, exists. The property list `\g__hook_⟨hook⟩_code_prop` cannot be used here because often it is necessary to add code to a hook without knowing if such hook was already declared, or even if it will ever be (for example, in case the package that defines it isn't loaded).

```

1018 \prg_new_conditional:Npnn \__hook_if_usable:n #1 { p , T , F , TF }
1019   {
1020     \tl_if_exist:cTF { __hook~#1 }
1021     { \prg_return_true: }
1022     { \prg_return_false: }

```

```

1023 }

(End definition for \__hook_if_usable:nTF.)

\__hook_if_structure_exist_p:n An internal check if the hook has already its basic internal structure set up with
\__hook_if_structure_exist:nTF \__hook_init_structure:n. This means that the hook was already used somehow (a
code chunk or rule was added to it), but it still wasn't declared with \hook_new:n.
1024 \prg_new_conditional:Npnn \__hook_if_structure_exist:n #1 { p , T , F , TF }
1025 {
1026   \prop_if_exist:cTF { g__hook_#1_code_prop }
1027   { \prg_return_true: }
1028   { \prg_return_false: }
1029 }

(End definition for \__hook_if_structure_exist:nTF.)

\__hook_if_declared_p:n Internal test to check if the hook was officially declared with \hook_new:n or a variant.
\__hook_if_declared:nTF 1030 \prg_new_conditional:Npnn \__hook_if_declared:n #1 { p , T , F , TF }
1031 {
1032   \tl_if_exist:cTF { g__hook_#1_declared_tl }
1033   { \prg_return_true: }
1034   { \prg_return_false: }
1035 }

(End definition for \__hook_if_declared:nTF.)

\__hook_if_reversed_p:n An internal conditional that checks if a hook is reversed.
\__hook_if_reversed:nTF 1036 \prg_new_conditional:Npnn \__hook_if_reversed:n #1 { p , T , F , TF }
1037 {
1038   \if_int_compare:w \cs:w g__hook_#1_reversed_tl \cs_end: 1 < 0 \exp_stop_f:
1039   \prg_return_true:
1040   \else:
1041   \prg_return_false:
1042   \fi:
1043 }

(End definition for \__hook_if_reversed:nTF.)

```

3.11 Messages

Hook errors are LaTeX kernel errors:

```

1044 \prop_gput:Nnn \g_msg_module_type_prop { hooks } { LaTeX }
And so are kernel errors (this should move elsewhere eventually).
1045 \prop_gput:Nnn \g_msg_module_type_prop { kernel } { LaTeX }
1046 %\prop_gput:Nnn \g_msg_module_name_prop { kernel } { } % <-- currently not working
1047 \msg_new:nnnn { hooks } { labels-incompatible }
1048 {
1049   Labels~'#1'~and~'#2'~are~incompatible
1050   \str_if_eq:nnF {#3} {??} { ~in-hook~'#3' } .~
1051   \int_compare:nNnTF {#4} = { 1 }
1052   { The~ code~ for~ both~ labels~ will~ be~ dropped. }
1053   { You~ may~ see~ errors~ later. }
1054 }
1055 { LaTeX~found~two~incompatible~labels~in~the~same~hook.~
1056   This~indicates~an~incompatibility~between~packages. }

```

```

1057 \msg_new:nnnn { hooks } { exists }
1058   { Hook~'#1'~ has~ already~ been~ declared. }
1059   { There~ already~ exists~ a~ hook~ declaration~ with~ this~
1060     name.\\
1061     Please~ use~ a~ different~ name~ for~ your~ hook.}
1062 \msg_new:nnnn { hooks } { hook-disabled }
1063   { Cannot~add~code~to~disabled~hook~'#1'. }
1064   {
1065     The~hook~'#1'~you~tried~to~add~code~to~was~previously~disabled~
1066     with~\iow_char:N\\hook_disable:n-or~\iow_char:N\\DisableHook,~so~
1067     it~cannot~have~code~added~to~it.
1068   }
1069 \msg_new:nnn { hooks } { empty-label }
1070   {
1071     Empty~code~label~\msg_line_context:~
1072     Using~'\_hook_currname_or_default:'~instead.
1073   }
1074 \msg_new:nnn { hooks } { no-default-label }
1075   {
1076     Missing~(empty)~default~label~\msg_line_context:. \\
1077     This~command~was~ignored.
1078   }
1079 \msg_new:nnnn { hooks } { unknown-rule }
1080   { Unknown~ relationship~ '#3'~
1081     between~ labels~ '#2'~ and~ '#4'~
1082     \str_if_eq:nnF {#1} {??} { ~in-hook~'#1' }. ~
1083     Perhaps~ a~ misspelling?
1084   }
1085   {
1086     The~ relation~ used~ not~ known~ to~ the~ system.~ Allowed~ values~ are~
1087     'before'~ or~ '<',~
1088     'after'~ or~ '>',~
1089     'incompatible-warning',~
1090     'incompatible-error',~
1091     'voids'~ or~
1092     'unrelated'.
1093   }
1094 \msg_new:nnnn { hooks } { misused-top-level }
1095   {
1096     Illegal~use~of~\iow_char:N \\AddToHook{#1}[top-level]{...}.\\
1097     'top-level'~is~reserved~for~the~user's~document.
1098   }
1099   {
1100     The~'top-level'~label~is~meant~for~user~code~only,~and~should~only~
1101     be~used~(sparingly)~in~the~main~document.~Use~the~default~label~
1102     '\_hook_currname_or_default:'~for~this~\@cls@pkg,~or~another~
1103     suitable~label.
1104   }
1105 \msg_new:nnn { hooks } { set-top-level }
1106   {
1107     You~cannot~change~the~default~label~#1~'top-level'.~Illegal \\
1108     \use:nn { ~ } { ~ } \iow_char:N \\#2{#3} \\

```

```

1109   \msg_line_context:.
1110 }
1111 \msg_new:nnn { hooks } { extra-pop-label }
1112 {
1113   Extra~\iow_char:N \PopDefaultHookLabel. \
1114   This~command~will~be~ignored.
1115 }
1116 \msg_new:nnn { hooks } { missing-pop-label }
1117 {
1118   Missing~\iow_char:N \PopDefaultHookLabel. \
1119   The~label~'#1'~was~pushed~but~never~popped.~Something~is~wrong.
1120 }
1121 \msg_new:nnn { kernel } { should-not-happen }
1122 {
1123   This~should~not~happen.~#1 \
1124   Please~report~at~https://github.com/latex3/latex2e.
1125 }
1126 \msg_new:nnn { hooks } { provide-disabled }
1127 {
1128   Cannot~ provide~ hook~ '#1'~ because~ it~ is~ disabled!
1129 }
1130 \msg_new:nnnn { hooks } { provide-error }
1131 {
1132   Hook~'#1'~ already~ declared~ as~ a~
1133   \__hook_if_reversed:nTF {#1} { reversed } { normal }~ hook!
1134 }
1135 {
1136   You~ attempted~ to~ provide~ the~ hook~'#1'~ as~ a~
1137   \__hook_if_reversed:nTF {#1} { normal } { reversed }~ hook,~ but~ it~
1138   was~ already~ previously~ declared~ as~ a~
1139   \__hook_if_reversed:nTF {#1} { reversed } { normal }~ hook.~
1140   A~ redeclaration~ is~ not~ possible.
1141 }

```

3.12 L^AT_EX 2_ε package interface commands

\NewHook Declaring new hooks ...

```

1142 \NewDocumentCommand \NewHook          { m }{ \hook_new:n {#1} }
1143 \NewDocumentCommand \NewReversedHook   { m }{ \hook_new_reversed:n {#1} }
1144 \NewDocumentCommand \NewMirroredHookPair { mm }{ \hook_new_pair:nn {#1}{#2} }

```

(End definition for \NewHook, \NewReversedHook, and \NewMirroredHookPair. These functions are documented on page 165.)

```

1145 <latexrelease>\IncludeInRelease{2021/06/01}%
1146 <latexrelease>      {\hook_provide:n}{Providing~hooks}

```

\ProvideHook Providing new hooks ...

```

1147 \NewDocumentCommand \ProvideHook          { m }{ \hook_provide:n {#1} }
1148 \NewDocumentCommand \ProvideReversedHook { m }{ \hook_provide_reversed:n {#1} }
1149 \NewDocumentCommand \ProvideMirroredHookPair { mm }{ \hook_provide_pair:nn {#1}{#2} }

```

(End definition for \ProvideHook, \ProvideReversedHook, and \ProvideMirroredHookPair. These functions are documented on page 166.)

\DisableHook Disabling a (generic) hook.

```
1150 \NewDocumentCommand \DisableHook { m }{ \hook_disable:n {#1} }

(End definition for \DisableHook. This function is documented on page 166.)

1151 \<latexrelease>\EndIncludeInRelease
1152 \<latexrelease>\IncludeInRelease{2020/10/01}
1153 \<latexrelease> \hook_provide:n}{Providing~hooks}
1154 \<latexrelease>
1155 \<latexrelease>\def \ProvideHook#1{}
1156 \<latexrelease>\def \ProvideReversedHook#1{}
1157 \<latexrelease>\def \ProvideMirroredHookPair#1#2{}
1158 \<latexrelease>
1159 \<latexrelease>\EndIncludeInRelease
```

\AddToHook

```
1160 \NewDocumentCommand \AddToHook { m o +m }
1161 { \hook_gput_code:nnn {#1} {#2} {#3} }

(End definition for \AddToHook. This function is documented on page 167.)
```

\AddToHookNext

```
1162 \NewDocumentCommand \AddToHookNext { m +m }
1163 { \hook_gput_next_code:nn {#1} {#2} }

(End definition for \AddToHookNext. This function is documented on page 168.)
```

\RemoveFromHook

```
1164 \NewDocumentCommand \RemoveFromHook { m o }
1165 { \hook_gremove_code:nn {#1} {#2} }

(End definition for \RemoveFromHook. This function is documented on page 167.)
```

\SetDefaultHookLabel
\PushDefaultHookLabel
\PopDefaultHookLabel

FMi: Docu task: At some point this code for this should be moved to the label section earlier and here we should keep only the interface commands.

The token list `\g__hook_hook_curr_name_tl` stores the name of the current package/file to be used as label for hooks. Providing a consistent interface is tricky, because packages can be loaded within packages, and some packages may not use `\SetDefaultHookLabel` to change the default label (in which case `\@currname` is used).

To pull that one off, we keep a stack that contains the default label for each level of input. The bottom of the stack contains the default label for the `top-level` (this stack should never go empty). If we're building the format, set the default label to be `top-level`:

```
1166 \tl_gset:Nn \g__hook_hook_curr_name_tl { top-level }
```

Then, in case we're in `latexrelease` we push something on the stack to support roll forward. But in some rare cases, `latexrelease` may be loaded inside another package (notably `platexrelease`), so we'll first push the `top-level` entry:

```
1167 \<latexrelease>\seq_if_empty:NT \g__hook_name_stack_seq
1168 \<latexrelease> { \seq_gput_right:Nn \g__hook_name_stack_seq { top-level } }
```

then we dissect the `\@currnamestack`, adding `\@currname` to the stack:

```

1169 <latexrelease>\cs_set_protected:Npn \__hook_tmp:w #1 #2 #3
1170 <latexrelease> {
1171 <latexrelease>   \quark_if_recursion_tail_stop:n {#1}
1172 <latexrelease>   \seq_gput_right:Nn \g__hook_name_stack_seq {#1}
1173 <latexrelease>   \__hook_tmp:w
1174 <latexrelease> }
1175 <latexrelease>\exp_after:wN \__hook_tmp:w \@currnamestack
1176 <latexrelease> \q_recursion_tail \q_recursion_tail
1177 <latexrelease> \q_recursion_tail \q_recursion_stop

```

and finally set the default label to be the `\@currname`:

```

1178 <latexrelease>\tl_gset:Nx \g__hook_hook_curr_name_tl { \@currname }
1179 <latexrelease>\seq_gpop_right:NN \g__hook_name_stack_seq \l__hook_tmpa_tl

```

Two commands keep track of the stack: when a file is input, `__hook_curr_name_push:n` pushes the current default label to the stack, and sets the new default label in one go:

```

1180 \cs_new_protected:Npn \__hook_curr_name_push:n #1
1181 { \exp_args:Nx \__hook_curr_name_push_aux:n { \__hook_make_name:n {#1} } }
1182 \cs_new_protected:Npn \__hook_curr_name_push_aux:n #1
1183 {
1184   \tl_if_blank:nTF {#1}
1185   { \msg_error:nn { hooks } { no-default-label } }
1186   {
1187     \str_if_eq:nnTF {#1} { top-level }
1188     {
1189       \msg_error:nnnnn { hooks } { set-top-level }
1190       { to } { PushDefaultHookLabel } {#1}
1191     }
1192     {
1193       \seq_gpush:NV \g__hook_name_stack_seq \g__hook_hook_curr_name_tl
1194       \tl_gset:Nn \g__hook_hook_curr_name_tl {#1}
1195     }
1196   }
1197 }

```

and when an input is over, the topmost item of the stack is popped, since the label will not be used again, and `\g__hook_hook_curr_name_tl` is updated to the now topmost item of the stack:

```

1198 \cs_new_protected:Npn \__hook_curr_name_pop:
1199 {
1200   \seq_gpop:NNTF \g__hook_name_stack_seq \l__hook_return_tl
1201   { \tl_gset_eq:NN \g__hook_hook_curr_name_tl \l__hook_return_tl }
1202   { \msg_error:nn { hooks } { extra-pop-label } }
1203 }

```

At the end of the document we want to check if there was no `__hook_curr_name_push:n` without a matching `__hook_curr_name_pop:` (not a critical error, but it might indicate that something else is not quite right):

```

1204 \tl_gput_right:Nn \@kernel@after@enddocument@afterlastpage
1205 { \__hook_end_document_label_check: }
1206 \cs_new_protected:Npn \__hook_end_document_label_check:
1207 {
1208   \seq_gpop:NNT \g__hook_name_stack_seq \l__hook_return_tl

```

```

1209     {
1210       \msg_error:nxx { hooks } { missing-pop-label }
1211       { \g__hook_hook_curr_name_tl }
1212       \tl_gset_eq:NN \g__hook_hook_curr_name_tl \l__hook_return_tl
1213       \__hook_end_document_label_check:
1214     }
1215   }

```

The token list `\g__hook_hook_curr_name_tl` is but a mirror of the top of the stack.

Now define a wrapper that replaces the top of the stack with the argument, and updates `\g__hook_hook_curr_name_tl` accordingly.

```

1216 \NewDocumentCommand \SetDefaultHookLabel { m }
1217 {
1218   \seq_if_empty:NTF \g__hook_name_stack_seq
1219   {
1220     \msg_error:nnnnn { hooks } { set-top-level }
1221     { for } { SetDefaultHookLabel } {#1}
1222   }
1223   { \exp_args:Nx \__hook_set_default_label:n { \__hook_make_name:n {#1} } }
1224 }
1225 \cs_new_protected:Npn \__hook_set_default_label:n #1
1226 {
1227   \str_if_eq:nnTF {#1} { top-level }
1228   {
1229     \msg_error:nnnnn { hooks } { set-top-level }
1230     { to } { SetDefaultHookLabel } {#1}
1231   }
1232   { \tl_gset:Nn \g__hook_hook_curr_name_tl {#1} }
1233 }

```

The label is only automatically updated with `\@onefilewithoptions` (`\usepackage` and `\documentclass`), but some packages, like `TikZ`, define package-like interfaces, like `\usetikzlibrary` that are wrappers around `\input`, so they inherit the default label currently in force (usually `top-level`, but it may change if loaded in another package). To provide a package-like behaviour also for hooks in these files, we provide high-level access to the default label stack.

```

1234 \NewDocumentCommand \PushDefaultHookLabel { m }
1235 { \__hook_curr_name_push:n {#1} }
1236 \NewDocumentCommand \PopDefaultHookLabel { }
1237 { \__hook_curr_name_pop: }

```

The current label stack holds the labels for all files but the current one (more or less like `\@currnamestack`), and the current label token list, `\g__hook_hook_curr_name_tl`, holds the label for the current file. However `\@pushfilename` happens before `\@currname` is set, so we need to look ahead to get the `\@currname` for the label. `expl3` also requires the current file in `\@pushfilename`, so here we abuse `\@expl@push@filename@aux@@` to do `__hook_curr_name_push:n`.

```

1238 \cs_gset_protected:Npn \@expl@push@filename@aux@@ #1#2#3
1239 {
1240   \__hook_curr_name_push:n {#3}
1241   \str_gset:Nx \g_file_curr_name_str {#3}
1242   #1 #2 {#3}
1243 }

```

(End definition for `\SetDefaultHookLabel` and others. These functions are documented on page 170.)

\UseHook Avoid the overhead of xparse and its protection that we don't want here (since the hook should vanish without trace if empty)!

\UseOneTimeHook

```
1244 \cs_new:Npn \UseHook      { \hook_use:n }
1245 \cs_new:Npn \UseOneTimeHook { \hook_use_once:n }
```

(End definition for \UseHook and \UseOneTimeHook. These functions are documented on page 166.)

\ShowHook

\LogHook

```
1246 \cs_new_protected:Npn \ShowHook { \hook_show:n }
1247 \cs_new_protected:Npn \LogHook { \hook_log:n }
```

(End definition for \ShowHook and \LogHook. These functions are documented on page 173.)

\DebugHooksOn

\DebugHooksOff

```
1248 \cs_new_protected:Npn \DebugHooksOn { \hook_debug_on: }
1249 \cs_new_protected:Npn \DebugHooksOff { \hook_debug_off: }
```

(End definition for \DebugHooksOn and \DebugHooksOff. These functions are documented on page 174.)

\DeclareHookRule

```
1250 \NewDocumentCommand \DeclareHookRule { m m m m }
1251 { \hook_gset_rule:nnnn {#1}{#2}{#3}{#4} }
```

(End definition for \DeclareHookRule. This function is documented on page 172.)

\DeclareDefaultHookRule

This declaration is only supported before \begin{document}.

```
1252 \NewDocumentCommand \DeclareDefaultHookRule { m m m }
1253 { \hook_gset_rule:nnnn {??}{#1}{#2}{#3} }
1254 \@onlypreamble\DeclareDefaultHookRule
```

(End definition for \DeclareDefaultHookRule. This function is documented on page 172.)

\ClearHookRule

A special setup rule that removes an existing relation. Basically @@_rule_gclear:nnm plus fixing the property list for debugging.

FMi: Needs perhaps an L3 interface, or maybe it should get dropped?

```
1255 \NewDocumentCommand \ClearHookRule { m m m }
1256 { \hook_gset_rule:nnnn {#1}{#2}{unrelated}{#3} }
```

(End definition for \ClearHookRule. This function is documented on page 172.)

\IfHookEmptyTF

Here we avoid the overhead of xparse, since \IfHookEmptyTF is used in \end (that is, every L^AT_EX environment). As a further optimisation, use \let rather than \def to avoid one expansion step.

```
1257 \cs_new_eq:NN \IfHookEmptyTF \hook_if_empty:nTF
```

(End definition for \IfHookEmptyTF. This function is documented on page 173.)

\IfHookExistsTF

Marked for removal and no longer documented in the doc section!

PhO: \IfHookExistsTF is used in jlreq.cls, pxatbegshi.sty, pxeveryrel.sty, pxeveryshi.sty, so the public name may be an alias of the internal conditional for a while. Regardless, those packages' use for \IfHookExistsTF is not really correct and can be changed.

```
1258 \cs_new_eq:NN \IfHookExistsTF \__hook_if_usable:nTF
```

(End definition for \IfHookExistsTF.)

3.13 Internal commands needed elsewhere

Here we set up a few horrible (but consistent) L^AT_EX_{2 ϵ} names to allow for internal commands to be used outside this module. We have to unset the @@ since we want double “at” sign in place of double underscores.

1259 <@@=>

\@expl@@@initialize@all@@

\@expl@@@hook@curr@name@pop@@

1260 \cs_new_eq:NN \@expl@@@initialize@all@@

1261 __hook_initialize_all:

1262 \cs_new_eq:NN \@expl@@@hook@curr@name@pop@@

1263 __hook_curr_name_pop:

(End definition for \@expl@@@initialize@all@@ and \@expl@@@hook@curr@name@pop@@.)

Rolling back here doesn’t undefine the interface commands as they may be used in packages without rollback functionality. So we just make them do nothing which may or may not work depending on the code usage.

1264 %

1265 <latexrelease>\IncludeInRelease{0000/00/00}%

1266 <latexrelease>\lthooks{The~hook~management}%

1267 <latexrelease>

1268 <latexrelease>\def \NewHook#1{}

1269 <latexrelease>\def \NewReversedHook#1{}

1270 <latexrelease>\def \NewMirroredHookPair#1#2{}

1271 <latexrelease>

1272 <latexrelease>\def \DisableHook #1{}

1273 <latexrelease>

1274 <latexrelease>\long\def \AddToHookNext#1#2{}

1275 <latexrelease>

1276 <latexrelease>\def \AddToHook#1{\@gobble\AddToHook@args}

1277 <latexrelease>\providecommand\@gobble\AddToHook@args[2] [] {}

1278 <latexrelease>

1279 <latexrelease>\def \RemoveFromHook#1{\@gobble\RemoveFromHook@arg}

1280 <latexrelease>\providecommand\@gobble\RemoveFromHook@arg[1] [] {}

1281 <latexrelease>

1282 <latexrelease>\def \UseHook #1{}

1283 <latexrelease>\def \UseOneTimeHook #1{}

1284 <latexrelease>\def \ShowHook #1{}

1285 <latexrelease>\let \DebugHooksOn \@empty

1286 <latexrelease>\let \DebugHooksOff \@empty

1287 <latexrelease>

1288 <latexrelease>\def \DeclareHookRule #1#2#3#4{}

1289 <latexrelease>\def \DeclareDefaultHookRule #1#2#3{}

1290 <latexrelease>\def \ClearHookRule #1#2#3{}

If the hook management is not provided we make the test for existence false and the test for empty true in the hope that this is most of the time reasonable. If not a package would need to guard against running in an old kernel.

1291 <latexrelease>\long\def \IfHookExistsTF #1#2#3{#3}

1292 <latexrelease>\long\def \IfHookEmptyTF #1#2#3{#2}

1293 <latexrelease>

1294 <latexrelease>\EndModuleRelease

1295 \ExplSyntaxOff

1296 </2ekernel | latexrelease>

File i

ltxcmdhooks.dtx

Contents

1 Introduction

This file implements generic hooks for (arbitrary) commands. In theory every command `\<name>` offers now two associated hooks to which code can be added using `\AddToHook` or `\AddToHookNext`.¹⁰ These are

cmd/<name>/before This hook is executed at the very start of the command execution after its arguments (if any) are parsed. The hook `<code>` is wrapped in the command inside a call to `\UseHook{cmd/<name>/before}`, so the arguments passed to the command are *not* available in the hook `<code>`.

cmd/<name>/after This hook is similar to `cmd/<name>/before`, but it is executed at the very end of the command body. This hook is implemented as a reversed hook.

The hooks are not physically present before `\begin{document}` (i.e., using a command in the preamble will never execute them) and if nobody has declared any code for them, then they are not added to the command code ever. For example, if we have the following definition

```
\newcommand\foo[2]{Code #1 for #2!}
```

then executing `\foo{A}{B}` will simply run `Code_A_for_B!` as it was always the case. However, if somebody, somewhere (e.g., in a package) adds

```
\AddToHook{cmd/foo/before}{<before code>}
```

then, after `\begin{document}` the definition of `\foo` will be:

```
\renewcommand\foo[2]{\UseHook{cmd/foo/before}Code #1 for #2!}
```

and similarly `\AddToHook{cmd/foo/after}{<after code>}` alters the definition to

```
\renewcommand\foo[2]{Code #1 for #2!\UseHook{cmd/foo/after}}
```

In other words, the mechanism is similar to what `etoolbox` offers with `\pretocmd` and `\apptocmd` with the important differences

- that code can be prepended or appended (i.e., added to the hooks) even if the command itself is not defined, because the defining package has not yet been loaded
- and that by using the hook management interface it is now possible to define how the code chunks added in these places are ordered, if different packages want to add code at these points.

¹⁰In practice this is not supported for all types of commands, see section 2.2 for the restrictions that apply and what happens if one tries to use this with commands for which this is not supported.

2 Restrictions and Operational details

Adding arbitrary material to commands is tricky because most of the time we do not know what the macro expects as arguments when expanding and \TeX doesn't have a reliable way to see that, so some guesswork has to be employed.

2.1 Patching

The code here tries to find out if a command was defined with `\newcommand` or `\DeclareRobustCommand` or `\NewDocumentCommand`, and if so it *assumes* that the argument specification of the command is as expected (which is not fail-proof, if someone redefines the internals of these commands in devious ways, but is a reasonable assumption).

If the command is one of the defined types, the code here does a sandboxed expansion of the command such that it can be redefined again exactly as before, but with the hook code added.

If however the command is not a known type (it was defined with `\def`, for example), then the code uses an approach similar to `etoolbox`'s `\patchcmd` to retokenize the command with the hook code in place. This procedure, however, is more likely to fail if the catcode settings are not the same as the ones at the time of command's definition, so not always adding a hook to a command will work.

2.1.1 Timing

When `\AddToHook` (or its `expl3` equivalent) is called with a generic `cmd` hook, say, `cmd/foo/before`, for the first time (that is, no code was added to that same hook before), in the preamble of a document, it will store a patch instruction for that command until `\begin{document}`, and only then all the commands which had hooks added will be patched in one go. That means that no command in the preamble will have hooks patched into them.

At `\begin{document}` all the delayed patches will be executed, and if the command doesn't exist the code is still added to the hook, but it will not be executed. After `\begin{document}`, when `\AddToHook` is called with a generic `cmd` hook the first time, the command will be immediately patched to include the hook, and if it doesn't exist or if it can't be patched for any reason, an error is thrown; if `\AddToHook` was already used in the preamble no new patching is attempted.

This has the consequence that a command defined or redefined after `\begin{document}` only uses generic `cmd` hook code if `\AddToHook` is called for the first time after the definition is made, or if the command explicitly uses the generic hook in its definition by declaring it with `\NewHookPair` adding `\UseHook` as part of the code.¹¹

2.2 Commands that look ahead

Some commands are defined in different "steps" and they look ahead in the input stream to find more arguments. If you try to add some code to the `cmd/<name>/after` hook of such command, it will not work, and it is not possible to detect that programmatically, so the user has to know (or find out) which commands can or cannot have hooks attached to them.

¹¹We might change this behavior in the main document slightly after gaining some usage experience.

One good example is the `\section` command. You can add something to the `cmd/section/before` hook, but if you try to add something to the `cmd/section/after` hook, `\section` will no longer work. That happens because the `\section` macro takes no argument, but instead calls a few internal L^AT_EX macros to look for the optional and mandatory arguments. By adding code to the `cmd/section/after` hook, you get in the way of that scanning.

3 Package Author Interface

The `cmd` hooks are, by default, available for all commands that can be patched to add the hooks. For some commands, however, the very beginning or the very end of the code is not the best place to put the hooks, for example, if the command looks ahead for arguments (see section 2.2).

If you are a package author and you want to add the hooks to your own commands in the proper position you can define the command and manually add the `\UseHook` calls inside the command in the proper positions, and manually define the hooks with `\NewHook` or `\NewReversedHook`. When the hooks are explicitly defined, patching is not attempted so you can make sure your command works properly. For example, an (admittedly not really useful) command that typesets its contents in a framed box with width optionally given in parentheses:

```
\newcommand\fancybox{\@ifnextchar({\@fancybox}{\@fancybox(5cm)}}
\def\@fancybox(#1)#2{\fbox{\parbox{#1}{#2}}}
```

If you try that definition, then add some code after it with

```
\AddToHook{cmd/fancybox/after}{<code>}
```

and then use the `\fancybox` command you will see that it will be completely broken, because the hook will get executed in the middle of parsing for optional (...) argument.

If, on the other hand, you want to add hooks to your command you can do something like:

```
\newcommand\fancybox{\@ifnextchar({\@fancybox}{\@fancybox(5cm)}}
\def\@fancybox(#1)#2{\fbox{%
    \UseHook{cmd/fancybox/before}%
    \parbox{#1}{#2}%
    \UseHook{cmd/fancybox/after}}}
\NewHook{cmd/fancybox/before}
\NewReversedHook{cmd/fancybox/after}
```

then the hooks will be executed where they should and no patching will be attempted. It is important that the hooks are declared with `\NewHook` or `\NewReversedHook`, otherwise the command hook code will try to patch the command. Note also that the call to `\UseHook{cmd/fancybox/before}` does not need to be in the definition of `\fancybox`, but anywhere it makes sense to insert it (in this case in the internal `\@fancybox`).

Alternatively, if for whatever reason your command does not support the generic hooks provided here, you can disable a hook with `\DisableHook`¹², so that when someone

¹²Please use `\DisableHook` if at all, only on hooks that you “own”, i.e., for commands that your package or class defines and not second guess whether or not hooks of other packages should get disabled!

tries to add code to it they will get an error. Or if you don't want the error, you can simply declare the hook with `\NewHook` and never use it.

The above approach is useful for really complex commands where for one or the other reason the hooks can't be placed at the very beginning and end of the command body and some hand-crafting is needed. However, in the example above the real (and in fact only) issue is the cascading argument parsing in the style developed long ago in L^AT_EX 2.09. Thus, a much simpler solution for this case is to replace it with the modern `\NewDocumentCommand` syntax and define the command as follows:

```
\DeclareDocumentCommand\fancybox{D()}{5cm}m}{\fbox{\parbox{#1}{#2}}}
```

If you do that then both hooks automatically work and are patched into the right places.

4 The Implementation

4.1 Execution plan

To add **before** and **after** hooks to a command we will need to peek into the definition of a command, which is always a tricky thing to do. Some cases are easy because we know how the command was defined, so we can assume how its *⟨parameter text⟩* looks like (for example a command defined with `\newcommand` may have an optional argument followed by a run of mandatory arguments), so we can just expand that command and make it grab `#1`, `#2`, etc. as arguments and define it all back with the hooks added.

Life's usually not that easy, so with some commands we can't do that (a `#1` might as well be `#12112` instead of the expected `#6112`, for example) so we need to resort to "patching" the command: read its `\meaning`, and tokenize it again with `\scantokens` and hope for the best.

So the overall plan is:

1. Check if a command is of a known type (that is, defined with `\newcommand`¹³, `\DeclareRobustCommand`, or `\New(Expandable)DocumentCommand`), and if is, take appropriate action.
2. If the command is not a known type, we'll check if the command can be patched. Two things will prevent a command from being patched: if it was defined in a nonstandard catcode setting, or if it is an internal expl3 command with `__⟨module⟩` in its name, in which case we refuse to patch.
3. If the command was defined in nonstandard catcode settings, we will try a few standard ones to try our best to carry out the patching. If this doesn't help either, the code will give up and throw an error.

```
1 <@@=hook>
2 <*2kernel | latexrelease>
3 \ExplSyntaxOn
4 <latexrelease> \NewModuleRelease{2021/06/01}{ltxcmdhooks}
5 <latexrelease> {The-hook-management-system-for-commands}
```

¹³It's not always possible to reliably detect this case because a command defined with no optional argument is indistinguishable from a `\defed` command.

4.2 Variables

<code>\g_hook_patch_action_list_tl</code>	<p>Pairs of <code>\if<cmd>..\patch<cmd></code> to be used with <code>\robust@command@act</code> when looking for a known patching rule. This token list is exposed because we see some future applications (with very specialized packages, such as <code>etoolbox</code> that may want to extend the pairs processed. It is not meant for general use which is why it is not documented in the interface documentation above.</p> <pre> 6 \tl_new:N \g_hook_patch_action_list_tl </pre> <p><i>(End definition for <code>\g_hook_patch_action_list_tl</code>.)</i></p>
<code>\l__hook_patch_num_args_int</code>	<p>The number of arguments in a macro being patched.</p> <pre> 7 \int_new:N \l__hook_patch_num_args_int </pre> <p><i>(End definition for <code>\l__hook_patch_num_args_int</code>.)</i></p>
<code>\l__hook_patch_prefixes_tl</code> <code>\l__hook_patch_param_text_tl</code> <code>\l__hook_patch_replacement_tl</code>	<p>The prefixes and parameters of the definition for the macro being patched.</p> <pre> 8 \tl_new:N \l__hook_patch_prefixes_tl 9 \tl_new:N \l__hook_patch_param_text_tl 10 \tl_new:N \l__hook_patch_replacement_tl </pre> <p><i>(End definition for <code>\l__hook_patch_prefixes_tl</code>, <code>\l__hook_patch_param_text_tl</code>, and <code>\l__hook_patch_replacement_tl</code>.)</i></p>
<code>\g_hook_delayed_patches_prop</code>	<p>A list containing the patches delayed to <code>\begin{document}</code>, so that patching is not attempted twice.</p> <pre> 11 \prop_new:N \g_hook_delayed_patches_prop </pre> <p><i>(End definition for <code>\g_hook_delayed_patches_prop</code>.)</i></p>
<code>__hook_patch_debug:x</code>	<p>A helper for patching debug info.</p> <pre> 12 \cs_new_protected:Npn __hook_patch_debug:x #1 13 { __hook_debug:n { \iow_term:x { [lthooks]~#1 } } } </pre> <p><i>(End definition for <code>__hook_patch_debug:x</code>.)</i></p>

4.3 Variants

<code>\tl_rescan:nV</code>	<p><code>expl3</code> function variants used throughout the code.</p> <pre> 14 \cs_generate_variant:Nn \tl_rescan:nn { nV } </pre> <p><i>(End definition for <code>\tl_rescan:nV</code>.)</i></p>
----------------------------	---

4.4 Patching or delaying

`__hook_try_put_cmd_hook:n` Before `\begin{document}` all patching is delayed. This function is called from within `\AddToHook`, when code is added to a generic `cmd` hook is newly declared. It checks whether the patch position is valid, then proceeds to trying to patch or delaying to `\begin{document}` if in the preamble.

```

15 \cs_new_protected:Npn \__hook_try_put_cmd_hook:n #1
16   { \__hook_try_put_cmd_hook:w #1 / / / \s__hook_mark {#1} }
17 \cs_new_protected:Npn \__hook_try_put_cmd_hook:w
18   #1 / #2 / #3 / #4 \s__hook_mark #5
19   {
20     \__hook_debug:n { \iow_term:n { ->~Adding~cmd~hook~to~'~#2'~(~#3): } }
21     \str_case:nnTF {#3}
22       { { before } { } { after } { } }
23       { \exp_args:Nc \__hook_patch_cmd_or_delay:Nnn {#2} {#2} {#3} }
24       { \msg_error:nnnn { hooks } { wrong-cmd-hook } {#2} {#3} }
25   }

```

(End definition for `__hook_try_put_cmd_hook:n` and `__hook_try_put_cmd_hook:w`.)

`__hook_patch_cmd_or_delay:Nnn` In the preamble, `__hook_patch_cmd_or_delay:Nnn` just adds the patch instruction to a property list to be executed later.

```

26 \cs_new_protected:Npn \__hook_patch_cmd_or_delay:Nnn #1 #2 #3
27   {
28     \__hook_debug:n { \iow_term:n { ->~Add~generic~cmd~hook~for~#2~(~#3). } }
29     \__hook_debug:n
30     { \iow_term:n { !~In~the~preamble:~delaying. } }
31     \prop_gput:Nnn \g__hook_delayed_patches_prop { #2 / #3 }
32     { \__hook_cmd_try_patch:nn {#2} {#3} }
33   }

```

The delayed patches are added to a property list to prevent duplication, and the code stored in the property list for each key is executed. The function `__hook_patch_cmd_or_delay:Nnn` is also redefined to be `__hook_patch_command:Nnn` so that no further delaying is attempted.

```

34 \cs_new_protected:Npn \__hook_cmd_begindocument_code:
35   {
36     \cs_gset_eq:NN \__hook_patch_cmd_or_delay:Nnn \__hook_patch_command:Nnn
37     \prop_map_function:NN \g__hook_delayed_patches_prop { \use_i:nn }
38     \prop_gclear:N \g__hook_delayed_patches_prop
39     \cs_undefine:N \__hook_cmd_begindocument_code:
40   }
41 \g@addto@macro \@kernel@after@begindocument
42   { \__hook_cmd_begindocument_code: }

```

(End definition for `__hook_patch_cmd_or_delay:Nnn` and `__hook_cmd_begindocument_code:.`)

`__hook_cmd_try_patch:nn` At `\begin{document}` tries patching the command if the hook was not manually created in the meantime. If the document does not exist, no error is raised here as it may hook into a package that wasn't loaded. Hooks added to commands in the document body still raise an error if the command is not defined.

```

43 \cs_new_protected:Npn \__hook_cmd_try_patch:nn #1 #2
44   {
45     \__hook_debug:n

```

```

46     { \iow_term:x { ->~\string\begin{document}~try~cmd / #1 / #2. } }
47   \__hook_if_declared:nTF { cmd / #1 / #2 }
48   {
49     \__hook_debug:n
50     { \iow_term:n { .->~Giving~up:~hook~already~created. } }
51   }
52   {
53     \cs_if_exist:cT {#1}
54     { \exp_args:Nc \__hook_patch_command:Nnn {#1} {#1} {#2} }
55   }
56 }

```

(End definition for __hook_cmd_try_patch:nn.)

4.5 Patching commands

__hook_patch_command:Nnn __hook_patch_command:Nnn will do some sanity checks on the argument to detect if it is possible to add hooks to the command, and raises an error otherwise. If the command can contain hooks, then it uses \robust@command@act to find out what type is the command, and patch it accordingly.

```

\__hook_patch_command:Nnn \__hook_patch_check:NNnn
\__hook_if_public_command:NTF
\__hook_if_public_command:w
57 \cs_new_protected:Npn \__hook_patch_command:Nnn #1 #2 #3
58 {
59   \__hook_patch_debug:x { analyzing~'\token_to_str:N #1' }
60   \__hook_patch_debug:x { \token_to_str:N #1 = \token_to_meaning:N #1 }
61   \__hook_patch_check:NNnn \cs_if_exist:NTF #1 { undef }
62   {
63     \__hook_patch_debug:x { ++control~sequence~is~defined }
64     \__hook_patch_check:NNnn \token_if_macro:NTF #1 { macro }
65     {
66       \__hook_patch_debug:x { ++control~sequence~is~a~macro }
67       \__hook_patch_check:NNnn \__hook_if_public_command:NTF #1 { expl3 }
68       {
69         \__hook_patch_debug:x { ++macro~is~not~private }
70         \robust@command@act
71         \g_hook_patch_action_list_tl #1
72         \__hook_retokenize_patch:Nnn { #1 {#2} {#3} }
73       }
74     }
75   }
76 }

```

And here's the auxiliary used above:

```

77 \cs_new_protected:Npn \__hook_patch_check:NNnn #1 #2 #3 #4
78 {
79   #1 #2 {#4}
80   {
81     \msg_error:nnxx { hooks } { cant-patch }
82     { \token_to_str:N #2 } {#3}
83   }
84 }

```

and a conditional __hook_if_public_command:N to check if a command has __ in its name (no other checking is performed). Primitives with :D in their name could be included here, but they are already discarded in the \token_if_macro:NTF test above.

```

85 \use:x
86 {
87   \prg_new_protected_conditional:Npnn
88     \exp_not:N \__hook_if_public_command:N ##1 { TF }
89   {
90     \exp_not:N \exp_last_unbraced:Nf
91     \exp_not:N \__hook_if_public_command:w
92     { \exp_not:N \cs_to_str:N ##1 }
93     \tl_to_str:n { _ _ } \s__hook_mark
94   }
95 }
96 \exp_last_unbraced:NNNNo
97 \cs_new_protected:Npn \__hook_if_public_command:w
98   #1 \tl_to_str:n { _ _ } #2 \s__hook_mark
99   {
100     \tl_if_empty:nTF {#2}
101     { \prg_return_true: }
102     { \prg_return_false: }
103   }

```

(End definition for `__hook_patch_command:Nnn` and others.)

4.5.1 Patching by expansion and redefinition

This is the list of known command types and the function that patches the command hooks into them. The conditionals are taken from `\ShowCommand`, `\NewCommandCopy` and `__kernel_cmd_if_xparse:NTF` defined in `ltxcmd`.

```

104 \tl_gset:Nn \g_hook_patch_action_list_tl
105 {
106   { \@if@DeclareRobustCommand \__hook_patch_DeclareRobustCommand:Nnn }
107   { \@if@newcommand \__hook_patch_newcommand:Nnn }
108   { \__kernel_cmd_if_xparse:NTF \__hook_cmd_patch_xparse:Nnn }
109 }

```

(End definition for `\g_hook_patch_action_list_tl`.)

At this point we know that the commands can be patched by expanding then redefining. These are the cases of commands defined with `\newcommand` with an optional argument or with `\DeclareRobustCommand`.

With `__hook_patch_DeclareRobustCommand:Nnn` we check if the command has an optional argument (with a test counter-intuitively called `\@if@newcommand`). If so, we forward the action to `__hook_patch_newcommand:Nnn`, otherwise call the patching engine `__hook_patch_expand_redefine:NNnn` with a `\c_false_bool` to indicate that there is no optional argument.

```

110 \cs_new_protected:Npn \__hook_patch_DeclareRobustCommand:Nnn #1
111 {
112   \exp_args:Nc \@if@newcommand { \cs_to_str:N #1 ~ }
113   { \exp_args:Nc \__hook_patch_newcommand:Nnn }
114   { \exp_args:NNc \__hook_patch_expand_redefine:NNnn \c_false_bool }
115   { \cs_to_str:N #1 ~ }
116 }

```

(End definition for `__hook_patch_DeclareRobustCommand:Nnn`.)

`_hook_patch_newcommand:Nnn` If the command was defined with `\newcommand` and an optional argument, call the patching engine with a `\c_true_bool` to flag the presence of an optional argument, and with `\command` to patch the actual code for `\command`.

```

117 \cs_new_protected:Npn \_hook_patch_newcommand:Nnn #1
118 {
119     \exp_args:Nnc \_hook_patch_expand_redefine:NNnn \c_true_bool
120     { \c_backslash_str \cs_to_str:N #1 }
121 }

```

(End definition for `_hook_patch_newcommand:Nnn`.)

`_hook_cmd_patch_xparse:Nnn` And for commands defined by the `xparse` commands use this for patching:

```

122 \cs_new_protected:Npn \_hook_cmd_patch_xparse:Nnn #1
123 {
124     \exp_args:Nnc \_hook_patch_expand_redefine:NNnn \c_false_bool
125     { \cs_to_str:N #1 ~ code }
126 }

```

(End definition for `_hook_cmd_patch_xparse:Nnn`.)

`_hook_patch_expand_redefine:NNnn` Now the real action begins. Here we have in `#1` a boolean indicating if the command has a [...] delimited argument, in `#2` the command control sequence, in `#3` the name of the command (note that `#1 ≠ \csname#2\endcsname` at this point!), and in `#4` the hook position, either before or after.

`_hook_make_prefixes:w`

```

127 \cs_new_protected:Npn \_hook_patch_expand_redefine:NNnn #1 #2 #3 #4
128 {
129     \_hook_patch_debug:x { ++~command~can~be~patched~without~rescanning }

```

We'll start by counting the number of arguments in the command by counting the number of characters in the `\cs_argument_spec:N` of the macro, divided by two, and subtracting one if the command has an optional argument (that is, an extra `[]` in its *parameter text*).

```

130     \int_set:Nn \l__hook_patch_num_args_int
131     {
132         \exp_args:Nf \str_count:n { \cs_argument_spec:N #2 } / 2
133         \bool_if:NT #1 { -1 }
134     }

```

Now build two token lists:

`\l__hook_patch_param_text_tl` will contain the *parameter text* to be used when redefining the macro. It should be identical to the *parameter text* used when originally defining that macro.

`\l__hook_patch_replacement_tl` will contain braced pairs of `#12<num>` to feed to the macro when expanded. This token list as well as the previous will have the first item surrounded by [...] in the case of an optional argument.

```

135     \int_compare:nNnTF { \l__hook_patch_num_args_int } > { \c_zero_int }
136     {
137         \tl_set:Nx \l__hook_patch_param_text_tl
138         { \bool_if:NTF #1 { [####1] } { #####1 } }
139         \tl_set:Nx \l__hook_patch_replacement_tl
140         { \bool_if:NTF #1 { [####1] } { {#####1} } }
141         \int_step_inline:nnn { 2 } { \l__hook_patch_num_args_int }
142         {

```

```

143         \tl_put_right:Nn \l__hook_patch_param_text_tl { ## #####1 }
144         \tl_put_right:Nn \l__hook_patch_replacement_tl { { ## #####1 } }
145     }
146 }
147 {
148     \tl_clear:N \l__hook_patch_param_text_tl
149     \tl_clear:N \l__hook_patch_replacement_tl
150 }

```

Finally, before redefining, we need to also get the prefixes used when defining the command. Here we ensure that the `\escapechar` is printable, otherwise a macro defined with prefixes `\protected \long` will have it `\meaning` printed as `protectedlong`, making life unnecessarily complicated. Here the `\escapechar` is changed to `/`, then we loop between pairs of `/.../` extracting the prefixes.

```

151 \group_begin:
152 \int_set:Nn \tex_escapechar:D { '/' }
153 \use:x
154 {
155 \group_end:
156 \tl_set:Nx \exp_not:N \l__hook_patch_prefixes_tl
157 { \exp_not:N \__hook_make_prefixes:w \cs_prefix_spec:N #2 / / }
158 }

```

Now that all the needed tools are ready, without further ado we'll redefine the command `#2`. The definition uses the prefixes gathered in `\l__hook_patch_prefixes_tl`, a primitive `\tex_def:D` to avoid adding extra prefixes, and the *parameter text* from `\l__hook_patch_param_text_tl`.

Then finally, in the body of the definition, we insert `cmd/#3/before` if `#4` is `before`, the code of the command expanded once grabbing the parameters in `\l__hook_patch_replacement_tl`, and `cmd/#3/after` if `#4` is `after`.

```

159 \use:x
160 {
161     \l__hook_patch_prefixes_tl \tex_def:D
162     \exp_not:N #2 \exp_not:N \l__hook_patch_param_text_tl
163     {
164         \str_if_eq:nnT {#4} { before }
165         { \exp_not:N \UseHook { cmd / #3 / #4 } }
166         \exp_args:No \exp_not:o
167         { \exp_after:wN #2 \l__hook_patch_replacement_tl }
168         \str_if_eq:nnT {#4} { after }
169         { \exp_not:N \UseHook { cmd / #3 / #4 } }
170     }
171 }
172 }

```

Here's the auxiliary that makes the prefix control sequences for the redefinition. Each item has to be `\tl_trim_spaces:n'd` because the last item (and not any other) has a trailing space.

```

173 \cs_new:Npn \__hook_make_prefixes:w / #1 /
174 {
175     \tl_if_empty:nF {#1}
176     {
177         \exp_not:c { tex_ \tl_trim_spaces:n {#1} :D }
178         \__hook_make_prefixes:w /

```

```

179     }
180 }

```

(End definition for `_hook_patch_expand_redefine:NNnn` and `_hook_make_prefixes:w.`)

4.5.2 Patching by retokenization

At this point we've drained the possibilities of patching a command by expansion-and-redefinition, so we have to resort to patching by retokenizing the command. Patching by retokenization is done by getting the `\meaning` of the command, doing the necessary manipulations on the generated string, and the retokenizing that again by using `\scantokens`.

Patching by retokenization is definitely a riskier business, because it relies that the tokens printed by `\meaning` produce the exact same tokens as the ones in the original definition. That is, the catcode régime must be exactly(ish) the same, and there is no way of telling except by trial and error.

`_hook_retokenize_patch:Nnn` This is the macro that will control the whole process. First we'll try out one final, rather trivial case, of a command with no arguments; that is, a token list. This case can be patched with the expand-and-redefine routine but it has to be the very last case tested for, because most (all?) robust commands start with a top-level macro with no arguments, so testing this first would short-circuit `\robust@command@act` and the top-level macros would be incorrectly patched. In that case, we just check if the `\cs_argument_spec:N` is empty, and call `_hook_patch_expand_redefine:NNnn`.

```

181 \cs_new_protected:Npn \_hook_retokenize_patch:Nnn #1 #2 #3
182 {
183   \_hook_patch_debug:x { ..~command~can~only~be~patched~by~rescanning }
184   \str_if_eq:eeTF { \cs_argument_spec:N #1 } { }
185     { \_hook_patch_expand_redefine:NNnn \c_false_bool #1 {#2} {#3} }
186     {

```

Otherwise, we start the actual patching by retokenization job. The code calls `_hook_try_patch_with_catcodes:Nnnnw` with a different catcode setting:

- The current catcode setting;
- Switching the catcode of `@`;
- Switching the `expl3` syntax on or off;
- Both of the above.

If patching succeeds, `_hook_try_patch_with_catcodes:Nnnnw` has the side-effect of patching the macro `#1` (which may be an internal from the command whose name is `#2`).

```

187   \tl_set:Nx \l__hook_tmpa_tl
188   {
189     \int_compare:nNnTF { \char_value_catcode:n {'@ } } = { 12 }
190     { \exp_not:N \makeatletter } { \exp_not:N \makeatother }
191   }
192   \tl_set:Nx \l__hook_tmpb_tl
193   {
194     \bool_if:NTF \l__kernel_expl_bool
195     { \ExplSyntaxOff } { \ExplSyntaxOn }
196   }

```



```

197     \use:x
198     {
199         \exp_not:N \__hook_try_patch_with_catcodes:Nnnnw
200         \exp_not:n { #1 {#2} {#3} }
201         { \prg_do_nothing: }
202         { \exp_not:V \l__hook_tmpa_tl } % @
203         { \exp_not:V \l__hook_tmpb_tl } % _:
204         {
205             \exp_not:V \l__hook_tmpa_tl    % @
206             \exp_not:V \l__hook_tmpb_tl    % _:
207         }
208     }
209     \q_recursion_tail \q_recursion_stop

```

If no catcode setting succeeds, give up and raise an error. The command isn't changed in any way in that case.

```

210     {
211         \msg_error:nnxx { hooks } { cant-patch }
212         { \c_backslash_str #2 } { retok }
213     }
214 }
215 }

```

(End definition for __hook_retokenize_patch:Nnn.)

__hook_try_patch_with_catcodes:Nnnnw

This function is a simple wrapper around __hook_cmd_if_scanable:NnTF and __hook_patch_retokenize:Nnnn if the former returns *true*, plus some debug messages.

```

216 \cs_new_protected:Npn \__hook_try_patch_with_catcodes:Nnnnw #1 #2 #3 #4
217 {
218     \quark_if_recursion_tail_stop_do:nn {#4} { \use:n }
219     \__hook_patch_debug:x { ++~trying-to~patch~by~retokenization }
220     \__hook_cmd_if_scanable:NnTF {#1} {#4}
221     {
222         \__hook_patch_debug:x { ++~macro~can~be~retokenized~cleanly }
223         \__hook_patch_debug:x { ==~retokenizing~macro~now }
224         \__hook_patch_retokenize:Nnnn #1 {#2} {#3} {#4}
225         \use_i_delimit_by_q_recursion_stop:nw \use_none:n
226     }
227     {
228         \__hook_patch_debug:x { ---macro~cannot~be~retokenized~cleanly }
229         \__hook_try_patch_with_catcodes:Nnnnw #1 {#2} {#3}
230     }
231 }

```

(End definition for __hook_try_patch_with_catcodes:Nnnnw.)

\kerneltmpDoNotUse

This is an oddity required to be safe (as safe as reasonably possible) when patching the command. The entirety of

<prefixes> \def <cs> <parameter text> {<replacement text>}

will go through \scantokens. The *<parameter text>* and *<replacement text>* are what we are trying to retokenize, so not much worry there. The other items, however, should “just work”, so some care is needed to not use too fancy catcode settings. Therefore we can't use an expl3-named macro for *<cs>*, nor the expl3 versions of \def or the *<prefixes>*. That

is why the definitions that will eventually go into `\scantokens` will use the oddly (but hopefully clearly)-named `\kerneltmpDoNotUse`:

```
232 \cs_new_eq:NN \kerneltmpDoNotUse !
```

PhO: Maybe this can be avoided by running the \langle parameter text \rangle and the \langle replacement text \rangle separately through `\scantokens` and then putting everything together at the end.

(End definition for `\kerneltmpDoNotUse`.)

`__hook_patch_required_catcodes:` Here are the catcode settings that are *mandatory* when retokenizing commands. These are the minimum necessary settings to perform the definitions: they identify control sequences, which must be escaped with `_0`, delimit the definition with `{_1` and `}_2`, and mark parameters with `#_6`. Everything else may be changed, but not these.

```
233 \cs_new_protected:Npn \__hook_patch_required_catcodes:
234 {
235   \char_set_catcode_escape:N \_
236   \char_set_catcode_group_begin:N \{
237   \char_set_catcode_group_end:N \}
238   \char_set_catcode_parameter:N \#
239   % \int_set:Nn \tex_endlinechar:D { -1 }
240   % \int_set:Nn \tex_newlinechar:D { -1 }
241 }
```

PhO: etoolbox sets the `\endlinechar` and `\newlinechar` when patching, but as far as I tested these didn't make much of a difference, so I left them out for now. Maybe `\newlinechar=-1` avoids a space token being added after the definition.

PhO: If the patching is split by \langle parameter text \rangle and \langle replacement text \rangle , then only `#` will have to stay in that list.

*PhO: Actually now that we patch `\UseHook{cmd/foo/before}`, all the tokens there need to have the right catcodes, so this list now includes all lowercase letters, *U* and *H*, the slash, and whatever characters in the command name... sigh...*

(End definition for `__hook_patch_required_catcodes:`.)

`__hook_cmd_if_scanable:NnTF` Here we'll do a quick test if the command being patched can in fact be retokenized with the specific catcode setting without changing in meaning. The test is straightforward:

1. apply `\meaning` to the command;
2. split the \langle prefixes \rangle , \langle parameter text \rangle and \langle replacement text \rangle and arrange them as

$$\langle$$
prefixes \rangle `\def\kerneltmpDoNotUse` \langle parameter text \rangle `{` \langle replacement text \rangle `}`
3. rescan that with the given catcode settings, and do the definition; then finally
4. compare `\kerneltmpDoNotUse` with the original command.

If both are `\ifx`-equal, the command can be safely patched.

```
242 \prg_new_protected_conditional:Npnn \__hook_cmd_if_scanable:Nn #1 #2 { TF }
243 {
244   \cs_set_eq:NN \kerneltmpDoNotUse \scan_stop:
245   \cs_set_eq:NN \__hook_tmp:w \scan_stop:
246   \use:x
247   {
248     \cs_set:Npn \__hook_tmp:w
```

```

249     #####1 \tl_to_str:n { macro: } #####2 -> #####3 \s__hook_mark
250     { #####1 \def \kerneltmpDoNotUse #####2 {#####3} }
251     \tl_set:Nx \exp_not:N \l__hook_tmpa_tl
252     { \exp_not:N \__hook_tmp:w \token_to_meaning:N #1 \s__hook_mark }
253   }
254   \tl_rescan:nV { #2 \__hook_patch_required_catcodes: } \l__hook_tmpa_tl
255   \token_if_eq_meaning:NNTF #1 \kerneltmpDoNotUse
256   { \prg_return_true: }
257   { \prg_return_false: }
258 }

```

(End definition for __hook_cmd_if_scanable:NnTF.)

__hook_patch_retokenize:Nnnn Then, if __hook_cmd_if_scanable:NnTF returned true, we can go on and patch the command.

```

259 \cs_new_protected:Npn \__hook_patch_retokenize:Nnnn #1 #2 #3 #4
260 {

```

Start off by making some things \relax to avoid lots of \noexpand below.

```

261   \cs_set_eq:NN \kerneltmpDoNotUse \scan_stop:
262   \cs_set_eq:NN \__hook_tmp:w \scan_stop:
263   \use:x
264   {

```

Now we'll define __hook_tmp:w such that it splits the \meaning of the macro (#1) into its three parts:

```

#####1. <prefixes>

#####2. <parameter text>

#####3. <replacement text>

```

and arrange that a complete definition, then place the before or after hooks around the <replacement text>: accordingly.

```

265   \cs_set:Npn \__hook_tmp:w
266   #####1 \tl_to_str:n { macro: } #####2 -> #####3 \s__hook_mark
267   {
268     #####1 \def \kerneltmpDoNotUse #####2
269     {
270       \str_if_eq:nnT {#3} { before }
271       { \token_to_str:N \UseHook { cmd / #2 / #3 } }
272     #####3
273     \str_if_eq:nnT {#3} { after }
274     { \token_to_str:N \UseHook { cmd / #2 / #3 } }
275   }
276 }

```

Now we just have to get the \meaning of the command being patched and pass it through the meat grinder above.

```

277   \tl_set:Nx \exp_not:N \l__hook_tmpa_tl
278   { \exp_not:N \__hook_tmp:w \token_to_meaning:N #1 \s__hook_mark }
279 }

```

Now rescan with the given catcode settings (overridden by the `__hook_patch_required_catcodes:`), and implicitly (by using the rescanned token list) carry out the definition from above.

```
280 \tl_rescan:nV { #4 \__hook_patch_required_catcodes: } \l__hook_tmpa_tl
```

And to close, copy the newly-defined command into the old name and the patching is finally completed:

```
281 \cs_set_eq:NN #1 \kerneltmpDoNotUse
282 }
```

(End definition for `__hook_patch_retokenize:Nnnn.`)

4.6 Messages

```
283 \msg_new:nnnn { hooks } { wrong-cmd-hook }
284 {
285   Generic-hook~‘cmd/#1/#2’~is~invalid.
286 %   The~hook~should~be~‘cmd/#1/before’~or~‘cmd/#1/after’.
287 }
288 {
289   You~tried~to~add~a~generic~hook~to~command~\iow_char:N \#1,~but~‘#2’~
290   is~an~invalid~component.~Only~‘before’~or~‘after’~are~allowed.
291 }
292 \msg_new:nnnn { hooks } { cant-patch }
293 {
294   Generic-hooks~cannot~be~added~to~‘#1’.
295 }
296 {
297   You~tried~to~add~a~hook~to~‘#1’,~but~LaTeX~was~unable~to~
298   patch~the~command~because~it~\__hook_unpatchable_cases:n {#2}.
299 }
300 \cs_new:Npn \__hook_unpatchable_cases:n #1
301 {
302   \str_case:nn {#1}
303   {
304     { undef } { doesn’t~exist }
305     { macro } { is~not~a~macro }
306     { expl3 } { is~a~private~expl3~macro }
307     { retok } { can’t~be~retokenized~cleanly }
308   }
309 }
310 <latexrelease> \IncludeInRelease{0000/00/00}{ltcmdhooks}%
311 <latexrelease> {The~hook~management~system~for~commands}
312 <latexrelease>
```

The command `__hook_cmd_begindocument_code:` is used in an internal hook, so we need to make sure it has a harmless definition after rollback as that will not remove it from the kernel hook.

```
313 <latexrelease> \cs_set_eq:NN \__hook_cmd_begindocument_code: \prg_do_nothing:
314 <latexrelease>
315 <latexrelease> \EndModuleRelease
316 \ExplSyntaxOff
317 </2ekernel | latexrelease>
318 <@@=>
```

File j

lalloc.dtx

1 Counters

This section deals with counter and other variable allocation.

```
1 <*2ekernel>
```

The following are from plain T_EX:

`\z@` A zero dimen or number. It's more efficient to write `\parindent\z@` than `\parindent 0pt`.

`\@ne` The number 1.

`\m@ne` The number -1 .

`\tw@` The number 2.

`\sist@@n` The number 16.

`\@m` The number 1000.

`\@MM` The number 20000.

`\@xxxii` The constant 32.

```
2 \chardef\@xxxii=32
```

(End definition for \@xxxii.)

`\@Mi` Constants 10001–10004.

```
\@Mii 3 \mathchardef\@Mi=10001
```

```
\@Miii 4 \mathchardef\@Mii=10002
```

```
\@Miv 5 \mathchardef\@Miii=10003
```

```
6 \mathchardef\@Miv=10004
```

(End definition for \@Mi and others.)

`\@tempcnta` Scratch count registers used by L^AT_EX kernel commands.

```
\@tempcntb 7 \newcount\@tempcnta
```

```
8 \newcount\@tempcntb
```

(End definition for \@tempcnta and \@tempcntb.)

`\if@tempswa` General boolean switch used by L^AT_EX kernel commands.

```
9 \newif\if@tempswa
```

(End definition for \if@tempswa.)

`\@tempdima` Scratch dimen registers used by L^AT_EX kernel commands.

```
\@tempdimb 10 \newdimen\@tempdima
```

```
\@tempdimc 11 \newdimen\@tempdimb
```

```
12 \newdimen\@tempdimc
```

(End definition for \@tempdima, \@tempdimb, and \@tempdimc.)

`\@tempboxa` Scratch box register used by L^AT_EX kernel commands.
¹³ `\newbox\@tempboxa`
(End definition for \@tempboxa.)

`\@tempskipa` Scratch skip registers used by L^AT_EX kernel commands.
`\@tempskipb` ¹⁴ `\newskip\@tempskipa`
¹⁵ `\newskip\@tempskipb`
(End definition for \@tempskipa and \@tempskipb.)

`\@temptokena` Scratch token register used by L^AT_EX kernel commands.
¹⁶ `\newtoks\@temptokena`
(End definition for \@temptokena.)

`\@flushglue` Glue used for `\right-` & `\leftskip = 0pt plus 1fil`
¹⁷ `\newskip\@flushglue \@flushglue = 0pt plus 1fil`
(End definition for \@flushglue.)
¹⁸ `</2ekernel>`

File k

ltcntrl.dtx

1 Program control structure

This section defines a number of control structure macros, such as while-loops and for-loops.

Historical L^AT_EX 2.09 comments (not necessarily accurate any more):

```

1 \*2kernel)
2 \message{control,}

\@whilenum TEST \do {BODY}
\@whiledim TEST \do {BODY} : These implement the loop
    while TEST do BODY od
    where TEST is a TeX \ifnum or \ifdim test, respectively.
    They are optimized for the normal case of TEST initially false.

\@whilesw SWITCH \fi {BODY} : Implements the loop
    while SWITCH do BODY od
    Optimized for normal case of SWITCH initially false.

\@for NAME := LIST \do {BODY} : Assumes that LIST expands to A1,A2,
    ... ,An .
    Executes BODY n times, with NAME = Ai on the i-th iteration.
    Optimized for the normal case of n = 1. Works for n=0.

\@tfor NAME := LIST \do {BODY}
    if, before expansion, LIST = T1 ... Tn where each Ti is a
    token or {...}, then executes BODY n times, with NAME = Ti
    on the i-th iteration. Works for n=0.

NOTES: 1. These macros use no \@temp sequences.
        2. These macros do not work if the body contains anything that
        looks syntactically to TeX like an improperly balanced \if
        \else \fi.

\@whilenum TEST \do {BODY} ==
BEGIN
    if TEST
    then BODY
        \@iwhilenum{TEST \relax BODY}
    END

\@iwhilenum {TEST BODY} ==
BEGIN
    if TEST
    then BODY
        \@nextwhile = def(\@iwhilenum)

```

```

        else \@nextwhile = def(\@whilenoop)
      fi
      \@nextwhile {TEST BODY}
    END

\@whilesw SWITCH \fi {BODY} ==
BEGIN
  if SWITCH
    then BODY
      \@iwhilesw {SWITCH BODY}\fi
    fi
  END

\@iwhilesw {SWITCH BODY} \fi ==
BEGIN
  if SWITCH
    then BODY
      \@nextwhile = def(\@iwhilesw)
    else \@nextwhile = def(\@whileswnoop)
    fi
  \@nextwhile {SWITCH BODY} \fi
END

```

End of historical L^AT_EX 2.09 comments.

```

\@whilenoop
\@whilenum      3 \long\def\@whilenum#1\do #2{\ifnum #1\relax #2\relax\@iwhilenum{#1\relax
\@iwhilenum      4 #2\relax}\fi}
5 \long\def\@iwhilenum#1{\ifnum #1\expandafter\@iwhilenum
6   \else\expandafter\@gobble\fi{#1}}

```

(End definition for \@whilenoop, \@whilenum, and \@iwhilenum.)

```

\@whiledim
\@iwhiledim      7 \long\def\@whiledim#1\do #2{\ifdim #1\relax#2\@iwhiledim{#1\relax#2}\fi}
8 \long\def\@iwhiledim#1{\ifdim #1\expandafter\@iwhiledim
9   \else\expandafter\@gobble\fi{#1}}

```

(End definition for \@whiledim and \@iwhiledim.)

```

\@whileswnoop
\@whilesw      10 \long\def\@whilesw#1\fi#2{#1#2\@iwhilesw{#1#2}\fi\fi}
\@iwhilesw      11 \long\def\@iwhilesw#1\fi{#1\expandafter\@iwhilesw
12   \else\@gobbletwo\fi{#1}\fi}

```


(End definition for \@whileswnoop, \@whilesw, and \@iwhilesw.)

Historical L^AT_EX 2.09 comments (not necessarily accurate any more):

```
\@for NAME := LIST \do {BODY} ==
  BEGIN \@forloop expand(LIST),\@nil,\@nil \@@ NAME {BODY} END
```

```
\@forloop CAR, CARCDR, CDRCDR \@@ NAME {BODY} ==
  BEGIN
    NAME = CAR
    if def(NAME) = def(\@nnil)
      else BODY;
      NAME = CARCDR
      if def(NAME) = def(\@nnil)
        else BODY
          \@iforloop CDRCDR \@@ NAME \do {BODY}
        fi
      fi
  END
```

```
\@iforloop CAR, CDR \@@ NAME {BODY} =
  NAME = CAR
  if def(NAME) = def(\@nnil)
    then \@nextwhile = def(\@fornoop)
    else BODY ;
      \@nextwhile = def(\@iforloop)
    fi
  \@nextwhile name cdr {body}
```

```
\@tfor NAME := LIST \do {BODY}
  = \@tforloop LIST \@nil \@@ NAME {BODY}
```

```
\@tforloop car cdr \@@ name {body} =
  name = car
  if def(name) = def(\@nnil)
    then \@nextwhile == \@fornoop
    else body ;
      \@nextwhile == \@forloop
    fi
  \@nextwhile name cdr {body}
```

End of historical L^AT_EX 2.09 comments.

\@nnil

```
13 \def\@nnil{\@nil}
```

(End definition for \@nnil.)

\@empty

```
14 \def\@empty{}
```

(End definition for \@empty.)

```

\@fornoop
15 \long\def\@fornoop#1\@@#2#3{}
(End definition for \@fornoop.)

\@for
16 \long\def\@for#1:=#2\do#3{%
17   \expandafter\def\expandafter\@fortmp\expandafter{#2}%
18   \ifx\@fortmp\@empty \else
19     \expandafter\@forloop#2,\@nil,\@nil\@@#1{#3}\fi}
(End definition for \@for.)

\@forloop
20 \long\def\@forloop#1,#2,#3\@@#4#5{\def#4{#1}\ifx #4\@nnil \else
21   #5\def#4{#2}\ifx #4\@nnil \else#5\@iforloop #3\@@#4{#5}\fi\fi}
(End definition for \@forloop.)

\@iforloop
22 \long\def\@iforloop#1,#2\@@#3#4{\def#3{#1}\ifx #3\@nnil
23   \expandafter\@fornoop \else
24   #4\relax\expandafter\@iforloop\fi#2\@@#3{#4}}
(End definition for \@iforloop.)

\@tfor
25 \def\@tfor#1:={\@tf@r#1 }
26 \long\def\@tf@r#1#2\do#3{\def\@fortmp{#2}\ifx\@fortmp\space\else
27   \@tforloop#2\@nil\@nil\@@#1{#3}\fi}
28 \long\def\@tforloop#1#2\@@#3#4{\def#3{#1}\ifx #3\@nnil
29   \expandafter\@fornoop \else
30   #4\relax\expandafter\@tforloop\fi#2\@@#3{#4}}
(End definition for \@tfor.)

\@break@tfor Break out of a \@tfor loop. This should be called inside the scope of an \if. See
\@iffileonpath for an example.
31 \long\def\@break@tfor#1\@@#2#3{\fi\fi}
(End definition for \@break@tfor.)

\@removeelement Removes an element from a comma-separated list and puts it into a control se-
quence, called as \@removeelement{<element>}{<list>}{<cs>}. Due to the implemen-
tation method the <element> is not allowed to contain braces.
32 \def\@removeelement#1#2#3{%
33   \def\reserved@a##1,#1,##2\reserved@a{##1,##2\reserved@b}%
34   \def\reserved@b##1,\reserved@b##2\reserved@b{%
35     \ifx,##1\@empty\else##1\fi}%
36   \edef#3{%
37     \expandafter\reserved@b\reserved@a,#2,\reserved@b,#1,\reserved@a}}
(End definition for \@removeelement.)
38 </2kernel>

```

File 1

lterror.dtx

1 Error handling and tracing

This section defines L^AT_EX's error commands.

```
1 <*2kernel>
```

The ‘2kernel’ code ensures that a `\usepackage{autoerr}` is essentially ignored if a ‘full’ format is being used that has the error messages already in the format.

These days we don't support autoloading approach any longer, but this part bit is kept in case it is used in old documents.

```
2 \expandafter\let\csname ver@autoerr.sty\endcsname\fmtversion
```

1.1 General commands

\MessageBreak This command prints a new-line inside a message, followed by a continuation line begun with `\@msg@continuation`. Normally it is defined to be `\relax`, but inside messages, it is let to `\@message@break`.

```
3 \let\MessageBreak\relax
```

(End definition for \MessageBreak.)

\GenericInfo This takes two arguments: a continuation and a message, and sends the result to the log file.

```
4 \DeclareRobustCommand{\GenericInfo}[2]{%
5   \begingroup
6     \def\MessageBreak{^^J#1}%
7     \set@display@protect
8     \immediate\write\m@ne{#2\on@line.}%
9   \endgroup
10 }
```

(End definition for \GenericInfo.)

\GenericWarning This takes two arguments: a continuation and a message, and sends the result to the screen.

```
11 \DeclareRobustCommand{\GenericWarning}[2]{%
12   \begingroup
13     \def\MessageBreak{^^J#1}%
14     \set@display@protect
15     \immediate\write\@unused{^^J#2\on@line.^^J}%
16   \endgroup
17 }
```

(End definition for \GenericWarning.)

`\GenericError` This macro takes four arguments: a continuation, an error message, where to go for further information, and the help information. It displays the error message, and sets the error help (the result of typing `h` to the prompt), and does a horrible hack to turn the last context line (which by default is the only context line) into just three dots. This could be made more efficient.

```

18 \bgroup
19 \lccode'\@=' \ %
20 \lccode'\~=' \ %
21 \lccode'\}=' \ %
22 \lccode'\{=' \ %
23 \lccode'\T=' \T%
24 \lccode'\H=' \H%
25 \catcode'\ =11\relax%
26 \lowercase{%
27 \egroup%
```

Unfortunately \TeX versions older than 3.141 have a bug which means that `^^J` does not force a linebreak in `\message` and `\errmessage` commands. So for these old \TeX 's we use `\typeout` to produce the message, and then have an empty `\errmessage` command. This causes an extra line of the form

.

To appear on the terminal, but if you do not like it, you can always upgrade your \TeX ! In order for your format to use this version, you must define the macro `\TeXversion` to be the version number, e.g., 3.14 of the underlying \TeX . See the comments in `ltdircheck.dtx`.

```

28 \dimen@ifx\TeXversion\@undefined4\else\TeXversion\fi\p@%
29 \ifdim\dimen@>3.14\p@%
    First the 'standard case'.
30 \DeclareRobustCommand{\GenericError}[4]{%
31 \begingroup%
32 \immediate\write\@unused{}%
33 \def\MessageBreak{^^J}%
34 \set@display@protect%
35 \edef%
36 % %<-----do not delete this space!----->%
37 \@err@
38 {{#4}}%
39 \errhelp
40 % %<-----do not delete this space!----->%
41 \@err@
42 \let
43 % %<-----do not delete this space!----->%
44 \@err@
45 \@empty
46 \def\MessageBreak{^^J#1}%
47 \def~{\errmessage{%
48 #2.^^J^^J%
49 #3^^J%
50 Type H <return> for immediate help%
51 % %<-----do not delete this space!----->%
52 \@err@
```

```

53 }}%
54 ~%
55 \endgroup}%
56 \else%
    Secondly the version for old TEX's.
57 \DeclareRobustCommand{\GenericError}[4]{%
58 \begingroup%
59 \immediate\write\@unused{}%
60 \def\MessageBreak{^^J}%
61 \set@display@protect%
62 \edef%
63 %    %<-----do not delete this space!----->%
64 \@err@
65 {{#4}}%
66 \errhelp
67 %    %<-----do not delete this space!----->%
68 \@err@
69 \let
70 %    %<-----do not delete this space!----->%
71 \@err@
72 \errmessage
73 \def\MessageBreak{^^J#1}%
74 \def~{\typeout{! %
75 #2.^^J^^J%
76 #3^^J%
77 Type H <return> for immediate help.}%
78 %    %<-----do not delete this space!----->%
79 \@err@
80 {}}%
81 ~%
82 \endgroup}%
83 \fi}%

```

(End definition for \GenericError.)

<code>\PackageError</code> <code>\PackageWarning</code> <code>\PackageWarningNoLine</code> <code>\PackageInfo</code> <code>\ClassError</code> <code>\ClassWarning</code> <code>\ClassWarningNoLine</code> <code>\ClassInfo</code>	<p>These commands are intended for use by package and class writers, to give information to authors. The syntax is:</p> <pre> \PackageError{<package>}{<error>}{<help>} \PackageWarning{<package>}{<warning>} \PackageWarningNoLine{<package>}{<warning>} \PackageInfo{<package>}{<info>} </pre>
--	--

and similarly for classes. The `Error` commands print the `<error>` message, and present the interactive prompt; if the author types `h`, then the `<help>` information is displayed. The `Warning` commands produce a warning but do not present the interactive prompt. The `WarningNoLine` commands do the same, but don't print the input line number. The `Info` commands write the message to the `log` file. Within the messages, the command `\MessageBreak` can be used to break a line, `\protect` can be used to protect command names, and `\space` is a space, for example:

```

\newcommand{\foo}{F00}
\PackageWarning{ethel}{%
  Your hovercraft is full of eels,\MessageBreak
  and \protect\foo\space is \foo}

```

produces:

```

Package ethel warning: Your hovercraft is full of eels,
(ethel)                and \foo is F00 on input line 54.

```

```

84 \gdef\PackageError#1#2#3{%
85   \GenericError{%
86     (#1)\@spaces\@spaces\@spaces\@spaces
87   }{%
88     Package #1 Error: #2%
89   }{%
90     See the #1 package documentation for explanation.%
91   }{#3}%
92 }
93 \def\PackageWarning#1#2{%
94   \GenericWarning{%
95     (#1)\@spaces\@spaces\@spaces\@spaces
96   }{%
97     Package #1 Warning: #2%
98   }%
99 }
100 \def\PackageWarningNoLine#1#2{%
101   \PackageWarning{#1}{#2\@gobble}%
102 }
103 \def\PackageInfo#1#2{%
104   \GenericInfo{%
105     (#1) \@spaces\@spaces\@spaces
106   }{%
107     Package #1 Info: #2%
108   }%
109 }
110 \gdef\ClassError#1#2#3{%
111   \GenericError{%
112     (#1) \space\@spaces\@spaces\@spaces
113   }{%
114     Class #1 Error: #2%
115   }{%
116     See the #1 class documentation for explanation.%
117   }{#3}%
118 }
119 \def\ClassWarning#1#2{%
120   \GenericWarning{%
121     (#1) \space\@spaces\@spaces\@spaces
122   }{%
123     Class #1 Warning: #2%
124   }%
125 }
126 \def\ClassWarningNoLine#1#2{%

```

```

127   \ClassWarning{#1}{#2\@gobble}%
128 }
129 \def\ClassInfo#1#2{%
130   \GenericInfo{%
131     (#1) \space\space\@spaces\@spaces
132   }{%
133     Class #1 Info: #2%
134   }%
135 }

```

(End definition for \PackageError and others.)

```

\@latex@error Errors and other info, for use in the LATEX core.
\@latex@warning
\@latex@warning@no@line
\@latex@info
\@latex@info@no@line
136 \gdef\@latex@error#1#2{%
137   \GenericError{%
138     \space\space\space\@spaces\@spaces\@spaces
139   }{%
140     LaTeX Error: #1%
141   }{%
142     See the LaTeX manual or LaTeX Companion for explanation.%
143   }{#2}%
144 }
145 \def\@latex@warning#1{%
146   \GenericWarning{%
147     \space\space\space\@spaces\@spaces\@spaces
148   }{%
149     LaTeX Warning: #1%
150   }%
151 }
152 \def\@latex@warning@no@line#1{%
153   \@latex@warning{#1\@gobble}}
154 \def\@latex@info#1{%
155   \GenericInfo{%
156     \@spaces\@spaces\@spaces
157   }{%
158     LaTeX Info: #1%
159   }%
160 }
161 \def\@latex@info@no@line#1{%
162   \@latex@info{#1\@gobble}}

\@font@warning and \@font@info are defined later since they have to be redefined
by the tracefnt package.

def\@font@warning#1{%
  \GenericWarning{%
    {(font)\@spaces\@spaces}%
    {Font Warning: #1}%
  }
}
def\@font@info#1{%
  \GenericInfo{%
    (font)\space\@spaces
  }{%

```

```

        Font Info: #1%
    }%
}

(End definition for \@latex@error and others.)

\c@errorcontextlines \errorcontextlines as a LATEX counter, so that it may be manipulated with \setcounter
(once it is defined :-)
163 \let\c@errorcontextlines\errorcontextlines
164 \c@errorcontextlines=-1

(End definition for \c@errorcontextlines.)

\on@line The message ‘ on input line n’.
165 \def\on@line{ on input line \the\inputlineno}

(End definition for \on@line.)

\@warning Older LATEX messages. For the moment, these \let to the new message commands. They
\@warning may be changed later, once only obsolete packages and classes contain them.
\@latexerr
166 \let\@warning\@latex@warning
167 \let\@warning\@latex@warning@no@line
168 \global\let\@latexerr\@latex@error

(End definition for \@warning, \@warning, and \@latexerr.)

\@spaces Four spaces.
169 \def\@spaces{\space\space\space\space}

(End definition for \@spaces.)

```

1.2 Specific errors

```

\@eha The more common error help messages.
\@ehb
170 \gdef\@eha{%
\@ehc
171 Your command was ignored.\MessageBreak
\@ehd
172 Type \space I <command> <return> \space to replace it %
173 with another command,\MessageBreak
174 or \space <return> \space to continue without it.}
175 \gdef\@ehb{%
176 You’ve lost some text. \space \@ehc}
177 \gdef\@ehc{%
178 Try typing \space <return> %
179 \space to proceed.\MessageBreak
180 If that doesn’t work, type \space X <return> \space to quit.}
181 \gdef\@ehd{%
182 You’re in trouble here. \space \@ehc}

(End definition for \@eha and others.)

```


\@notdefinable Error message generated in \@ifdefinable from calls to one of the commands \newcommand, \newlength or \newtheorem specifying an already-defined command name or one that begins \end....

```

183 \gdef\@notdefinable{%
184   \@latex@error{%
185     Command \@backslashchar\reserved@a\space
186     already defined.\MessageBreak
187     Or name \@backslashchar\@qend... illegal,
188     see p.192 of the manual}\@eha}

```

(End definition for \@notdefinable.)

\@nolnerr Generated by \newline and \\ when called in vertical mode.

```

189 \gdef\@nolnerr{%
190   \@latex@error{There's no line here to end}\@eha}

```

(End definition for \@nolnerr.)

\@nocounterr Generated by \setcounter, \addtocounter or \newcounter if applied to an undefined counter <cnt>.

Obsolete error message generated in L^AT_EX2.09 by \setcounter, \addtocounter or \newcounter for undefined counter. DO NOT use for L^AT_EX2_ε it MIGHT vanish! Use \@nocounterr{<cnt>} instead.

```

191 \gdef\@nocounterr#1{%
192   \@latex@error{No counter '#1' defined}\@eha}
193 \gdef\@nocnterr{\@nocounterr?}

```

(End definition for \@nocounterr and \@nocnterr.)

\@ctrerr Called when trying to print the value of a counter numbered by letters that's greater than 26.

```

194 \gdef\@ctrerr{%
195   \@latex@error{Counter too large}\@ehb}

```

(End definition for \@ctrerr.)

\@nodocument Error produced if paragraphs are typeset in the preamble.

```

196 \gdef\@nodocument{%
197   \@latex@error{Missing \protect\begin{document}}\@ehd}

```

(End definition for \@nodocument.)

\@badend Called by \end that doesn't match its \begin. RmS 1992/08/24: added code to \@badend to display position of non-matching \begin. FMi 1993/01/14: missing space added.

```

198 \gdef\@badend#1{%
199   \@latex@error{\protect\begin{\@currenvir}\@currenvline
200     \space ended by \protect\end{#1}}\@eha}

```

(End definition for \@badend.)

\@badmath Called by \[, \], \(or \) when used in wrong mode.

```

201 \gdef\@badmath{%
202   \@latex@error{Bad math environment delimiter}\@eha}

```

(End definition for \@badmath.)

\@toodeep Called by a list environment nested more than six levels deep, or an enumerate or itemize nested more than four levels.

```
203 \gdef\@toodeep{%  
204   \@latex@error{Too deeply nested}\@ehd}
```

(End definition for \@toodeep.)

\@badpoptabs Called by \endtabbing when not enough \poptabs have occurred, or by \poptabs when too many have occurred.

```
205 \gdef\@badpoptabs{%  
206   \@latex@error{\protect\pushtabs\space and \protect\poptabs  
207     \space don't match}\@ehd}
```

(End definition for \@badpoptabs.)

\@badtab Called by \>, \+ , \- or \< when stepping to an undefined tab.

```
208 \gdef\@badtab{%  
209   \@latex@error{Undefined tab position}\@ehd}
```

(End definition for \@badtab.)

\@preamerr This error is special: it appears in places where we normally have to \protect expansions. However, to prevent a protection of the error message itself (which would result in the message getting printed not issued on the terminal) we need to locally reset \protect to \relax.

```
210 \gdef\@preamerr#1{%  
211   \begingroup  
212     \let\protect\relax  
213     \@latex@error{\ifcase #1 Illegal character\or  
214       Missing @-exp\or Missing p-arg\fi\space  
215       in array arg}\@ehd  
216   \endgroup}
```

(End definition for \@preamerr.)

\@badlinearg Occurs in \line and \vector command when a bad slope argument is encountered.

```
217 \gdef\@badlinearg{%  
218   \@latex@error{%  
219     Bad \protect\line\space or \protect\vector  
220     \space argument}\@ehb}
```

(End definition for \@badlinearg.)

\@parmoderr Occurs in a float environment or a \marginpar when encountered in inner vertical mode.

```
221 \gdef\@parmoderr{%  
222   \@latex@error{Not in outer par mode}\@ehb}
```

(End definition for \@parmoderr.)

\@fltovf Occurs in float environment or \marginpar when there are no more free boxes for storing floats.

```
223 \gdef\@fltovf{%  
224   \@latex@error{Too many unprocessed floats}\@ehb}
```

(End definition for \fltovf.)

\@latexbug Occurs in output routine. This is bad news.

```
225 \gdef\@latexbug{%  
226   \@latex@error{This may be a LaTeX bug}{Call for help}}
```

(End definition for \@latexbug.)

\@badcrerr This error was removed and replaced by \@nolnerr.

```
227 %\def\@badcrerr {\@latex@error{Bad use of \protect\\}\@ehc}
```

(End definition for \@badcrerr.)

\@noitemerr \addvspace or \addpenalty was called when not in vmode. Probably caused by a missing \item.

```
228 \gdef\@noitemerr{%  
229   \@latex@error{Something's wrong--perhaps a missing %  
230     \protect\item}\@ehc}
```

(End definition for \@noitemerr.)

\@notprerr A command that can be used only in the preamble appears after the command \begin{document}.

```
231 \gdef\@notprerr{%  
232   \@latex@error{Can be used only in preamble}\@eha}
```

(End definition for \@notprerr.)

\@inmatherr Issued by commands that don't work correctly within math (like \item). There is no real error recovery happening, e.g., the user might get additional errors afterwards.

```
233 \gdef\@inmatherr#1{%  
234   \relax  
235   \ifmmode  
236     \@latex@error{Command \protect#1 invalid in math mode}\@ehc  
237     \fi}
```

(End definition for \@inmatherr.)

\@invalidchar An error for use with invalid characters. This is commented out, since we decided to use catcode 15 instead.

```
238 %\def\@invalidchar{\@latex@error{Invalid character in input}\@ehc}
```

(End definition for \@invalidchar.)

As well as the above error commands some error messages are directly coded to save space. The messages already present in L^AT_EX 2.09 include:

Environment --- undefined

Issued by \begin for undefined environment.

Tab overflow

Occurs in \= when maximum number of tabs exceeded.

\< in mid line

Occurs in \< when it appears in middle of line.

Float(s) lost

In output routine, caused by a float environment or \marginpar occurring in inner vertical mode.

1.3 Tracing

The `trace` package implements the commands `\traceon` and `\traceoff` that work similar to `\tracingall` but skip certain code blocks that produce a lot of tracing output being of no interest during debugging (for example loading a font). Code blocks that should be hidden during tracing need to be surrounded by the macros `\conditionally@traceoff` and `\conditionally@traceon`.

For the kernel code the `trace` package then redefines a number of macros to include this tracing support.

However, in order to allow any macro package to react to `\traceon` we also provide dummy definitions for the two commands in the kernel so that they can be used by external packages without the need to distinguish between `trace` being loaded or not.

```
\conditionally@traceon  These are only dummy definitions. For details see the trace package.
\conditionally@traceoff
239 \let\conditionally@traceon\@empty
240 \let\conditionally@traceoff\@empty

(End definition for \conditionally@traceon and \conditionally@traceoff.)
241 </2ekernel>
```

File m

ltpar.dtx

1 Paragraphs

This section of the kernel declares the commands used to set `\par` and `\everypar` when ever their function needs to be changed for a long time.

This file here describes the interfaces that have been in the kernel forever, used to implement the scenarios described below. They remain valid but are now augmented in the next file (`ltpara.dtx`) to add hooks to paragraphs. At some point we will consolidate the two files further.

There are two situations in which `\par` may be changed:

- Long-term changes, in which the new value is to remain in effect until the current environment is left. The environments that change `\par` in this way are the following:
 - All list environments (itemize, quote, etc.)
 - Environments that turn `\par` into a noop: tabbing, array and tabular.
- Temporary changes, in which `\par` is restored to its previous value the next time it is executed. The following are all such uses.
 - `\end` when preceded by `\@endparenv`, which is called by `\endtrivlist`
 - The mechanism for avoiding page breaks and getting the spacing right after section heads.

1.1 Implementation

`\@setpar` To permit the proper interaction of these two situations, long-term changes are made by the `\@setpar{<VAL>}` command. It's function is:

To set `\par`. It `\def`'s `\par` and `\@par` to `<VAL>`.

`\@restorepar` Short-term changes are made by the usual `\def\par` commands. The original values are restored after a short-term change by the `\@restorepar` commands.

`\@@par` `\@@par` always is defined to be the original TeX `\par`.

`\everypar` `\everypar` is changed only for the short term. Whenever `\everypar` is set non-null, it should restore itself to null when executed.

The following commands change `\everypar` in this way:

- `\item`
- `\end` when preceded by `\@endparenv`, which is called by `\endtrivlist`
- `\minipage`

When dealing with `\par` and `\everypar` remember the following two warnings:

1. Commands that make short-term changes to `\par` and `\everypar` must take account of the possibility that the new commands and the ones that do the restoration may be executed inside a group. In particular, `\everypar` is executed inside a group whenever a new paragraph begins with a left brace. The `\everypar` command that restores its definition should be local to the current group (in case the command is inside a minipage used inside someplace where `\everypar` has been redefined). Thus, if `\everypar` is redefined to do an `\everypar{}` it could take several executions of `\everypar` before the restoration “holds”. This usually causes no problem. However, to prevent the extra executions from doing harm, use a global switch to keep anything harmful in the new `\everypar` from being done twice.
2. Commands that change `\everypar` should remember that `\everypar` might be supposed to set the following switches false:

- `@nobreak`
- `@minipage`

they should do the setting if necessary.

```

1  \if*2kernel
2  \message{par,}

```

`\@setpar` Initiate a long-term change to `\par`.

```

\@par 3 \def\@setpar#1{\def\par{#1}\def\@par{#1}}

```

The default definition of `\@par` will ensure that if `\@restorepar` defines `\par` to execute `\@par` it will redefine itself to the primitive `\@@par` after one iteration.

```

4 \def\@par{\let\par\@@par\par}

```

(End definition for \@setpar and \@par.)

`\@restorepar` Restore from a short-term change to `\par`.

```

5 \def\@restorepar{\def\par{\@par}}
6 \if*2kernel

```

(End definition for \@restorepar.)

File n

ltpara.dtx

Abstract

This code defines four special kernel hooks to support paragraph tagging as well as four public hooks which can be occasionally useful.

Contents

1 Introduction

The building of paragraphs in the T_EX engine(s) has a number of peculiarities that makes it on one hand fairly flexible but on the other hand somewhat awkward to control or reliably to extend. Thus to better understand the code below we start with a brief introduction of the mechanism; for more details refer to the T_EXbook [?, chap. 14] (for the full truth you may even have to study the program code).

1.1 The default processing done by the engine

T_EX automatically starts building a paragraph when it is currently in vertical mode and encounters anything that can only live in horizontal mode. Most often this is a character, but there are also many commands that can be used only in horizontal mode. If any of them is encountered, T_EX will immediately back up (i.e., the character or command is read later again), adds a `\parskip` glue to the current vertical list unless the list is empty, switches to horizontal mode, starts its special “start of paragraph processing” and only then rereads the character or command that caused the mode change.¹⁴

This “start of paragraph processing” first adds an empty box at the start of the horizontal list of width `\parindent` (which represents the paragraph indentation) unless the paragraph was started with `\noindent` in which case no such box is added¹⁵. It then reads and processes all tokens stored in the special engine token register `\everypar`. After that it reads and processes whatever has caused the paragraph to start.

Thus out of the box, T_EX offers the possibility to put some special code into `\everypar` to gain control at (more or less) the start of the paragraph. For example, in LaTeX and a number of packages, special code like the following is sometimes used:

```
\everypar{{\setbox\z@\lastbox}\everypar{}} ...}
```

This removes the paragraph indentation box again (that was already placed by T_EX), then resets `\everypar` so that it doesn’t do anything on the next paragraph start and then does whatever it wants to do, e.g., in an `\item` of a list it will typeset the label in front of the paragraph text. However, there is only one such `\everypar` token register and if different packages and/or the kernel all attempt to add their own code here, coordination is very difficult if not impossible.

¹⁴Already not quite true: the command `\noindent` starts the paragraph but influences the special processing by suppressing the paragraph indentation box normally inserted by it.

¹⁵That’s a bit different from placing a zero-sized box!

The process when the paragraph ends has different mechanisms and interfaces. A paragraph ends when the engine primitive `\par` is called while `TeX` is in unrestricted horizontal mode, i.e., is building a paragraph. At other times this primitive does nothing or generates an error depending on the mode `TeX` is in, e.g., the `\par` in `\hbox{a\par b}` is ignored, but `$a\par b$` would complain.

If this primitive ends the paragraph it does some special “end of horizontal list” processing, then calls `TeX` paragraph builder that breaks the horizontal list into lines then these lines are added as boxes to the enclosing vertical list and `TeX` returns to vertical mode.

This `\par` command can be given explicitly, but there are also situations in which `TeX` is generating it on the fly. Most often this happens when `TeX` encounters a blank line which is automatically changed to a `\par` command which is then executed. The other possibility is that `TeX` encounters a command which is incompatible with horizontal processing, e.g., `\vskip` (a request for adding vertical space). In such case it silently backs up, and inserts a `\par` in the hope that this gets it out of horizontal mode and makes the offending command acceptable.

The important point to note here is that `TeX` really inserts the command `\par` which can be redefined. Thus, it may not have its original “primitive” meaning and therefore may not end the horizontal list and call the paragraph builder. This approach offers some flexibility but also allows you to easily produce a `TeX` document that loops forever, for example, the simple line

```
A \let\par\relax \vskip
```

will start a horizontal list at `A`, redefines `\par`, then sees `\vskip` and inserts `\par` to end the paragraph. But this now only runs `\relax` so nothing changes and `\vskip` is read again, issues a `\par` which In short, it takes a plain `TeX` document with five tokens to run forever (as not even memory is consumed and therefore eventually exhausted).

There are no other ways than changing `\par` to gain control at the end of a paragraph, i.e., there is no token list like `\everypar` that is inserted, i.e., the only way to change the default behavior is to modify the action that `\par` executes with similar issues as outlined before: different processes need to ensure that they do not overwrite their modifications or worse, think that the `\par` in front of them is the engine primitive while in fact it has already been changed by other code.

To make matters slightly worse there are a few places where `TeX` handles the situation differently (most likely for speed reasons back when computers were much slower). If `TeX` finds itself in unrestricted horizontal mode at the end of building a vertical box (or an `\insert`, `\adjust` or at the end of executing the output routine code), it will finish the horizontal list not by issuing a `\par` command (which would be consistent with all other places, but by simply executing the primitive version of `\par` regardless of the definition that `\par` has at the time.

Thus, if you have carefully crafted a redefined `\par` to execute some special actions at the end of a paragraph and you write something like

```
\vbox{Some paragraph ... text.}
```

you will find that your code has never run for the last paragraph in that box. `LaTeX` avoids this problem, by making sure that all its boxes (such as `\parbox` or the `minipage` environment, etc.) all internally add an explicit `\par` at the end so that such code is run and `TeX` finds itself in vertical mode already without the need to start up the paragraph builder internally. But, of course, this only works for boxes under direct control of the

L^AT_EX kernel, if some package uses low-level `\vboxes` without adding this precaution the T_EX optimization kicks in and no special `\par` code is executed.

And there is another optimization that is painful: if a paragraph is interrupted by a mathematical display, e.g., `\[...\]` in L^AT_EX or `$$...$$` in plain T_EX, then T_EX will resume horizontal mode afterward, i.e., build a new horizontal list (without inserting an indentation box or `\everypar` at that point). However, if that list immediately ends with an explicit or implicit `\par` then T_EX will simply throw away this “null” paragraph and not do its usual “end of horizontal list” processing, so this special case need to be accounted for when introducing some extended processing.

2 The new mechanism implemented for L^AT_EX

To improve the situation (and also to support automatic tagging of PDF documents) we now offer public as well as private hooks at the start and end of the paragraph processing. The public hooks can be used by packages (or by the user in the preamble or within the document) and using the hook mechanisms it is possible to reorder or arrange code from different packages in a way that it can safely coexist.

To make that happen we have to make use of the basic functionality that is offered by T_EX, e.g., we install special code inside `\everypar` to provide hooks at the beginning and we redefine `\par` to do some special processing when appropriate to install hooks at the end of the paragraph.

In order to make this work, we have to ensure that package use of `\everypar` is not overwriting our code. This is done through a trick: we basically hide the real `\everypar` from the packages and offer them a new token register (with the same name). So if they install their own code it doesn’t overwrite ours. Our code then inserts the new `\everypar` at the right place inside the process so that it looks as if it was the primitive `\everypar`.¹⁶

At the end of the paragraph it would be great if we could use a similar trick. However, due to the fact that T_EX inserts the token `\par` (that doesn’t have a defined meaning) we can’t hide “the real thingTM” and offer the package an indistinguishable alternate.

Fortunately, L^AT_EX has already redefined `\par` for its own purposes. As a result there aren’t many packages that attempt to change `\par`, because without a lot of extra care that would fail miserably. But bottom line, if you load a package that alters `\par` then the end of paragraph hooks are most likely not executing while that redefinition is active.¹⁷

¹⁶Ideally, `\everypar` wouldn’t be used at all by packages and instead they would simply write their code into the hooks now offered by the kernel. However, while this is the longterm goal and clearly an improvement (because then the packages do no longer need to worry about getting their code overwritten or needing to account for already existing code in `\everypar`), this will not happen overnight. For that reason support for this legacy method is retained.

¹⁷Similarly to the `\everypar` situation, the remedy is that such packages stop doing this and instead add their alterations into the paragraph hooks now provided.

2.1 The provided hooks

`para/before`
`para/begin`
`para/end`
`para/after`

The following four public hooks are defined and executed for each paragraph:

para/before This hook is executed after the kernel hook `\@kernel@before@para@before` (discussed below) in vertical mode immediately after \TeX has contributed `\parskip` to the vertical list and before the actual paragraph processing in horizontal mode starts.

This hook should either not produce any typeset material or add only vertical material. If it starts a paragraph an error is generated. The reason is that we are in the starting process of processing a paragraph and so this would lead to endless recursion.¹⁸

para/begin This hook is executed after the kernel hook `\@kernel@before@para@begin` (discussed below) in horizontal mode immediately before the indentation box is placed (if there is any, i.e., if the paragraph hasn't been started with `\noindent`).

The indentation box to be typeset is available to the hook as `\IndentBox` and its automatic placement (after the hook is executed) can be prevented through `\OmitIndent`. More precisely `\OmitIndent` voids the box.

The indentation box is then typeset directly after the hook execution by something equivalent to `\box\IndentBox` followed by the current content of the token register `\everypar` that it is available to the kernel or to packages (that run some legacy code).

One has to be careful not to add any code to the hook that starts its own paragraph (e.g., by adding a `\parbox` or a `\marginpar` inside) because that would call the hook inside again (as a new paragraph is started there) and thus lead to an endless recursion ending only after exhausting the available memory. This can only be done by making sure that is not executed for the inner paragraphs (or at least not recursively forever).

para/end This hook is executed at the end of a paragraph when \TeX is ready to return to vertical mode and after it has removed the last horizontal glue (but not kern) placed on the horizontal list. The code is still executed in horizontal mode so it is possible to add further horizontal material at this point, but it should not alter the mode (even a temporary exit from horizontal mode would create chaos—any attempt will cause an error message)! After the hook has ended the kernel hook `\@kernel@after@para@end` is executed and then \TeX returns to vertical mode.

The hook is offered as public hook, but because of the requirement to stay within horizontal mode one needs to be careful in what is placed into the hook.¹⁹

This hook is implemented as a reversed hook.

para/after This hook is executed directly after \TeX has returned to vertical mode and after any material that migrated out of the horizontal list (e.g., from a `\vadjust`) has processed.

¹⁹Maybe we should guard against that, but it would be rather tricky to implement as mode changes can happen across group boundaries so one would need to keep a private stack just for that. Well, something to ponder.

This hook should either not produce any typeset material or add only vertical material. However, for this hook starting a new paragraph is not a disaster so that it isn't prevented.

This hook is implemented as a reversed hook.

Once that hook code has been processed the kernel hook `\@kernel@after@para@after` is executed as the final action of the paragraph processing.

```
\@kernel@before@para@before
\@kernel@after@para@after
\@kernel@before@para@begin
\@kernel@after@para@end
```

As already mentioned above there are also four kernel hooks that are executed at the start and end of the processing.

`\@kernel@before@para@before` For future extensions, not currently used by the kernel.

`\@kernel@after@para@after` For future extensions, not currently used by the kernel.

`\@kernel@before@para@begin` Used by the kernel to implement tagging. This hook is executed at the very beginning of a paragraph after `TEX` has switched to horizontal mode but before any indentation box got added or any `\everypar` was run.

It should not generate typeset material that could alter the position. Note that it should never leave hmode, otherwise you will end with a loop! We could guard against this, but since it is an internal kernel hook that shouldn't be touched this isn't checked.

`\@kernel@after@para@end` Used by the kernel to implement tagging. It is executed directly after the public `para/end` hook. After it there is a quick check that we are still in horizontal mode, i.e., that the public hook has not mistakenly ended horizontal mode prematurely (this is an incomplete check just testing the mode and could perhaps be improved (at the cost of speed)).

2.2 Altered and newly provided commands

```
\par
\endgraf
\para_end:
```

An explicit request for ending a paragraph is known in plain `TEX` under the name `\endgraf` where it simply calls the paragraph primitive (regardless of what `\par` may have as its current definition). In `LATEX` `\endgraf` with that behavior was also made available.

With the new paragraph handling in `LATEX`, ending a paragraph means a bit more than just calling the engine's paragraph builder: the process also has to add any hook code for the end of a paragraph. Thus `\endgraf` was changed to provide this additional functionality (and so by extension `\par` subject to its current meaning).

The `expl3` name for the functionality is `\para_end:`.

Note: *The next two commands are still under discussion and may slightly change their semantics (as described in the document) and/or their names between now and the 2021 Spring release!*

<hr/> <code>\OmitIndent</code> <code>\para_omit_indent:</code> <hr/>	<p>Inside the <code>para/begin</code> hook one can use this command to suppress the indentation box at the start of the paragraph. (Technically it is possible to use this command outside the hook as well, but this should not be relied upon.) The box itself remains available for use.</p> <p>The expl3 name for the function is <code>\para_omit_indent:</code>.</p>
<hr/> <code>\IndentBox</code> <code>\g_para_indent_box</code> <hr/>	<p>The box register holding the indentation box for the paragraph is available for inspection (or changes) inside hooks. It remains available even if the <code>\OmitIndent</code> command was used; in that case it will just not be automatically placed.</p> <p>The expl3 name for the box register is <code>\g_para_indent_box</code>.</p>
<hr/> <code>\RawIndent</code> <code>\para_raw_indent:</code> <code>\RawNoindent</code> <code>\para_raw_noindent:</code> <code>\RawParEnd</code> <code>\para_raw_end:</code> <hr/>	<p><code>\RawIndent</code> <i>hmode material</i> <code>\RawParEnd</code> <code>\RawNoindent</code> <i>hmode material</i> <code>\RawParEnd</code></p> <p>The commands <code>\RawIndent</code> and <code>\RawNoindent</code> are not meant for normal paragraph building (where the result is a textual paragraph in the the traditional meaning of the word), but for special cases where TeX’s low-level algorithm is used to achieve special effects, but where the result is not a “paragraph”.</p> <p>They are called “raw”, because they bypass L^AT_EX’s hook mechanism for paragraphs and simply invoke the low-level TeX algorithm. I.e., they are like the original TeX primitives <code>\indent</code> and <code>\noindent</code> (that is they execute no hooks other than <code>\everypar</code>) except that they can only be used in vertical mode and generate an error if found elsewhere.</p> <p>To avoid issues a paragraph started by them should always be ended by <code>\RawParEnd</code>²⁰ and not by <code>\par</code> (or a blank line), because the latter will execute hooks which then have no counterpart at the beginning of the paragraph. It is the responsibility of the programmer to make sure that they are properly paired. This also means that one should not put arbitrary user content between these commands if that content could contain stray <code>\pars</code>.</p> <p>The expl3 names for the functions are <code>\para_raw_indent:</code>, <code>\para_raw_indent:</code> and <code>\para_raw_end:</code>.</p>

2.3 Examples

None of the examples in this section are meant for real use as they are far too simple-minded but they should give some ideas of what could be possible if a bit more care is applied.

2.3.1 Testing the mechanism

The idea is to output for each paragraph encountered some information: a paragraph sequence number, a level number in roman numerals, the environment in which this paragraph appears, and the line number where the start or end of the paragraph is, e.g., something like

```

PARA: 1-i start (document env. on input line 38)
PARA: 1-i end   (document env. on input line 38)
PARA: 2-i start (document env. on input line 40)
PARA: 3-ii start (minipage env. on input line 40)
```

```

    PARA: 3-ii end    (minipage env. on input line 40)
    PARA: 2-i end     (document env. on input line 41)

```

As you can see paragraph 2 starts on line 40 and ends on 41 and inside a minipage started paragraph 3 (start and end on line 40). If you run this on some document you will find that L^AT_EX considers more things “a paragraph” than you have probably thought.

This was generated by the following hook code:

```

\newcounter{paracnt}          % sequence counter
\newcounter{paralevel}        % level counter

```

To support paragraph nesting we need to maintain a stack of the sequence numbers. This is most easily done using expl3 functions, so we switch over. This is not a very general implementation, just enough for what we need and a bit of L^AT_EX 2_ε thrown in as well. When popping the result gets stored in `\paracntvalue` and the `\ERROR` should never happen because it means we have tried to pop from an empty stack.

```

\ExplSyntaxOn
\seq_new:N \g_para_seq
\cs_new:Npn \ParaPush
  {\seq_gpush:No \g_para_seq {\the\value{paracnt}}}
\cs_new:Npn \ParaPop  {\seq_gpop:NNF \g_para_seq \paracntvalue \ERROR }
\ExplSyntaxOff

```

At the start of the paragraph increment both sequence counter and level and also save the then current sequence number on our stack.

```

\AddToHook{para/begin}{%
  \stepcounter{paracnt}\stepcounter{paralevel}%
  \ParaPush
}

```

To display the sequence number we `\typeout` the current sequence and level number. The command `\@currenenvir` gives us the current environment and `\on@line` produces a space and the current input line number.

```

\typeout{PARA: \arabic{paracnt}-\roman{paralevel} start
  (\@currenenvir\space env.\on@line)}%

```

We also typeset the sequence number as a tiny red number in a box that takes up no horizontal space. This helps us seeing where L^AT_EX sees the start and end of the paragraphs in the document.

```

\llap{\color{red}\tiny\arabic{paracnt}\ }%
}

```

At the end of the paragraph we display sequence number and level again. The level counter has the correct value but we need to retrieve the right sequence value by popping it off the stack after which it is available in `\paracntvalue` the way we have set this up above.

```

\AddToHook{para/end}{%
  \ParaPop
  \typeout{PARA: \paracntvalue-\roman{paralevel} end \space\space
    (\@currenenvir\space env.\on@line)}%
}

```

We also typeset again a tiny red number with that value, this time sticking out to the right.²¹ We also decrement the level counter since our level has finished.

```
\rlap{\color{red}\tiny\ \paracntvalue}%
\addtocounter{paralevel}{-1}%
}
\makeatother
```

2.3.2 Mark the first paragraph of each `itemize`

The code for this is rather simple. We apply hook code that is executed only once inside a hook that is executed at the begin of each `itemize`. We explicitly change the color back and forth so that we don't introduce grouping around the paragraph.

```
\AddToHook{env/itemize/begin}{%
\AddToHookNext{para/begin}{\color{blue}}%
\AddToHookNext{para/end}{\color{black}}%
}
```

As a result the first paragraph of each `itemize` will appear in blue.

2.4 Some technical notes

The code tries hard to be transparent for package code, but of course any change means that there is a potential for breaking other code. So in section we collect a few cases that may be of importance if low-level code is dealing with paragraphs that are now behaving slightly differently. The notes are from issues we observed and will probably grow over time.

2.4.1 Glue items between paragraphs (found with `fancypar`)

In the past \LaTeX placed two glue items between two consecutive paragraph, e.g.,

```
text1 \par text2 \par
```

would show something like

```
\glue(\parskip) 0.0 plus 1.0
\glue(\baselineskip) 5.16669
```

but now there is anothe `\parskip` glue (that is always 0pt):

```
\glue(\parskip) 0.0 plus 1.0
\glue(\parskip) 0.0
\glue(\baselineskip) 5.16669
```

The reason is that we generate a “fake” paragraph to gain control and safely add the early hooks, but this generates an additional glue item. That item doesn't contribute anything vertically but if somebody writes code the unravels a constructed list using `\lastbox`, `\unskip` and `\unpenalty` then the code has to remove one additional glue item or else will fail.

²¹Note that this can alter the document pagination, because a paragraph ending in a display (e.g., an equation) will get an extra line—in that case our tiny number has an effect even though it doesn't take up any space, because it paragraph is no longer empty and thus isn't dropped!

3 The Implementation

```

1 <@@=para>
2 <{*2ekernel | latexrelease>
3 \ExplSyntaxOn
4 <latexrelease> \NewModuleRelease{2021/06/01}{ltpara}
5 <latexrelease> {Paragraph~handling~and~hooks}

```

3.1 Providing hooks for paragraphs

para/before The public hooks. They are implemented as a paired set of hooks.

```

para/after 6 \hook_new_pair:nn{para/before}{para/after}
para/begin 7 \hook_new_pair:nn{para/begin}{para/end}
para/end

```

(End definition for `para/before` and others. These functions are documented on page 267.)

\@kernel@before@para@before The corresponding kernel hooks (for tagging and future extensions).

```

\@kernel@after@para@after 8 \let \@kernel@before@para@before \@empty
\@kernel@before@para@begin 9 \let \@kernel@before@para@begin \@empty
\@kernel@after@para@end 10 \let \@kernel@after@para@end \@empty
11 \let \@kernel@after@para@after \@empty

```

(End definition for `\@kernel@before@para@before` and others. These functions are documented on page 268.)

`\g__para_standard_everypar_tl`

Whenever \TeX starts a paragraph it inserts first an indentation box and then executes the tokens stored in `\tex_everypar:D` (known to \LaTeX as `\everypar`). We alter this behavior slightly here, so that hooks are added into the right place. Otherwise the process change remains transparent to any legacy code for this space.

We keep the standard code to be used by `\tex_everypar:D` in a separate token list because we have to switch back and forth for error recovery and so altering `\tex_everypar:D` all the time should be a tiny bit faster.

```
12 \tl_new:N \g__para_standard_everypar_tl
```

Here is now its definition:

```
13 \tl_gset:Nn \g__para_standard_everypar_tl {
```

First we remove the indentation box and store it in `\g_para_indent_box`. If there was none because the paragraph was started by `\noindent` the box register will be void.

```
14 \box_gset_to_last:N \g_para_indent_box
```

This will make the newly started horizontal list empty, so if we stop it now and return to vertical mode it will be dropped by \TeX . We do that but inside a group so that any `\parshape` settings will not get lost as we need them for later.

```

15 \group_begin:
16 \tex_par:D
17 \group_end:

```

We then change `\tex_everypar:D` to generate an error so that we can detect and report if the `para/before` hook illegally changed out of vmode.

```

18 \tex_everypar:D { \msg_error:nnnn { hooks }{ para-mode }{before}{vertical} }
19 \@kernel@before@para@before
20 \hook_use:n {para/before}

```

Assuming the hooks have been well behaved it is time to return to horizontal mode and start the paragraph in earnest. We already have the indentation box saved away so we now have to restart the paragraph with an empty `\tex_everypar:D` and with `\tex_noindent:D`. And we need to make sure not to get another `\parskip` or rather (since we can't prevent that) that it is of zero size.

```

21 \group_begin:
22   \tex_everypar:D {}
23   \skip_zero:N \tex_parskip:D
24   \tex_noindent:D
25 \group_end:

```

That brings us back to the start of the horizontal list but we need to change `\tex_everypar:D` back to its normal content in case there are nested paragraphs coming up.

```

26 \tex_everypar:D{\g__para_standard_everypar_tl}

```

This is followed by executing the kernel and the public hook. The kernel hook is there to enable tagging.

```

27 \@kernel@before@para@begin
28 \hook_use:n {para/begin}

```

If we aren't in horizontal mode any longer the hooks above misbehaved.

```

29 \if_mode_horizontal: \else:
30   \msg_error:nnnn { hooks }{ para-mode }{begin}{vertical} \fi:

```

Finally we reinsert the indentation box (unless suppressed) and then call `\everypar` the way legacy L^AT_EX code expects it.

However, adding the public `\everypar` is a bit tricky (see below) so we add that later, and indirectly.

```

31 \__para_handle_indent:
32 % \the \everypar          % <--- done differently below
33 }

```

(End definition for `\g__para_standard_everypar_tl`.)

`\tex_everypar:D` `\tex_everypar:D` then only has to execute `\g__para_standard_everypar_tl` by default.

```

34 \tex_everypar:D{\g__para_standard_everypar_tl}

```

(End definition for `\tex_everypar:D`.)

`\everypar` Tokens inserted at the beginning of the paragraph are placed into `\everypar` inside legacy L^AT_EX code, e.g., by the list environments or by headings to handle `\clubpenalty`, etc. Now this isn't any longer the primitive but simply a toks register used in the code above but to legacy L^AT_EX code that is transparent.

There is, however, a problem: a handful packages use exactly the same trick and replace the primitive with a token register and call the token register inside the renamed primitive. That is they assume that `\everypar` is the primitive and that it will still be called at the start of the paragraph even if renamed.

But if we have already replaced it by a token register then all they do is to give that token register a new name. Thus our code in `\tex_everypar:D` would call `\everypar` (which is their now token register) and the code that they added ends up in our token register which is then never used at all. A bit mind boggling I guess.

So what we have to do is not to call the token register `\everypar` by its name inside `\tex_everypar:D` but by using its actual register number.

```

35 \newtoks \everypar

```


After we have allocated a new toks register with the name `\everypar` the actual register number is available (briefly) inside `\allocationnumber`. So instead of `\the\everypar` we have to put `\the\toks<allocated number>` at the end of `\tex_everypar:D`.

So what remains doing is to append a few tokens to the token list `\g__para_standard_everypar_tl` which we do now. We use x expansion here to get the value of `\allocationnumber` in, all the other tokens should not be expanded at this point.

One important point here is to terminate the register allocation number with a real space. This space will get swallowed up when the number is read. Anything else, such as `\scan_stop:` would remain in the input and that would mean that it would interfere with `\everypar` code that attempts to scan ahead to see how the paragraph text starts.

```

36 \tl_gput_right:Nx \g__para_standard_everypar_tl {
37   \exp_not:N \the
38   \exp_not:N \toks
39   \the \allocationnumber
40   \c_space_tl
41 }

```

(End definition for `\everypar`.)

`\g_para_indent_box` For managing the indentation we need to provide a public accessible box register

```

42 \box_new:N \g_para_indent_box

```

(End definition for `\g_para_indent_box`. This function is documented on page 269.)

`__para_handle_indent:` Adding (typesetting) the indent box is straight forward. If it was emptied before it does nothing.

```

43 \cs_new:Npn \__para_handle_indent: {
44   \box_use_drop:N \g_para_indent_box
45 }

```

The declaration `\para_omit_indent:` (or `\OmitIndent`) changes that to do nothing.

```

46 \cs_new:Npn \para_omit_indent: {
47   \box_gclear:N \g_para_indent_box
48 }

```

(End definition for `__para_handle_indent:`.)

`\IndentBox` The L^AT_EX 2_ε names for the indentation box and for suppressing it for use in the `para/begin` hook.

```

49 \cs_set_eq:NN \IndentBox \g_para_indent_box
50 \cs_set_eq:NN \OmitIndent \para_omit_indent:

```

(End definition for `\IndentBox` and `\OmitIndent`. These functions are documented on page 269.)

`\para_end:` Adding hooks to the end of a paragraph is similar but here we need to alter the command that is used by T_EX to end horizontal mode and return to vertical mode, i.e., `\par`.

This is a bit more complicated as this command can appear anywhere either explicitly or implicitly added by T_EX in certain situations:

- when using `\par` in the code or the document
- when using a blank line (which is converted to `\par`)
- when T_EX finds any commands incompatible with horizontal mode it issues a `\par` and then rereads the command.

Unfortunately, T_EX has some (these days) unnecessary optimization: if a `\vbox` ends and T_EX is still in horizontal mode it simply exercises the paragraph builder instead of issuing a `\par`. It is therefore necessary for L^AT_EX to ensure that this case doesn't happen and all boxes internally have a `\par` command at their end.

This `\par` may or may not run the “par primitive” (which is always available as `\tex_par:D` in `expl3`); it is permissible to have a changed meaning and it is in fact changed by L^AT_EX in various ways at various points inside `latex.ltx`. For this L^AT_EX 2_ε code has the following conventions: `\@@par` and `\endgraf` both refer to the default meaning (in the past this was the initex primitive) while `\par` is the current meaning which may do something else.

We are now going to change this default meaning to run `\para_end:` instead, which ultimately executes the initex primitive but additionally adds our hooks when appropriate. This way the change is again transparent to the legacy L^AT_EX 2_ε code.

In most cases `\para_end:` should behave exactly like the primitive and we achieve this by simply expanding it to the primitive which is available to us as `\tex_par:D`. This way we don't have to care about whether T_EX just does nothing (e.g., if in vertical mode already) or generate an error, etc.

```
51 \cs_new_protected:Npn \para_end: {
```

The only case we care about is when we are in horizontal mode (i.e., doing typesetting) and not also in inner mode (i.e., making paragraphs and not building an `\hbox`).

```
52 % \bool_lazy_and:nnT
53 % { \mode_if_horizontal_p: }
54 % { \bool_not_p:n { \mode_if_inner_p: } }
55 % { ...
```

Since this is executed for each and every paragraph in a document we try to stay as fast as possible. So we are aren't using the above construct but two conditionals instead. Using low-level `\if_mode...` conditions would be even faster but has the danger to conflict with conditionals in the user hooks.

```
56 \mode_if_horizontal:TF {
57 \mode_if_inner:F {
```

In that case the action of the primitive would be to remove the last glue (not kern) from the horizontal list constructed to form a paragraph then append the a penalty of 10000 and the `\parfillskip` at the end and pass the whole list to the paragraph builder which breaks it into lines and T_EX then returns to vertical mode.

What we want to do instead is to add our hook code at the end of the horizontal list before that happens and the code is passed to the paragraph builder. If there was a glue item at the end then it should get removed before the hook code gets added so we have to arrange for its removal ourselves.

There is not much point in checking if there was really a glue item at the end of the horizontal list, instead we simply try to remove one using `\tex_unskip:D`, if there wasn't one this will do nothing.

```
58 \tex_unskip:D
```

The we execute the public hook (which may add final typesetting material) followed by the kernel hook we need for adding tagging support. None of this is supposed to change the mode—at the moment we make only a very simple test for this, more devious changes go unnoticed, but too bad, that will then probably badly backfire.

```
59 \hook_use:n{para/end}
60 \@kernel@after@para@end
61 \mode_if_horizontal:TF {
```

The final action (before getting to the point where `\tex_par:D` is called) is to add a kern item so that the primitive is prevented from removing glue (if there was some). If we don't do this and the horizontal list ended in several glue items we would end up with removing two instead of just the last one, which would be wrong. We use a kern as that is minimally faster.

There is however one other T_EX optimization that hurts: in a sequence like this `$$... $$ \par` T_EX will be in horizontal mode after the display, ready to receive further paragraph text, but since the `\par` follows immediately there is a “null” paragraph at the end and T_EX simply throws that away. The space between `$$` and `\par` got already dropped during the display processing so the `\par` is not removing any space and appending `\parfillskip`, instead it simply goes silently to vmode. Now if we would had added something (to prevent glue removal) that would look to T_EX like material after the display and so we would end up with an empty paragraph just containing `\parfillskip`.

We therefore check if the current hlist is empty (`\tex_lastnodetype:D` has the value -1 and only if not we add our kern.

```

62         \if_int_compare:w 0 < \tex_lastnodetype:D
63         \tex_kern:D \c_zero_dim
64         \fi:

```

To run the `para/after` hook we first end the paragraph. This means that the `\tex_par:D` at the very end is unnecessary but executing it there unnecessarily is better than having code that test for all the different mode possibilities.

```

65         \tex_par:D
66         \hook_use:n{para/after}
67         \@kernel@after@para@after
68     }

```

If we haven't been in horizontal mode then the earlier hook `para/end` is at fault and we report that.

```

69         { \msg_error:nnnn { hooks }{ para-mode }{end}{horizontal} }

```

Finally close out the nested conditionals.

```

70     }
71 }

72 \tex_par:D
73 }

```

(End definition for `\para_end:`. This function is documented on page 268.)

`\para_raw_indent:` The commands `\para_raw_indent:` and `\para_raw_noindent:` are like the primitives `\indent` and `\noindent` except that they can only be used in vertical mode.

`\para_raw_noindent:` To avoid issues a paragraph started by them should always be ended by `\para_raw_end:` and not by `\para_end:` or `\par` as the latter will execute hooks which then have no counterpart at the beginning of the paragraph. It is the responsibility of the programmer to make sure that they are properly paired.

`\para_raw_end:`

```

74 \cs_new:Npn \para_raw_indent: {
75     \mode_if_vertical:TF
76     {
77         \tex_everypar:D {
78             \box_gset_to_last:N \g_para_indent_box
79             \tex_everypar:D { \g__para_standard_everypar_tl }
80             \__para_handle_indent:
81             \the\everypar }

```

```

82     }
83     { \msg_error:nn { kernel }{ raw-para } }
84     \tex_indent:D
85 }

86 \cs_new:Npn \para_raw_noindent: {
87     \mode_if_vertical:TF
88     {
89         \tex_everypar:D {
90             \tex_everypar:D { \g__para_standard_everypar_tl }
91             \the\everypar }
92     }
93     { \msg_error:nn { kernel }{ raw-para } }
94     \tex_noindent:D
95 }

96 \cs_new_eq:NN \para_raw_end: \tex_par:D

```

(End definition for `\para_raw_indent:`, `\para_raw_noindent:`, and `\para_raw_end:`. These functions are documented on page 269.)

\RawIndent The L^AT_EX 2_ε names for starting and ending a paragraph without adding any hooks.
\RawNoIndent 97 \cs_set_eq:NN \RawIndent \para_raw_indent:
\RawParEnd 98 \cs_set_eq:NN \RawNoindent \para_raw_noindent:
99 \cs_set_eq:NN \RawParEnd \para_raw_end:

(End definition for `\RawIndent`, `\RawNoIndent`, and `\RawParEnd`. These functions are documented on page 269.)

This ends the `para` module code.

```
100 <@@=>
```

\par Having the new default definition for `\par` we also have to set it up so that it gets used.
\endgraf This is needed in three places `\par`, `\@@par` (to which L^AT_EX resets `\par` occasionally)
\@@par and `\endgraf` which is another name for the “default” action of `\par`.

```

101 \cs_set_eq:NN \par \para_end:
102 \cs_set_eq:NN \@@par \para_end:
103 \cs_set_eq:NN \endgraf \para_end:

```

(End definition for `\par`, `\endgraf`, and `\@@par`. These functions are documented on page 268.)

While this is not integrated properly into the format we have to redo the `\everypar` setting from the kernel, otherwise that gets lost (as it happens before that file is loaded).

```
104 \everypar{\@nodocument} %% To get an error if text appears before the
```

3.2 The error messages

This one is used when we detect that some hook code has changed the mode where it shouldn't, e.g., by starting or ending a paragraph. The first argument is the hook name second the mode it should have stayed in but didn't.

```

105 \msg_new:nnnn { hooks } { para-mode }
106 {
107     Illegal-mode~ change~ in~ hook~ 'para/#1'.\\
108     Hook~ code~ did~ not~ remain~ in~ #2~ mode.
109 }
110 {

```

```

111 Paragraph~ hooks~ cannot~ change~ the~ TeX~ mode~ without~ causing~
112 endless~ recursion.~ The~ hook~ code~ in~ 'para/#1'~ needs~ to~ stay~
113 in~ #2~ mode,~ but~ it~ didn't.~ Examine~ the~ hook~
114 code~ with~ \iow_char:N \ShowHook~ to~ find~ the~ issue.
115 }

```

And here is one used in the “raw” commands when they are used outside of vertical mode.

```

116 \msg_new:nnnn { kernel } { raw-para }
117 {
118   Not~ in~ vertical~ mode.
119 }
120 {
121   Starting~ a~ paragraph~ with~ \iow_char:N \RawIndent~ or~
122   \iow_char:N \RawNoindent \
123   (or~ \iow_char:N \para_raw_indent:~ or~
124   \iow_char:N \para_raw_noindent:~ is~ only~ allowed \
125   if~ LaTeX~ is~ in~ vertical~ mode.
126 }

127 %
128 <latexrelease>\IncludeInRelease{0000/00/00}%
129 <latexrelease>                                {ltpara}{Undo~hooks~for~paragraphs}
130 <latexrelease>
131 <latexrelease>\let \OmitIndent \undefined
132 <latexrelease>\let \IndentBox \undefined
133 <latexrelease>\let \RawIndent \undefined
134 <latexrelease>\let \RawNoindent \undefined
135 <latexrelease>\let \RawParEnd \undefined
136 <latexrelease>
137 <latexrelease>\cs_set_eq:NN \par \tex_par:D
138 <latexrelease>\cs_set_eq:NN \@par \tex_par:D
139 <latexrelease>\cs_set_eq:NN \endgraf \tex_par:D
140 <latexrelease>
141 <latexrelease>\EndModuleRelease
142 \ExplSyntaxOff
143 </2ekernel | latexrelease>

```

File o

ltspace.dtx

1 Spacing

This section deals with spacing, and line- and page-breaking.

1.1 User Commands

`\nolinebreak` [$\langle i \rangle$] : $\langle i \rangle = 0, \dots, 4$.
 Default argument = 4. Puts a penalty into the vertical list output as follows:
 0 : penalty = 0
 1 : penalty = `\@lowpenalty`
 2 : penalty = `\@medpenalty`
 3 : penalty = `\@highpenalty`
 4 : penalty = 10000
`\pagebreak` [$\langle i \rangle$] : same as except negatives of its penalty
`\linebreak` [$\langle i \rangle$] : analog of the above
`\nolinebreak` [$\langle i \rangle$] : analog of the above
`\samepage` : inhibits page breaking most places by setting the following penalties to 10000:
 `\interlinepenalty`
 `\postdisplaypenalty`
 `\interdisplaylinepenalty`
 `\@beginparpenalty`
 `\@endparpenalty`
 `\@itempenalty`
 `\@secpenalty`
 `\interfootnotelinepenalty`
`\` : initially defined to be `\newline`
`\` [$\langle length \rangle$] : initially defined to be `\vspace{\langle length \rangle}\newline`
 Note: `\`* adds a `\vadjust{\penalty 10000}`
 OBSOLETE COMMANDS (which never made it into the manual):
 `\obeycr` : defines `<CR> == \`
 `\restorecr` : restores `<CR>` to its usual meaning.

1.2 Chris' comments

There are several aspects of the handling of space in horizontal mode that are inconsistent or do not work well in some cases. These are largely concerned with ignoring the effect of space tokens that would otherwise typeset an inter-word space.

Negating the effect of such space tokens is achieved by two mechanisms:

- `\unskip` is used to remove the glue just added by a space that has already had its effect; it is sometimes invoked after an `\ifdim` test on `\lastskip` (see below);
- `\ignorespaces` is used to ignore space-tokens yet to come.

The test done on `\lastskip` is sometimes for equality with zero and sometimes for being positive. Recall also that the test is only on the natural length of the glue and that no glue cannot be distinguished from glue whose natural length is zero: to summarise, a pretty awful test. It is not clear why these tests are not all the same; I think that they should all be for equality. One place where `\unskip` is often used is just before a `\par` (which itself internally does an `\unskip`) and one bit of code (in `\@item`) even has two `\unskips` before a `\par`. These uses may be fossil code but if they are necessary, maybe `\@killglue` would be even safer.

Such removal of glue by `\unskip` may sometimes have the wrong result, removing not the glue from a space-token but other explicit glue; this is sometimes not what is intended.

A common way to prevent such removal is to add an `\hskip\z@` after the glue that should not be removed. This protects that glue against one `\unskip` with no test but not against more than one. It does work for ‘tested `\unskips`’. This is used by `\hspace*` but not by `\hspace`; this is inconsistent as the star is supposed to prevent removal only at the beginning of a line, not at the end, or in a tabular, etc.

If this reason for removing glue were the only consideration then a tested-`\unskip` and protection by `\hskip\z@` would suffice but would need to be consistently implemented.

However, the class of invisibles, commands and environments tries to be even cleverer: one of these tries to leave only one inter-word space whenever there is one before it and one after it; and it does this quite well.

But problems can arise when there is not a space-token on both sides of it; in particular, when an invisible appears at the beginning or end of a piece of text the method still leaves one space token whereas usually in these cases it should leave none.

Also, the current rules do not work well when more than one such command appears consecutively, separated by space-tokens; it leaves glue between every other invisible.

There is also a question about what these commands should do when they occur next to spaces that do not come from space tokens but, for example, from `\hspace`. Should they still produce ‘just one space’? If so, which one? It is good to note that the manual is sufficiently cautious about invisibles that we are not obliged to make anything work.

Another interesting side-road to explore is whether the space-tokens either side of an `\hspace{...}` should be ignored.

One alternative to the current algorithm that is often suggested is that all glue around the invisible should be consolidated into a space after it (usually without stating how much glue should be put there). The command `\nolinebreak` is implemented this way (and `\linebreak` should also be). This does not work correctly for the following common case:

```
... some text
\index{some-word}
some-word and more text.
```

This is optimal coding since it is normal to index a word that gets split across a page-break on its starting page. This would, on the other hand, fix another common (and documented) failure of the current system: when the invisible is the last thing in a paragraph the space before it is not removed and, worse, it is also hidden from the paragraph-ending mechanism so that an ‘empty’ line can be created at the end of the paragraph.

Another deficiency (I think) of the current system is that the following is treated as having the `\index` command between the paragraphs, which is probably not what the author intended (since there is no empty line after it).

```
\index{beginnings}
Beginnings of paragraphs ...
```

I know of no algorithm that will handle satisfactorily even all the most common cases; note that it could be that the best algorithm may be different for different invisibles since, for example, the common uses and expected behaviour of `\index`, `\marginpar`, `\linebreak`, `\pagebreak` and `\vspace` are somewhat different. [For example, is `\vspace` ever used in the middle of a paragraph?]

One method that can (and is) used to make invisible commands produce no space when used at the beginning of text is to put in some glue that is nearly enough the same as no glue or glue of zero length in all respects except for the precise test for not being exactly equal to zero; examples of such glue are `\hskip 1sp` and, possibly better but more complex, `\hskip -1sp \hskip 1sp`. However, this only works when it is known that user-supplied text is about to start.

Some similar concerns apply to the handling of space and penalties in vertical mode; there is an extra hurdle here as `\unskip` does not work on the main vertical list. The complexity of the tests done by `\addvspace` have never been explained.

The implementation of space hacks etc for vertical mode is another major area that needs further attention; my earlier experiments did not produce much improvement over the current unsatisfactory situation.

One particular problem is what happens when the following very natural coding is used (part of the problem here is that this looks like an hmode problem, but it is not):

```
... end of text.

\begin{enumerate}
  \item \label{item:xxx} Item text.
\end{enumerate}
```

1.3 Some immediate actions

- Fix bug in `\linebreak`.
- Fix bug in `*`.
- Reimplement `\\`, etc, removing extra `\adjusts` and getting better error trapping (this seems to involve a lot more tokens).
- Investigate whether `\\`, etc need to be errors in vmode; I think that they could be noops (maybe with a warning).
- Make all(?) `\unskips` include test for zero skip (rather than other tests or no test).
- Consider replacing `\hskip 1sp` by something better (here called an ‘infinitesimal’ skip).
- Look at all `\hskip\z@` (or similar) to see if they should be changed to an ‘infinitesimal’ skip.

- Resolve the inconsistency between `\hspace` and `\hspace*`.
- Remove unnecessary `\unskips`.
- Investigate and rationalise the ‘newline’ code.
- Find better algorithms for all sorts of things or, easier(?), fix T_EX itself.

1.4 The code

```

1  <*2ekernel>
2  \message{spacing,}
3  </2ekernel>
4  <*2ekernel | latexrelease>
5  <latexrelease>\IncludeInRelease{2019/10/01}%
6  <latexrelease>                {\pagebreak}{Make commands robust}%

\pagebreak
\nopagebreak
7  \DeclareRobustCommand\pagebreak{\@testopt{\@no@pgbk-}4}
8  \DeclareRobustCommand\nopagebreak{\@testopt{\@no@pgbk4}

(End definition for \pagebreak and \nopagebreak.)

\linebreak
\nolinebreak
9  \DeclareRobustCommand\linebreak{\@testopt{\@no@lnbk-}4}
10 \DeclareRobustCommand\nolinebreak{\@testopt{\@no@lnbk4}

(End definition for \linebreak and \nolinebreak.)

\samepage
11 \DeclareRobustCommand\samepage{\interlinepenalty\@M
12   \postdisplaypenalty\@M
13   \interdisplaylinepenalty\@M
14   \@beginparpenalty\@M
15   \@endparpenalty\@M
16   \@itempenalty\@M
17   \@secpenalty\@M
18   \interfootnotelinepenalty\@M}

(End definition for \samepage.)

19 </2ekernel | latexrelease>
20 <latexrelease>\EndIncludeInRelease
21 <latexrelease>\IncludeInRelease{0000/00/00}%
22 <latexrelease>                {\pagebreak}{Make commands robust}%
23 <latexrelease>
24 <latexrelease>\kernel@make@fragile\pagebreak
25 <latexrelease>\kernel@make@fragile\nopagebreak
26 <latexrelease>\kernel@make@fragile\linebreak
27 <latexrelease>\kernel@make@fragile\nolinebreak
28 <latexrelease>\kernel@make@fragile\samepage
29 <latexrelease>
30 <latexrelease>\EndIncludeInRelease
31 <*2ekernel>

```

`\@no@pgbk`

```
32 \def\@no@pgbk #1[#2]{%
33   \ifvmode
34     \penalty #1\@getpen{#2}%
35   \else
36     \@bsphack
37     \vadjust{\penalty #1\@getpen{#2}}%
38     \@esphack
39   \fi}
```

(End definition for \@no@pgbk.)

`\@no@lnbk`

```
40 \def\@no@lnbk #1[#2]{%
41   \ifvmode
42     \@nolnerr
43   \else
44     \@tempskipa\lastskip
45     \unskip
46     \penalty #1\@getpen{#2}%
47     \ifdim\@tempskipa>\z@
48       \hskip\@tempskipa
49       \ignorespaces
50     \fi
51   \fi}
```

(End definition for \@no@lnbk.)

`\@` The purpose of the new code is to fix a few bugs; however, it also attempts to optimize the following, in order of priority:

1. efficient execution of plain `\@`;
2. efficient execution of `\@[...]`;
3. memory use;
4. name-space use.

The changes should make no difference to the typeset output. It appears to be safe to use `\reserved@e` and `\reserved@f` here (other reserved macros are somewhat disastrous).

These changes made `\newline` even less robust than it had been, so now it is explicitly robust, like `\@`.

The internal definition of the ‘normal’ definition of `\@`.

`\@normalcr`

```
52 </2ekernel>
53 <*2ekernel | latexrelease>
54 <latexrelease> \IncludeInRelease{2020/02/02}%
55 <latexrelease>           {\@normalcr}{Make robust}%
56 \protected\def\@normalcr{%
57   \let \reserved@e \relax
58   \let \reserved@f \relax
59   \@ifstar{\let \reserved@e \vadjust \let \reserved@f \nobreak
60             \@xnewline}%
61   \@xnewline}
```

```

62 \let\\@normalcr
63 </2ekernel | latexrelease>
64 <latexrelease>\EndIncludeInRelease
65 <latexrelease>\IncludeInRelease{0000/00/00}%
66 <latexrelease>          {\@normalcr}{Make robust}%
67 <latexrelease>
68 <latexrelease>\DeclareRobustCommand\\{%
69 <latexrelease>  \let \reserved@e \relax
70 <latexrelease>  \let \reserved@f \relax
71 <latexrelease>  \@ifstar{\let \reserved@e \vadjust \let \reserved@f \nobreak
72 <latexrelease>          \@xnewline}%
73 <latexrelease>  \@xnewline}
74 <latexrelease>\expandafter\let\expandafter\@normalcr
75 <latexrelease>  \csname\expandafter\@gobble\string\\ \endcsname
76 <latexrelease>
77 <latexrelease>\EndIncludeInRelease
78 <*2ekernel>

```

(End definition for \\ and \@normalcr.)

\@vspace@calcify Helper command to produce a \vskip that is first run through \setlength. This way the calc package can operate on the argument value.

```

79 </2ekernel>
80 <*2ekernel | latexrelease>
81 <latexrelease>\IncludeInRelease{2020/10/01}%
82 <latexrelease>          {\@vspace@calcify}{Add calc support}%
83 \def\@vspace@calcify#1{\begingroup\setlength\skip@{#1}\vskip\skip@\endgroup}
84 </2ekernel | latexrelease>
85 <latexrelease>\EndIncludeInRelease
86 <latexrelease>\IncludeInRelease{0000/00/00}%
87 <latexrelease>          {\@vspace@calcify}{Add calc support}%
88 <latexrelease>
89 <latexrelease>\let\@vspace@calcify\@undefined
90 <latexrelease>\EndIncludeInRelease
91 <*2ekernel>

```

(End definition for \@vspace@calcify.)

\newline A simple form of the ‘normal’ definition of \\.

```

92 \DeclareRobustCommand\newline{\@normalcr\relax}

```

(End definition for \newline.)

\@xnewline

```

93 \def\@xnewline{\@ifnextchar[% ] bracket matching
94          \@newline
95          {\@gnewline\relax}}

```

(End definition for \@xnewline.)

\@newline

```

96 </2ekernel>
97 <*2ekernel | latexrelease>
98 <latexrelease>\IncludeInRelease{2020/10/01}%

```

```

99 <latexrelease>                                {\@newline}{\newline calc support}%
100 \def\@newline[#1]{\let \reserved@e \vadjust
101                      \@gnewline {\@vspace@calcify{#1}}}
102 </2ekernel | latexrelease>
103 <latexrelease>\EndIncludeInRelease

104 <latexrelease>\IncludeInRelease{0000/00/00}%
105 <latexrelease>                                {\@newline}{\newline calc support}%
106 <latexrelease>
107 <latexrelease>\def\@newline[#1]{\let \reserved@e \vadjust
108 <latexrelease>                                \@gnewline {\vskip #1}}
109 <latexrelease>\EndIncludeInRelease
110 <*2ekernel>

```

(End definition for \@newline.)

\@gnewline The \nobreak added to prevent null lines when \\ ends an overfull line. Change made 24 May 89 as suggested by Frank Mittelbach and Rainer Schöpf

```

111 \def\@gnewline #1{%
112   \ifvmode
113     \@nolnerr
114   \else
115     \unskip \reserved@e {\reserved@f#1}\nobreak \hfil \break
116   \fi}

```

(End definition for \@gnewline.)

\@getpen

```

117 \def\@getpen#1{\ifcase #1 \z@ \or \@lowpenalty\or
118               \@medpenalty \or \@highpenalty
119               \else \@M \fi}

```

(End definition for \@getpen.)

\if@nobreak Switch used to avoid page breaks caused by \label after a section heading, etc. It should be **GLOBALLY** set true after the \nobreak and **globally** set false by the next invocation of \everypar.

Commands that reset \everypar should globally set it false if appropriate.

```

120 \def\@nobreakfalse{\global\let\if@nobreak\iffalse}
121 \def\@nobreaktrue {\global\let\if@nobreak\iftrue}
122 \@nobreakfalse

```

(End definition for \if@nobreak.)

\@savsk Registers used to save the space factor and last skip.

```

123 \newdimen\@savsk
124 \newcount\@savsf

```

(End definition for \@savsk and \@savsf.)

`\@bsphack` `\@bsphack` and `\@esphack` used by macros such as `\index` and `\begin{@float} ... \end{@float}` that want to be invisible — i.e., not leave any extra space when used in the middle of text. Such a macro should begin with `\@bsphack` and end with `\@esphack`. The macro in question should not create any text, nor change the mode.

Before giving the current definition we give an extended definition that is currently not used (because it doesn't work as advertised:-)

These are generalised hacks which attempt to do sensible things when 'invisible commands' appear in vmode too.

They need to cope with space in both hmode (plus spacefactor) and vmode, and also cope with breaks etc. In vmode this means ensuring that any following `\advspace`, etc sees the correct glue in `\lastskip`.

In fact, these improved versions should be used for other cases of 'whatsits, thingies etc' which should be invisible. They are only for commands, not environments (see notes on `\@Esphack`).

BTW, anyone know why the standard hacks are surrounded by `\ifmmode\else` rather than simply `\ifhmode`?

And are there any cases where saving the spacefactor is essential? I have some extensions where it is, but it does not appear to be so in the standard uses.

```
def \@bsphack{%
  \relax \ifvmode
    \@savsk \lastskip
    \ifdim \lastskip=\z@
    \else
      \vskip -\lastskip
    \fi
  \else
    \ifhmode
      \@savsk \lastskip
      \@savsf \spacefactor
    \fi
  \fi
}
```

I think that, in vmode, it is the safest to put in a `\nobreak` immediately after such things since writes, inserts etc followed by glue give valid breakpoints and, in general, it is possible to create breaks but impossible to destroy them.

```
def \@esphack{%
  \relax \ifvmode
    \nobreak
    \ifdim \@savsk=\z@
    \else
      \vskip \@savsk
    \fi
  \else
    \ifhmode
      \spacefactor \@savsf
      \ifdim \@savsk>\z@
        \ignorespaces
      \fi
    \fi
  \fi
}
```

```

\fi
\fi

```

For the moment we are going to ignore the vertical versions until they are correct.

```

125 \def\bsphack{%
126   \relax
127   \ifhmode
128     \savsk\lastskip
129     \savsf\spacefactor
130   \fi}

```

(End definition for \bsphack.)

\@esphack Companion to \bsphack. If this command is not properly paired with \bsphack one might end up with a low-level T_EX error: “BAD spacefactor”. One possible cause is calling \bsphack in vertical mode, then doing something that gets you (sometimes) into horizontal mode and finally calling \@esphack. Even if no error is generated that is wrong, because \@esphack will then use the saved values for \savsk and \savsf from some earlier invocation of \bsphack which will have nothing to do with the current situation.

```

131 </2kernel>
132 <latexrelease>\IncludeInRelease{2018/10/10}%
133 <latexrelease>          {\@esphack}{hyphenation and nobreak after space hack}%
134 <*2kernel | latexrelease>
135 \def\@esphack{%
136   \relax
137   \ifhmode
138     \spacefactor\@savsf
139     \ifdim\@savsk>\z@
140       \ifdim\lastskip=\z@
141         \nobreak \hskip\z@skip
142       \fi
143       \ignorespaces
144     \fi
145   \else
146     \ifvmode
147       \if@nobreak\nobreak\else\if@noskipsec\nobreak\fi\fi
148     \fi
149   \fi}%
150 </2kernel | latexrelease>
151 <latexrelease>\EndIncludeInRelease
152 <latexrelease>\IncludeInRelease{2015/10/01}%
153 <latexrelease>          {\@esphack}{hyphenation and nobreak after space hack}%
154 <latexrelease>\def\@esphack{%
155 <latexrelease>  \relax
156 <latexrelease>  \ifhmode
157 <latexrelease>    \spacefactor\@savsf
158 <latexrelease>    \ifdim\@savsk>\z@
159 <latexrelease>      \ifdim\lastskip=\z@
160 <latexrelease>        \nobreak \hskip\z@skip
161 <latexrelease>      \fi

```

```

162 <latexrelease> \ignorespaces
163 <latexrelease> \fi
164 <latexrelease> \fi}%
165 <latexrelease>\EndIncludeInRelease
166 <latexrelease>\IncludeInRelease{2015/01/01}%
167 <latexrelease> {\@esphack}{hyphenation and nobreak after space hack}%
168 <latexrelease>\def\@esphack{%
169 <latexrelease> \relax
170 <latexrelease> \ifhmode
171 <latexrelease> \spacefactor\@savsf
172 <latexrelease> \ifdim\@savsk>\z@
173 <latexrelease> \nobreak \hskip\z@skip
174 <latexrelease> \ignorespaces
175 <latexrelease> \fi
176 <latexrelease> \fi}%
177 <latexrelease>\EndIncludeInRelease
178 <latexrelease>\IncludeInRelease{0000/00/00}%
179 <latexrelease> {\@esphack}{hyphenation and nobreak after space hack}%
180 <latexrelease>\def\@esphack{%
181 <latexrelease> \relax
182 <latexrelease> \ifhmode
183 <latexrelease> \spacefactor\@savsf
184 <latexrelease> \ifdim\@savsk>\z@
185 <latexrelease> \ignorespaces
186 <latexrelease> \fi
187 <latexrelease> \fi}%
188 <latexrelease>\EndIncludeInRelease
189 <*2ekernel>

```

(End definition for \@esphack.)

\@Esphack A variant of \@esphack that sets the @ignore switch to true (as \@esphack used to do previously). This is currently used only for floats and similar environments. w

```

190 </2ekernel>
191 <latexrelease>\IncludeInRelease{2015/01/01}%
192 <latexrelease> {\@Esphack}{hyphenation after space hack}%
193 <*2ekernel | latexrelease>
194 \def\@Esphack{%
195 \relax
196 \ifhmode
197 \spacefactor\@savsf
198 \ifdim\@savsk>\z@
199 \nobreak \hskip\z@skip
200 \@ignoretrue
201 \ignorespaces
202 \fi
203 \fi}%
204 </2ekernel | latexrelease>
205 <latexrelease>\EndIncludeInRelease
206 <latexrelease>\IncludeInRelease{0000/00/00}%
207 <latexrelease> {\@Esphack}{hyphenation after space hack}%
208 <latexrelease>\def\@Esphack{%
209 <latexrelease> \relax
210 <latexrelease> \ifhmode

```

```

211 <latexrelease> \spacefactor\@savsf
212 <latexrelease> \ifdim\@savsk>\z@
213 <latexrelease> \ignoretrue
214 <latexrelease> \ignorespaces
215 <latexrelease> \fi
216 <latexrelease> \fi}%
217 <latexrelease>\EndIncludeInRelease
218 <*2ekernel>

```

(End definition for \@Esphack.)

\@vbsphack Another variant which is useful for invisible things which should not live in vmode (this is how some people feel about marginals).

If it occurs in vmode then it enters hmode and ensures that \@savsk is nonzero so that the \ignorespaces is put in later. It is not used at present.

```

\def \@vbsphack{ %
  \relax \ifvmode
    \leavevmode
    \@savsk 1sp
    \@savsf \spacefactor
  \else
    \ifhmode
      \@savsk \lastskip
      \@savsf \spacefactor
    \fi
  \fi
}

```

(End definition for \@vbsphack.)

1.5 Vertical spacing

L^AT_EX supports the plain T_EX commands \smallskip, \medskip and \bigskip. However, it redefines them using \vspace instead of \vskip.

Extra vertical space is added by the command \addvspace{<skip>}, which adds a vertical skip of <skip> to the document. The sequence \addvspace{<s1>} \addvspace{<s2>} is equivalent to \addvspace{<maximum of s1, s2>}.

\addvspace should be used only in vertical mode, and gives an error if it's not. The \addvspace command does *not* add vertical space if @minipage is true. The minipage environment uses this to inhibit the addition of extra vertical space at the beginning.

Penalties are put into the vertical list with the \addpenalty{<penalty>} command. It works properly when \addpenalty and \addvspace commands are mixed.

The @nobreak switch is set true used when in vertical mode and no page break should occur. (Right now, it is used only by the section heading commands to inhibit page breaking after a heading.)

```

\addvspace{SKIP} ==
BEGIN
  if vmode
    then if @minipage

```



```

        else if \lastskip =0
        then \vskip SKIP
        else if \lastskip < SKIP
            then \vskip -\lastskip
            \vskip SKIP
            else if SKIP < 0 and \lastskip >= 0
            then \vskip -\lastskip
            \vskip \lastskip + SKIP
        fi      fi      fi      fi
    else useful error message (CAR).
fi
END

```

\@xaddvskip Internal macro for \vspace handling the case that space has previously been added.

```

219 \def\@xaddvskip{%
220   \ifdim\lastskip<\@tempskipb
221     \vskip-\lastskip
222     \vskip\@tempskipb
223   \else
224     \ifdim\@tempskipb<\z@
225       \ifdim\lastskip<\z@
226         \else
227           \advance\@tempskipb\lastskip
228           \vskip-\lastskip
229           \vskip \@tempskipb
230         \fi
231       \fi
232     \fi}

```

(End definition for \@xaddvskip.)

\addvspace Add vertical space taking into account space already added, as described above.

```

233 </2ekernel>
234 < *2ekernel | latexrelease>
235 < latexrelease> \IncludeInRelease{2020/10/01}%
236 < latexrelease> { \addvspace } { \addvspace calc support }%
237 \def\addvspace#1{%
238   \ifvmode
239     \if@minipage\else
240       \ifdim \lastskip =\z@
241         \@vspace@calcify{#1}%
242       \else
243         \setlength\@tempskipb{#1}%
244         \@xaddvskip
245       \fi
246     \fi
247   \else
248     \@noitemerr
249   \fi}
250 </2ekernel | latexrelease>
251 < latexrelease> \EndIncludeInRelease
252 < latexrelease> \IncludeInRelease{0000/00/00}%
253 < latexrelease> { \addvspace } { \addvspace calc support }%

```

```

254 <latexrelease>
255 <latexrelease>\def\addvspace#1{%
256 <latexrelease> \ifvmode
257 <latexrelease> \if@minipage\else
258 <latexrelease> \ifdim \lastskip =\z@
259 <latexrelease> \vskip #1\relax
260 <latexrelease> \else
261 <latexrelease> \@tempskipb#1\relax
262 <latexrelease> \@xaddvskip
263 <latexrelease> \fi
264 <latexrelease> \fi
265 <latexrelease> \else
266 <latexrelease> \@noitemerr
267 <latexrelease> \fi}
268 <latexrelease>\EndIncludeInRelease
269 <*2kernel>

```

(End definition for \addvspace.)

\addpenalty

```

270 </2kernel>
271 <latexrelease>\IncludeInRelease{2015/01/01}%
272 <latexrelease> \{\addpenalty\}\addpenalty}%
273 <*2kernel | latexrelease>

```

Fix provided by Donald (though the original fix was not good enough). In 2005 Plamen Tanovski discovered that this fix wasn't good enough either as the \vskip kept getting bigger if several \addpenalty commands followed each other. Donald kindly send a new fix.

```

274 \def\addpenalty#1{%
275 \ifvmode
276 \if@minipage
277 \else
278 \if@nobreak
279 \else
280 \ifdim\lastskip=\z@
281 \penalty#1\relax
282 \else
283 \@tempskipb\lastskip

```

We have to make sure the final \vskip seen by T_EX is the correct one, namely \@tempskipb. However we may have to adjust for \prevdepth when placing the penalty but that should not affect the skip we pass on to T_EX.

```

284 \begingroup
285 \@tempskipa\@tempskipb
286 \advance \@tempskipb
287 \ifdim\prevdepth>\maxdepth\maxdepth\else

```

If \prevdepth is -1000pt due to \nointerlineskip we better not add it!

```

288 \ifdim \prevdepth = -\@m\p@ \z@ \else \prevdepth \fi
289 \fi
290 \vskip -\@tempskipb
291 \penalty#1%
292 \ifdim\@tempskipa=\@tempskipb

```

Do nothing if the `\prevdepth` check made no adjustment.

```

293         \else
Combine the prevdepth adjustment into a single skip.
294         \advance\@tempskipb -\@tempskipa
295         \vskip \@tempskipb
296     \fi
The final skip is always the specified length.
297     \vskip \@tempskipa
298 \endgroup
299 \fi
300 \fi
301 \fi
302 \else
303     \@noitemerr
304 \fi}%

305 \</2ekernel | latexrelease>
306 \<latexrelease>\EndIncludeInRelease
307 \<latexrelease>\IncludeInRelease{0000/00/00}%
308 \<latexrelease>        {\addpenalty}{\addpenalty}%
309 \<latexrelease>\def\addpenalty#1{%
310 \<latexrelease> \ifvmode
311 \<latexrelease>     \if@minipage
312 \<latexrelease>     \else
313 \<latexrelease>     \if@nobreak
314 \<latexrelease>     \else
315 \<latexrelease>     \ifdim\lastskip=\z@
316 \<latexrelease>     \penalty#1\relax
317 \<latexrelease>     \else
318 \<latexrelease>     \@tempskipb\lastskip
319 \<latexrelease>     \vskip -\lastskip
320 \<latexrelease>     \penalty#1%
321 \<latexrelease>     \vskip\@tempskipb
322 \<latexrelease>     \fi
323 \<latexrelease>     \fi
324 \<latexrelease>     \fi
325 \<latexrelease> \else
326 \<latexrelease>     \@noitemerr
327 \<latexrelease> \fi}%
328 \<latexrelease>\EndIncludeInRelease
329 \<*2ekernel>

```

(End definition for `\addpenalty`.)

`\vspace`
`\@vspace`
`\@vspacer`

The new code for these commands depends on the following facts:

- The value of `prevdepth` is changed only when a box or rule is created and added to a vertical list;
- The value of `prevdepth` is used only when a box is created and added to a vertical list;
- The value of `prevdepth` is always local to the building of one vertical list.

```

330 \DeclareRobustCommand\vspace{\@ifstar\@vspacer\@vspace}

```

```

331 </2ekernel>
332 <*2ekernel | latexrelease>
333 <latexrelease>\IncludeInRelease{2020/10/01}%
334 <latexrelease>          {\@vspace}{Support calc in \vspace}%

```

We support calc syntax in the argument and therefore use \setlength.

```

335 \def\@vspace #1{%
336   \ifvmode
337     \@vspace@calcify{#1}%
338     \vskip\z@skip
339   \else
340     \@bsphack
341     \vadjust{\@restorepar
342               \@vspace@calcify{#1}%
343               \vskip\z@skip
344             }%
345     \@esphack
346   \fi}

347 \def\@vspacer#1{%
348   \ifvmode
349     \dimen@\prevdepth
350     \hrule \@height\z@
351     \nobreak
352     \@vspace@calcify{#1}%
353     \vskip\z@skip
354     \prevdepth\dimen@
355   \else
356     \@bsphack
357     \vadjust{\@restorepar
358               \hrule \@height\z@
359               \nobreak
360               \@vspace@calcify{#1}%
361               \vskip\z@skip}%
362     \@esphack
363   \fi}

364 </2ekernel | latexrelease>
365 <latexrelease>\EndIncludeInRelease
366 <latexrelease>\IncludeInRelease{0000/00/00}%
367 <latexrelease>          {\@vspace}{Support calc in \vspace}%
368 <latexrelease>

369 <latexrelease>\def\@vspace #1{%
370 <latexrelease>  \ifvmode
371 <latexrelease>    \vskip #1
372 <latexrelease>    \vskip\z@skip
373 <latexrelease>  \else
374 <latexrelease>    \@bsphack
375 <latexrelease>    \vadjust{\@restorepar
376 <latexrelease>                  \vskip #1
377 <latexrelease>                  \vskip\z@skip
378 <latexrelease>                }%
379 <latexrelease>    \@esphack
380 <latexrelease>  \fi}
381 <latexrelease>\def\@vspacer#1{%
382 <latexrelease>  \ifvmode

```

```

383 <latexrelease> \dimen@prevdepth
384 <latexrelease> \hrule \@height\z@
385 <latexrelease> \nobreak
386 <latexrelease> \vskip #1
387 <latexrelease> \vskip\z@skip
388 <latexrelease> \prevdepth\dimen@
389 <latexrelease> \else
390 <latexrelease> \@bsphack
391 <latexrelease> \vadjust{\@restorepar
392 <latexrelease> \hrule \@height\z@
393 <latexrelease> \nobreak
394 <latexrelease> \vskip #1
395 <latexrelease> \vskip\z@skip}%
396 <latexrelease> \@esphack
397 <latexrelease> \fi}
398 <latexrelease>\EndIncludeInRelease
399 <*2ekernel>

```

(End definition for `\vspace`, `\@vspace`, and `\@vspacer`.)

```

\smallskip
\medskip 400 \def\smallskip{\vspace\smallskipamount}
\bigskip 401 \def\medskip{\vspace\medskipamount}
         402 \def\bigskip{\vspace\bigskipamount}

```

(End definition for `\smallskip`, `\medskip`, and `\bigskip`.)

```

\smallskipamount
\medskipamount 403 \newskip\smallskipamount \smallskipamount=3pt plus 1pt minus 1pt
\bigskipamount 404 \newskip\medskipamount \medskipamount =6pt plus 2pt minus 2pt
               405 \newskip\bigskipamount \bigskipamount =12pt plus 4pt minus 4pt

```

(End definition for `\smallskipamount`, `\medskipamount`, and `\bigskipamount`.)

1.6 Horizontal space (and breaks)

`\nobreakdashes` This idea is borrowed from the `amsmath` package but here we define a robust command. This command is a low-level command designed for use only before hyphens or dashes (such as `-`, `--`, or `---`).

It could probably be better implemented: it may need its own private token register and temporary command.

Setting the hyphen in a box and then unboxing it means that the normal penalty will not be added after it—and if the penalty is not there a break will not be taken (unless an explicit penalty or glue follows, thus the final `\nobreak`).

Note that even if it is not followed by a `'-`, it still leaves vmode and sets the spacefactor; so use it carefully!

```

406 \DeclareRobustCommand{\nobreakdashes}{%
407   \leavevmode
408   \toks@{}%
409   \def\reserved@a##1{\toks@\expandafter{\the\toks@-}%
410     \futurelet\@let@token \reserved@b}%
411   \def\reserved@b  {\ifx\@let@token -%
412     \expandafter\reserved@a

```

```

413             \else
414             \setbox\z@ \hbox{\the\toks@\nobreak}%
415             \unhbox\z@
416             \spacefactor\sfcode'\-
417             \fi}%
418 \futurelet\@let@token \reserved@b
419 }

```

(End definition for \nobreakdashes.)

\nobreakspace This is a robust command that produces a horizontal space at which, in paragraph-mode, a line-break is not possible. We then define an active ~ to expand to it since this is the documented behaviour of ~. One reason for introducing this is that some 8-bit input encodings have a slot for such a space and we do not want to use active characters as the L^AT_EX internal commands.

\@xobeysp

The braces in the definition of ~ are needed to ensure that a following space is preserved when reading to/from internal files.

We need to keep \@xobeysp as it is widely used; so here it is let to the non-robust command \nobreakspace .

```

420 \DeclareRobustCommand{\nobreakspace}{%
421   \leavevmode\nobreak\ }
422 \catcode '\~=13
423 \def~{\nobreakspace{}}
424 \expandafter\let\expandafter\@xobeysp\csname nobreakspace \endcsname

```

(End definition for \nobreakspace and \@xobeysp.)

\@ Placed before a ' ', makes it a sentence-ending period. Does the right thing for other punctuation marks as well. Does this by setting spacefactor to 1000.

```

425 </2ekernel>
426 <latexrelease>\IncludeInRelease{2015/01/01}%
427 <latexrelease>          {\@}{Space after \@}%
428 <*2ekernel | latexrelease>
429 \def\@{\spacefactor\@m{}}%
430 </2ekernel | latexrelease>
431 <latexrelease>\EndIncludeInRelease
432 <latexrelease>\IncludeInRelease{0000/00/00}%
433 <latexrelease>          {\@}{Space after \@}%
434 <latexrelease>\def\@{\spacefactor\@m}%
435 <latexrelease>\EndIncludeInRelease
436 <*2ekernel>

```

(End definition for \@.)

\hspace

```

437 \DeclareRobustCommand\hspace{\@ifstar\@hspacer\@hspace}

```

(End definition for \hspace.)

\@hspace

```
438 </2ekernel>
439 <*2ekernel | latexrelease>
440 <latexrelease>\IncludeInRelease{2020/10/01}%
441 <latexrelease>{\@hspace}{Support calc with \hspace}%
442 \def\@hspace#1{\begingroup\setlength\skip@{#1}\hskip\skip@\endgroup}
443 </2ekernel | latexrelease>
444 <latexrelease>\EndIncludeInRelease
445 <latexrelease>\IncludeInRelease{0000/00/00}%
446 <latexrelease>{\@hspace}{Support calc with \hspace}%
447
448 <latexrelease>
449 <latexrelease>\def\@hspace#1{\hskip #1\relax}
450 <latexrelease>\EndIncludeInRelease
451 <*2ekernel>
```

(End definition for \@hspace.)

\@hspacer Extra \hskip Opt added 1985/17/12 to guard against a following \unskip \relax added 13 Oct 88 for usual T_EX lossage replaced both changes by \hskip\z@skip 27 Nov 91

```
452 \def\@hspacer#1{\vrule \@width\z@\nobreak
453 \hspace{#1}\hskip \z@skip}
```

(End definition for \@hspacer.)

\fill

```
454 \newskip\fill
455 \fill = Opt plus 1fill
```

(End definition for \fill.)

\stretch

```
456 \def\stretch#1{\z@ \@plus #1fill\relax}
```

(End definition for \stretch.)

```
457 </2ekernel>
458 <*2ekernel | latexrelease>
459 <latexrelease>\IncludeInRelease{2018/12/01}%
460 <latexrelease>{\thinspace}{Start LR-mode}%
```

\enspace

```
461 \DeclareRobustCommand\enspace{\leavevmode@ifvmode\kern.5em }
```

(End definition for \enspace.)

\leavevmode@ifvmode Leave vmode but only if we are really in vmode, otherwise the expansion is empty (which is not the case with the default definition).

```
462 \protected\def\leavevmode@ifvmode{\ifvmode\expandafter\indent\fi}
```

```

(End definition for \leavevmode@ifvmode.)

463 </2ekernel | latexrelease>
464 <latexrelease>\EndIncludeInRelease
465 <latexrelease>\IncludeInRelease{0000/00/00}%
466 <latexrelease>{\thinspace}{Start LR-mode}%
467 <latexrelease>\def\thinspace{\kern .16667em }
468 <latexrelease>\def\negthinspace{\kern-.16667em }
469 <latexrelease>\def\enspace{\kern.5em }
470 <latexrelease>\let\leavevmode@ifvmode\@undefined
471 <latexrelease>\EndIncludeInRelease
472 <*2ekernel>

\enskip
\quad 473 \def\enskip{\hskip.5em\relax}
\qquad 474 \def\quad{\hskip1em\relax}
475 \def\qquad{\hskip2em\relax}

(End definition for \enskip, \quad, and \qquad.)
For Unicode engines, make the Unicode soft hyphen an active character defined as
\-.

476 \ifx\Umathcode\@undefined\else
477 \catcode "AD=13
478 \def^^ad{-}
479 \fi

\obeycr The following definitions will probably get deleted or moved to compatibility mode soon.
\restorecr
480 {\catcode'\^^M=13 \gdef\obeycr{\catcode'\^^M13 \def^^M{\\relax}%
481 \gobblecr}%
482 {\catcode'\^^M=13 \gdef\gobblecr{\@ifnextchar
483 \gobble\ignorespaces}}
484 \gdef\restorecr{\catcode'\^^M5 }}

(End definition for \obeycr and \restorecr.)

485 </2ekernel>

```


File p

ltlogos.dtx

1 Logos

Various logos are defined here.

`\TeX` The $\mathrm{T}_{\mathrm{E}}\mathrm{X}$ logo, adjusted so that a full stop after the logo counts as ending a sentence.

```
1 <*2ekernel>
2 \DeclareRobustCommand\TeX{T\kern-.1667em\lower.5ex\hbox{E}\kern-.125emX\@}
```

(End definition for \TeX.)

`\LaTeX` The $\mathrm{L}^{\mathrm{A}}\mathrm{T}_{\mathrm{E}}\mathrm{X}$ logo.

```
3 \DeclareRobustCommand{\LaTeX}{L\kern-.36em%
4   {\sbox\z@ T%
5     \vbox to\ht\z@{\hbox{\check@mathfonts
6       \fontsize\sf@size\z@
7       \math@fontsfalse\selectfont
8       A}%
9       \vss}%
10  }%
11  \kern-.15em%
12  \TeX}
```

(End definition for \LaTeX.)

`\LaTeXe` The $\mathrm{L}^{\mathrm{A}}\mathrm{T}_{\mathrm{E}}\mathrm{X} 2_{\epsilon}$ logo as proposed by A-W designers.

```
13 \DeclareRobustCommand{\LaTeXe}{\mbox{\m@th
14   \if b\expandafter\@car\f@series\@nil\boldmath\fi
15   \LaTeX\kern.15em2$_{\textstyle\varepsilon}$}}
16 </2ekernel>
```

(End definition for \LaTeXe.)

File q

ltfiles.dtx

1 File Handling

The following user commands are defined in this part:

<code>\document</code>	(ie <code>\begin{document}</code>)
	Reads in the .AUX files and <code>\catcode</code> 's @ to 12.
<code>\nofiles</code>	
	Suppresses all file output by setting <code>\@files</code> false.
<code>\includeonly</code>	<code>{\NAME1, ... ,NAMEn}</code>
	Causes only parts NAME1, ... ,NAMEn to be read by their <code>\include</code> commands. Works by setting <code>partsw</code> true and setting <code>\@partlist</code> to NAME1, ... ,NAMEn.
<code>\include</code>	<code>{\NAME}</code>
	Does an <code>\input NAME</code> unless <code>\@partsw</code> is true and NAME is not in <code>\@partlist</code> . If <code>\@files</code> is true, then it directs .AUX output to NAME.AUX, including a checkpoint at the end.
<code>\input</code>	<code>{\NAME}</code>
	The same as TeX's <code>\input</code> , except it allows optional braces around the file name. In $\text{\LaTeX 2}_{\epsilon}$, it also avoids the primitive 'missing file' error, if the file can not be found.
<code>\IfFileExists</code>	<code>{\NAME}{\then}{\else}</code>
	If the file exists on the system, execute <i>then</i> otherwise execute <i>else</i> .
<code>\InputIfFileExists</code>	<code>{\NAME}{\then}{\else}</code>
	If the file exists on the system, execute <i>then</i> and input NAME otherwise execute <i>else</i> . <i>Historical \LaTeX 2.09 comments (not necessarily accurate any more):</i>

```

1 \*2kernel)
2 \message{files,}

```

VARIABLES, SWITCHES AND INTERNAL COMMANDS:

`\@mainaux` : Output file number for main .AUX file.
`\@partaux` : Output file number for current part's .AUX file.
`\@auxout` : Either `\@mainout` or `\@partout`, depending on which .AUX file output goes to.
`\@input{foo}` : If file foo exists, then `\input`'s it, otherwise types a warning message.
`@files` : Switch – set false if no .AUX, .TOC, .IDX etc files are to be written
`@partsw` : Set true by a `\includeonly` command.
`\@partlist` : Set to the argument of the `\includeonly` command.

`\cp@F00` : The checkpoint for `\include`'d file FOO.TEX, written by `\@writeckpt` at the end of file FOO.AUX

```

\includeonly{FILELIST} ==
BEGIN
  \@partsw := T

```

```

\@partlist := FILELIST
END

\include{FILE} ==
BEGIN
  \clearpage
  if \@files = T
    then \immediate\write\@mainaux{\string\@input{FILE.AUX}}
  fi
  if \@partsw = T
    then \@tempwa := F
    \reserved@a := FILE
    for \reserved@a := \@partlist
      do if eval(\reserved@a) = eval(\reserved@b)
        then \@tempwa := T          fi
      od
    fi
  fi

  if \@tempwa = T
    then \@auxout := \@partaux
    if \@files = T
      then \immediate\openout\@partaux{FILE.AUX}
      \immediate\write\@partaux{\relax}
    fi
    \input{FILE.TEX}
    \clearpage
    \writeckpt{FILE}
    if @files then \closeout \@partaux fi
    \@auxout := \@mainaux
  else \cp@FILE
  fi
END

\@writeckpt{FILE} ==
BEGIN
  if \@files = T
    \immediate\write on file \@partaux:
      \setckpt{FILE}{                                %% }
    for \reserved@a := \cl@ckpt
      do \immediate\write on file \@partaux:
        \global\string\setcounter
          {eval(\reserved@a)}{eval(\c@eval(\reserved@a))}
      od
    \immediate\write on file \@partaux:  }
  fi
END

\@setckpt{FILE}{LIST} ==
BEGIN
  G \cp@FILE := LIST

```

END

INITIALIZATION

`\@tempswa := T`

End of historical L^AT_EX 2.09 comments.

`\@mainaux`

`\@partaux`

3 `\newwrite\@mainaux`

4 `\newwrite\@partaux`

(End definition for \@mainaux and \@partaux.)

`\if@files`

`\if@partsw`

5 `\newif\if@files \@files>true`

6 `\newif\if@partsw \@partsw>false`

(End definition for \if@files and \if@partsw.)

`\@clubpenalty` This stores the current normal (non-infinite) value of `\clubpenalty`; it should therefore be reset whenever the normal value is changed (as in the bibliography in the standard styles).

7 `\newcount\@clubpenalty`

8 `\@clubpenalty \clubpenalty`

(End definition for \@clubpenalty.)

`\document`

9 `\</2ekernel`

10 `\<latexrelease>\IncludeInRelease{2020/10/01}%`

11 `\<latexrelease> {\document}{Added hook to load l3backend code}%`

12 `\<*2ekernel|latexrelease>`

13 `\def\document{%`

We do cancel the grouping as part of the `\begin` handling (this is now done inside `\begin` instead) so that the `env/<env>/begin` hook is not hidden inside `\begingroup ... \endgroup`.

14 `% \endgroup`

15 `\UseOneTimeHook{begindocument/before}%`

16 `\@kernel@after@begindocument@before`

Added hook to load l3backend code:

17 `\@expl@sys@load@backend@@`

18 `\ifx\@unusedoptionlist\empty\else`

19 `\@latex@warning@no@line{Unused global option(s):^^J%`

20 `\@spaces[\@unusedoptionlist]}%`

21 `\fi`

22 `\@colht\textheight`

23 `\@colroom\textheight \vsize\textheight`

24 `\columnwidth\textwidth`

25 `\@clubpenalty\clubpenalty`

26 `\if@twocolumn`

27 `\advance\columnwidth -\columnsep`

28 `\divide\columnwidth\tw@ \hsize\columnwidth \@firstcolumntrue`

```

29 \fi
30 \hsize\columnwidth \linewidth\hsize
31 \begingroup\@floatplacement\@dblfloatplacement
32 \makeatletter\let\@writefile\@gobbletwo
33 \global \let \@multiplelabels \relax
34 \input{\jobname.aux}%
35 \endgroup
36 \if@files
37 \immediate\openout\@mainaux\jobname.aux
38 \immediate\write\@mainaux{\relax}%
39 \fi

```

Dateline 1991/03/26: FMi added `\process@table` to support NFSS; This will also work with old fonts if no other style defines `\process@table`. The following line forces the initialization of the math fonts.

```

40 \process@table
41 \let\glb@currsize\@empty % Force math initialization.
42 \normalsize
43 \everypar{}%

```

So that punctuation in headings is not disturbed by verbatim or other local changes to the space factor codes, save the document default here. This will be locally reset by the output routine. For special cases a class may want to define `\normalsfcodes` directly, in case that definition will be used. (This is an old bug, problem existed in L^AT_EX 2.0x and plain T_EX.)

```

44 \ifx\normalsfcodes\@empty
45 \ifnum\sfcode'\.=\@m
46 \let\normalsfcodes\frenchspacing
47 \else
48 \let\normalsfcodes\nonfrenchspacing
49 \fi
50 \fi

```

For similar reasons also save the default language, this will be reset locally in the output routine. In particular it allows hyphenation in the page head even if the page break happens in verbatim. If this has already been set by a package, set to the value of `\language` at this point.

```

51 \ifx\document@default@language\m@ne
52 \chardef\document@default@language\language
53 \fi

```

Way back in 1991 (08/26) FMi & RmS set the `\@noskipsec` switch to true in the preamble and to false here. This was done to trap lists and related text in the preamble but it does not catch everything; hence Change 1.1g was introduced.

```

54 \@noskipsecfalse
55 \let \@refundefined \relax

```

Just before disabling the preamble commands we execute the begin document hook which contains any code contributed by `\AtBeginDocument`. Also disable the gathering of the file list, if no `\listfiles` has been issued. `\AtBeginDocument` is redefined at this point so that and such commands that get into the hook do not chase their tail...

```

56 \@kernel@before@begindocument

```

```

57 \UseOneTimeHook{begindocument}%
58 \@kernel@after@begindocument

```

Most of the following assignments will be done globally in case the user adds something like `\begin{multicols}` to the document hook, i.e. starts are group in `\begin{document}`.

Since a value of exactly 0pt for `\topskip` causes `\twocolumn[]` to misbehave, we add this check, hoping that it will not cause any problems elsewhere.

```

59 \ifdim\topskip<1sp\global\topskip 1sp\relax\fi
60 \global\@maxdepth\maxdepth
61 \global\let\@begindocumenthook\@undefined
62 \ifx\@listfiles\@undefined
63   \global\let\@filelist\relax
64   \global\let\@addtofilelist\@gobble
65 \fi

```

At the very end we disable all preamble commands. This has to happen after the begin document hooks was executed so that this hook can still use such commands.

```

66 \gdef\do##1{\global\let ##1\@notprerr}%
67 \@preamblecmds

```

The next line saves tokens and also allows `\@nodocument` to be used directly to trap preamble errors.

```

68 \global\let \@nodocument \relax

```

The next line is a pure safety measure in case a do list is ever expanded at the wrong place. In addition it will save a few tokens to get rid of the above definition.

```

69 \global\let\do\noexpand
70 \UseOneTimeHook{begindocument/end}%

```

Use of the hook might mean that we are already in horizontal mode, so ignore the space after `\begin{document}`.

```

71 \ignorespaces

```

The `begindocument` hook already existed in the kernel since 1994 under the name `\atbegindocumenthook` the additional ones are originally from the `etoolbox` package under the names `\@endpreamblehook` `\afterpreamble`.

```

72 \NewHook{begindocument}
73 \NewHook{begindocument/before}
74 \NewHook{begindocument/end}

```

Above we used two kernel only hooks to be run after the public `begindocument/before` and after `begindocument` hooks.

In `\@kernel@after@begindocument@before` we already place one action: drop the fast execution code for the `env/document/begin` hook. That hook marks the end of the preamble and should therefore only be run once. In a normal document that is anyway the case (so the code would just sit there taking up space afterwards, which these days is rather harmless), however, in more complicated scenarios where several full documents are combined to a single document it might get applied several times with harmful effects. We therefore explicitly drop it at this point. the coding is somewhat obscure due to the name of the macro which requires constructing.

```

75 \edef \@kernel@after@begindocument@before {%
76   \let\expandafter\noexpand\csname
77     __hook env/document/begin\endcsname
78   \noexpand\@empty}

```

```

\@kernel@after@begindocument@before
\@kernel@before@begindocument
\@kernel@after@begindocument

```

These internal hooks are already declared earlier (in `ltxexpl`) so that other modules could write to them.

```

79 %\let \@kernel@before@begindocument \@empty
80 %\let \@kernel@after@begindocument \@empty

81 </2ekernel | latexrelease>
82 <latexrelease>\EndIncludeInRelease

83 <latexrelease>\IncludeInRelease{2017/04/15}%
84 <latexrelease> {\document}{Save language for hyphenation}%
85 <latexrelease>
86 <latexrelease>\def\document{\endgroup
87 <latexrelease> \ifx\@unusedoptionlist\@empty\else
88 <latexrelease> \@latex@warning@no@line{Unused global option(s):^^J%
89 <latexrelease> \spaces[\@unusedoptionlist]}%
90 <latexrelease> \fi
91 <latexrelease> \@colht\textheight
92 <latexrelease> \@colroom\textheight \vsize\textheight
93 <latexrelease> \columnwidth\textwidth
94 <latexrelease> \@clubpenalty\clubpenalty
95 <latexrelease> \if@twocolumn
96 <latexrelease> \advance\columnwidth -\columnsep
97 <latexrelease> \divide\columnwidth\tw@ \hsize\columnwidth \@firstcolumntrue
98 <latexrelease> \fi
99 <latexrelease> \hsize\columnwidth \linewidth\hsize
100 <latexrelease> \begingroup\@floatplacement\@dblfloatplacement
101 <latexrelease> \makeatletter\let\@writefile\@gobbletwo
102 <latexrelease> \global \let \@multiplelabels \relax
103 <latexrelease> \input{\jobname.aux}%
104 <latexrelease> \endgroup
105 <latexrelease> \if@filesw
106 <latexrelease> \immediate\openout\@mainaux\jobname.aux
107 <latexrelease> \immediate\write\@mainaux{\relax}%
108 <latexrelease> \fi
109 <latexrelease> \process@table
110 <latexrelease> \let\glb@currsz\@empty % Force math initialization.
111 <latexrelease> \normalsize
112 <latexrelease> \everypar{}%
113 <latexrelease> \ifx\normalsfcodes\@empty
114 <latexrelease> \ifnum\sfcode'\.=\@m
115 <latexrelease> \let\normalsfcodes\frenchspacing
116 <latexrelease> \else
117 <latexrelease> \let\normalsfcodes\nonfrenchspacing
118 <latexrelease> \fi
119 <latexrelease> \fi
120 <latexrelease> \ifx\document@default@language\m@ne
121 <latexrelease> \chardef\document@default@language\language
122 <latexrelease> \fi
123 <latexrelease> \noskipsecfalse
124 <latexrelease> \let \crefundefined \relax
125 <latexrelease> \let \AtBeginDocument \@firstofone
126 <latexrelease> \@begindocumenthook
127 <latexrelease> \ifdim\topskip<1sp\global\topskip 1sp\relax\fi
128 <latexrelease> \global\@maxdepth\maxdepth
129 <latexrelease> \global\let\@begindocumenthook\@undefined

```

```

130 <latexrelease> \ifx\@listfiles\@undefined
131 <latexrelease> \global\let\@filelist\relax
132 <latexrelease> \global\let\@addtofilelist\@gobble
133 <latexrelease> \fi
134 <latexrelease> \gdef\do##1{\global\let ##1\@notprerr}%
135 <latexrelease> \@preamblecmds
136 <latexrelease> \global\let \@nodocument \relax
137 <latexrelease> \global\let\do\noexpand
138 <latexrelease> \ignorespaces}
139 <latexrelease>\EndIncludeInRelease
140 <latexrelease>
141 <latexrelease>\IncludeInRelease{0000/00/00}%
142 <latexrelease> {\document}{Save language for hyphenation}
143 <latexrelease>\def\document{\endgroup
144 <latexrelease> \ifx\@unusedoptionlist\@empty\else
145 <latexrelease> \@latex@warning@no@line{Unused global option(s):^^J%
146 <latexrelease> \@spaces[\@unusedoptionlist]}}%
147 <latexrelease> \fi
148 <latexrelease> \@colht\textheight
149 <latexrelease> \@colroom\textheight \vsize\textheight
150 <latexrelease> \columnwidth\textwidth
151 <latexrelease> \@clubpenalty\clubpenalty
152 <latexrelease> \if@twocolumn
153 <latexrelease> \advance\columnwidth -\columnsep
154 <latexrelease> \divide\columnwidth\tw@ \hsize\columnwidth
155 <latexrelease> \@firstcolumntrue
156 <latexrelease> \fi
157 <latexrelease> \hsize\columnwidth \linewidth\hsize
158 <latexrelease> \begingroup\@floatplacement\@dblfloatplacement
159 <latexrelease> \makeatletter\let\@writefile\@gobbletwo
160 <latexrelease> \global \let \@multiplelabels \relax
161 <latexrelease> \input{\jobname.aux}%
162 <latexrelease> \endgroup
163 <latexrelease> \if@files
164 <latexrelease> \immediate\openout\@mainaux\jobname.aux
165 <latexrelease> \immediate\write\@mainaux{\relax}%
166 <latexrelease> \fi
167 <latexrelease> \process@table
168 <latexrelease> \let\glb@currsz\@empty
169 <latexrelease> \normalsize
170 <latexrelease> \everypar{}%
171 <latexrelease> \ifx\normalsfcodes\@empty
172 <latexrelease> \ifnum\sfcode'\.=\@m
173 <latexrelease> \let\normalsfcodes\frenchspacing
174 <latexrelease> \else
175 <latexrelease> \let\normalsfcodes\nonfrenchspacing
176 <latexrelease> \fi
177 <latexrelease> \fi
178 <latexrelease> \@noskipsecfalse
179 <latexrelease> \let \@refundefined \relax
180 <latexrelease> \let\AtBeginDocument\@firstofone
181 <latexrelease> \@begindocumenthook
182 <latexrelease> \ifdim\topskip<1sp\global\topskip 1sp\relax\fi
183 <latexrelease> \global\@maxdepth\maxdepth

```



```

184 <latexrelease> \global\let\@begindocumenthook\@undefined
185 <latexrelease> \ifx\@listfiles\@undefined
186 <latexrelease> \global\let\@filelist\relax
187 <latexrelease> \global\let\@addtofilelist\@gobble
188 <latexrelease> \fi
189 <latexrelease> \gdef\do##1{\global\let ##1\@notprerr}%
190 <latexrelease> \@preamblecmds
191 <latexrelease> \global\let \@nodocument \relax
192 <latexrelease> \global\let\do\noexpand
193 <latexrelease> \ignorespaces}
194 <latexrelease>\EndIncludeInRelease
195 <*2ekernel>
196 \@onlypreamble\document

```

(End definition for \document and others.)

\normalsfcodes The setting of \empty is just a flag. This command may be defined in a class or package file. If it is still \empty at \begin{document} it will be defined to be \frenchspacing or \nonfrenchspacing, depending on which of those appears to be in effect at that point.

```
197 \let\normalsfcodes\empty
```

(End definition for \normalsfcodes.)

\nofiles Set \fileswfalse which suppresses the places where L^AT_EX makes \immediate writes. The \makeindex and \makeglossary are disabled. \protected@write is redefined not to write to the file specified, but rather to write a blank line to the log file. This ensures that a *whatsit* node is still created, and so spacing is not affected by the \nofiles command; to ensure this more generally, the \if@nobreak test is needed.

```

198 \def\nofiles{%
199   \fileswfalse
200   \typeout{No auxiliary output files.^^J}%
201   \long\def\protected@write##1##2##3%
202     {\write\m@ne{}\if@nobreak\ifvmode\nobreak\fi\fi}%
203   \let\makeindex\relax
204   \let\makeglossary\relax}
205 \@onlypreamble\nofiles

```

(End definition for \nofiles.)

\protected@write This takes three arguments: an output stream, some initialization code, and some text to write. It then writes this, with appropriate handling of \protect and \thepage.

```

206 \long\def \protected@write#1#2#3{%
207   \begingroup
208   \let\thepage\relax
209   #2%
210   \let\protect\@unexpandable@protect
211   \edef\reserved@a{\write#1{#3}}%
212   \reserved@a
213   \endgroup
214   \if@nobreak\ifvmode\nobreak\fi\fi
215 }

```

(End definition for \protected@write.)

```
216 \let\@auxout=\@mainaux
```

`\include` In the definition of `\include`, `\def\reserved@b` changed to `\edef\reserved@b` to be consistent with the `\edef` in `\includeonly`. (Suggested by Rainer Schöpf & Frank Mittelbach. Change made 20 Jul 88.)

Changed definition of `\include` to allow space at end of file name — otherwise, typing `\include{foo }` would cause L^AT_EX to overwrite `foo.tex`. Change made 24 May 89, suggested by Rainer Schöpf and Frank Mittelbach

Made `\include` check for being used inside an `\include`'d file, as this will not work and cause surprising results.

```

217 </2ekernel>
218 <*2ekernel | latexrelease>
219 <latexrelease>\IncludeInRelease{2020/10/01}%
220 <latexrelease>                                {\includeonly}{Spaces in file names}%
221 \def\include#1{\relax
222   \ifnum\@auxout=\@partaux
223     \@latex@error{\string\include\space cannot be nested}\@eha
224   \else

```

Here the normalization will add `.tex` for all files, (it uses the same normalization as the hooks), so we need to remove that manually. `\@strip@tex@ext` does that.

```

225   \set@curr@file{#1}%
226   \edef\@curr@file{\@strip@tex@ext\@curr@file}%

```

For historical reasons `\@include` expects an argument delimited by a space. This is kept (though unnecessary now) to avoid errors in other packages that use `\@include` directly.

```

227   \expandafter\@include\expandafter{\@curr@file} % deliberate space
228   \fi}

```

Here in `\includeonly` we also need to strip `.tex` after normalization:

```

229 \def\includeonly#1{%
230   \@partswtrue

```

Because the argument to `\includeonly` is a comma-separated list of filenames where there may be comma's preceding some of the filenames or trailing them. Therefore we need to take the list apart, remove the unwanted spaces while leaving the spaces *in* the filenames intact.

```

231   \let\@partlist\@empty
232   \@for\reserved@a:=#1 \do
233   {%
234     \expandafter\set@curr@file\expandafter{\reserved@a}%
235     \ifx\@partlist\@empty
236       \edef\@partlist{\@strip@tex@ext\@curr@file}%
237     \else
238       \edef\@partlist{\@partlist,\@strip@tex@ext\@curr@file}%
239     \fi
240   }%
241 }
242 \@onlypreamble\includeonly

```

(End definition for `\include` and `\includeonly`.)

`\@strip@tex@ext` These macros take a (`\detokenized` file name and remove any `.tex` extension). Extra care is taken to not remove the string `.tex` from the middle of a file name: it is only removed if it's the very last thing in the file name.

```

243 \def\reserved@a#1{%
244 \def\@strip@tex@ext##1{%
245   \expandafter\@strip@tex@ext@aux
246     ##1\@nil\@nil
247     #1\@nil\relax\@nnil}
248 \def\@strip@tex@ext@aux##1#1\@nil##2\@nnil{%
249   \ifx\relax##2\@empty
250     \expandafter\@cdr\expandafter\@empty\@cdr{}\##1%
251     \else##1\fi}}%
252 \expandafter\reserved@a
253 \expandafter{\detokenize{.tex}}
254 </2ekernel | latexrelease>

(End definition for \@strip@tex@ext and \@strip@tex@ext@aux.)

255 <latexrelease>\EndIncludeInRelease
256 <latexrelease>\IncludeInRelease{2019/10/01}%
257 <latexrelease>      {\includeonly}{Spaces in file names}%
258 <latexrelease>
259 <latexrelease>\def\includeonly#1{%
260 <latexrelease>  \@partswtrue
261 <latexrelease>  \set@curr@file{\zap@space#1 \@empty}%
262 <latexrelease>  \let\@partlist\@curr@file
263 <latexrelease>  }
264 <latexrelease>
265 <latexrelease>\def\include#1{\relax
266 <latexrelease>  \ifnum\@auxout=\@partaux
267 <latexrelease>    \@latex@error{\string\include\space cannot be nested}\@eha
268 <latexrelease>  \else
269 <latexrelease>    \set@curr@file{#1 }%
270 <latexrelease>    \expandafter\@include\@curr@file
271 <latexrelease>  \fi}
272 <latexrelease>
273 <latexrelease>\let\@strip@tex@ext\@undefined
274 <latexrelease>\let\@strip@tex@ext@aux\@undefined
275 <latexrelease>
276 <latexrelease>\EndIncludeInRelease

277 <latexrelease>\IncludeInRelease{0000/00/00}%
278 <latexrelease>      {\includeonly}{Spaces in file names}%
279 <latexrelease>\def\includeonly#1{%
280 <latexrelease>  \@partswtrue
281 <latexrelease>  \edef\@partlist{\zap@space#1 \@empty}}
282 <latexrelease>
283 <latexrelease>\def\include#1{\relax
284 <latexrelease>  \ifnum\@auxout=\@partaux
285 <latexrelease>    \@latex@error{\string\include\space cannot be nested}\@eha
286 <latexrelease>  \else \@include#1 \fi}
287 <latexrelease>
288 <latexrelease>\EndIncludeInRelease
289 <*2ekernel>

\@include

290 </2ekernel>
291 <*2ekernel | latexrelease>

```

```

292 <latexrelease>\IncludeInRelease{2020/10/01}%
293 <latexrelease>{\@include}{Spaces in file names and hooks}%

294 \def\@include#1 {%
295   \clearpage
296   \if@filesw
297     \immediate\write\@mainaux{\string\@input{#1.aux}}%
298   \fi
299   \@tempswtrue
300   \if@partsw
301     \@tempswafalse
302     \edef\reserved@a{#1}%
303     \for\reserved@a:=\@partlist\do
304       {\ifx\reserved@a\reserved@b\@tempswtrue\fi}%
305   \fi
306   \if@tempswa
307     \let\@auxout\@partaux
308     \if@filesw
309       \immediate\openout\@partaux "#1.aux"
310       \immediate\write\@partaux{\relax}%
311     \fi

```

Now before going to the hooks we need to set \CurrentFile:

```

312 %-----
313   \@filehook@set@CurrentFile

```

Execute the before hooks just after we switched the .aux file ...

```

314   \UseHook{include/before}%
315   \UseHook{include/before/#1}%
316 %-----
317   \@input@{#1.tex}%
318 %-----

```

... then end hooks ...

```

319   \UseHook{include/end/#1}%
320   \UseHook{include/end}%
321 %-----
322   \clearpage
323 %-----

```

... and after the \clearpage the after hooks followed by another \clearpage just in case new material got added (after all we need to be in well defined state after the \include).

```

324   \UseHook{include/after/#1}%
325   \UseHook{include/after}%
326   \clearpage
327 %-----
328   \@writeckpt{#1}%
329   \if@filesw
330     \immediate\closeout\@partaux
331   \fi
332 \else

```

If the file is not included, reset \deadcycles, so that a long list of non-included files does not generate an ‘Output loop’ error.

```

333   \deadcycles\z@

```

```

334 \nameuse{cp@#1}%
335 \fi
336 \let\@auxout\@mainaux}
337 <latexrelease>\EndIncludeInRelease
338 </2ekernel | latexrelease>

339 <latexrelease>\IncludeInRelease{0000/00/00}%
340 <latexrelease> \{\@include\}{Spaces in file names}%
341 <latexrelease>\def\@include#1 {%
342 <latexrelease> \clearpage
343 <latexrelease> \if@filesw
344 <latexrelease> \immediate\write\@mainaux{\string\@input{#1.aux}}%
345 <latexrelease> \fi
346 <latexrelease> \@tempwatrue
347 <latexrelease> \if@partsw
348 <latexrelease> \@tempwafalse
349 <latexrelease> \edef\reserved@b{#1}%
350 <latexrelease> \@for\reserved@a:=\@partlist\do
351 <latexrelease> {\ifx\reserved@a\reserved@b\@tempwatrue\fi}%
352 <latexrelease> \fi
353 <latexrelease> \if@tempwa
354 <latexrelease> \let\@auxout\@partaux
355 <latexrelease> \if@filesw
356 <latexrelease> \immediate\openout\@partaux #1.aux
357 <latexrelease> \immediate\write\@partaux{\relax}%
358 <latexrelease> \fi
359 <latexrelease> \@input@{#1.tex}%
360 <latexrelease> \clearpage
361 <latexrelease> \@writeckpt{#1}%
362 <latexrelease> \if@filesw
363 <latexrelease> \immediate\closeout\@partaux
364 <latexrelease> \fi
365 <latexrelease> \else
366 <latexrelease> \deadcycles\z@
367 <latexrelease> \nameuse{cp@#1}%
368 <latexrelease> \fi
369 <latexrelease> \let\@auxout\@mainaux}
370 <latexrelease>
371 <latexrelease>\EndIncludeInRelease
372 <*2ekernel>

```

(End definition for \@include.)

\@writeckpt

```

373 \def\@writeckpt#1{%
374 \if@filesw
375 \immediate\write\@partaux{\string\@setckpt{#1}\@charlb}%
376 {\let\@elt\@wckptelt \cl@ckpt}%
377 \immediate\write\@partaux{\@charrb}%
378 \fi}

```

(End definition for \@writeckpt.)

\@wckptelt

```

379 \def\@wckptelt#1{%

```

```

380 \immediate\write\@partaux{%
381 \string\setcounter{#1}{\the\@nameuse{c@#1}}}}

```

(End definition for \wckptelt.)

```

\@setckpt RmS 93/08/31: introduced \@setckpt
382 \def\@setckpt#1{\global\@namedef{cp@#1}}

```

(End definition for \@setckpt.)

```

\@charlb The following defines \@charlb and \@charrb to be { and }, respectively with \catcode
\@charrb 11.

```

```

383 {\catcode'\@charlb=1 \catcode'\@charrb=2
384 \catcode'\@charlb=11 \catcode'\@charrb=11
385 \gdef\@charlb[{
386 \gdef\@charrb[}]
387 ]% }brace matching

```

(End definition for \@charlb and \@charrb.)

1.1 Safe Input Macros

\@curr@file File name handling is done by generating a csname from the provided file name (which means that UTF-8 octets gets turned into strings as this is what happens if they appear in a csname due to the code in `utf8.def`). By setting `\escapechar` to -1 we ensure that we don't get a backslash in front. As a result we end up with all characters as catcode 12 (plus spaces). We then sometimes add quotes around the construct (removing any existing inner quotes. Sometimes we only remove the quotes if they have been supplied by the user. There is clearly some room for improvement.

A side effect of the new code is that we will see quotes around file name displays where there haven't been any before.

For compatibility with existing code using `{abc}.tex` or `{one.two}.png`, an initial brace group is discarded before expansion and `\string` is applied. The content of the brace group is discarded. This means that a leading space will be lost unless protected (by `{ }` or `" "` or `\space`) but filenames with a space are hopefully rare.

The definition below is from 2019 and only used during kernel bootstrapping, later on in `ltfilehook.dtx` it will get overwritten.

```

388 \def\set@curr@file#1{%
389 \begingroup
390 \escapechar\m@ne
391 \xdef\@curr@file{%
392 \expandafter\expandafter\expandafter\unquote@name
393 \expandafter\expandafter\expandafter{%
394 \expandafter\string
395 \csname\@firstofone#1\@empty\endcsname}}%
396 \endgroup
397 }

```

(End definition for \@curr@file and \set@curr@file.)

```

\quote@name Quoting spaces
\quote@@name
\unquote@name

```

```

a b c      -> "a b c"
"a b c"    -> "a b c"
a" "b" "c -> "a b c"
           -> ""

398 </2ekernel>
399 <*2ekernel | latexrelease>
400 <latexrelease>\IncludeInRelease{2019/10/01}%
401 <latexrelease>          {\quote@name}{Quote file names}%
402 \def\quote@name#1{"\quote@@name#1@gobble"}
403 \def\quote@@name#1"#1\quote@@name}

```

and removing quotes ...

```

404 \def\unquote@name#1{\quote@@name#1@gobble}

(End definition for \quote@name, \quote@@name, and \unquote@name.)

```

\IfFileExists

```

405 \DeclareRobustCommand\IfFileExists[1]{%
406   \set@curr@file{#1}%
407   \expandafter\IfFileExists@\expandafter{\@curr@file}}

(End definition for \IfFileExists.)

408 </2ekernel | latexrelease>
409 <latexrelease>\EndIncludeInRelease
410 <latexrelease>\IncludeInRelease{0000/00/00}%
411 <latexrelease>          {\quote@name}{Quote file names}%
412 <latexrelease>
413 <latexrelease>\let\quote@name\@undefined
414 <latexrelease>\let\quote@@name\@undefined
415 <latexrelease>\let\unquote@name\@undefined
416 <latexrelease>
417 <latexrelease>\long\def \IfFileExists#1#2#3{%
418 <latexrelease>  \openin\@inputcheck#1 %
419 <latexrelease>  \ifeof\@inputcheck
420 <latexrelease>    \ifx\input@path\@undefined
421 <latexrelease>    \def\reserved@a{#3}%
422 <latexrelease>  \else
423 <latexrelease>    \def\reserved@a{\@iffilenamepath{#1}{#2}{#3}}%
424 <latexrelease>  \fi
425 <latexrelease> \else
426 <latexrelease>  \closein\@inputcheck
427 <latexrelease>  \edef\@filef@und{#1}%
428 <latexrelease>  \def\reserved@a{#2}%
429 <latexrelease>  \fi
430 <latexrelease>  \reserved@a}
431 <latexrelease>
432 <latexrelease>\EndIncludeInRelease
433 <*2ekernel>

```

\IfFileExists@ Argument #1 is \@curr@file so catcode 12 string with no quotes.

The original definition picked up arguments #2 and #3 in a way that they couldn't contain unbalanced conditionals. A better implementation would have been not to pick up the arguments at all but instead use the usual \@firstoftwo and \@secondoftwo.

However, that changes how # is interpreted and so we can't do that noways without invalidating a lot of code. Therefore the somewhat curious construction near the end.

```

434 </2ekernel>
435 <*2ekernel | latexrelease>
436 <latexrelease>\IncludeInRelease{2021/06/01}%
437 <latexrelease>{\IfFileExists@}{manage unbalanced conditionals}
438 \long\def \IfFileExists@#1#2#3{%
439   \openin\@inputcheck"#1" %
440   \ifeof\@inputcheck
441     \ifx\input@path\@undefined
442       \let\reserved@a\@secondoftwo
443     \else
444       \def\reserved@a{\@iffileonpath{#1}}%
445     \fi
446   \else
447     \closein\@inputcheck
448     \edef\@filef@und{"#1" }%
449     \let\reserved@a\@firstoftwo
450   \fi

```

This is just there so that any # inside #2 or #3 needs doubling (as that was the case in the past).

```

451   \expandafter\def\expandafter\reserved@a
452     \expandafter{\reserved@a{#2}{#3}}%
453 \reserved@a}
454 </2ekernel | latexrelease>
455 <latexrelease>\EndIncludeInRelease
456 <latexrelease>\IncludeInRelease{2019/10/01}%
457 <latexrelease>{\IfFileExists@}{manage unbalanced conditionals}
458 <latexrelease>
459 <latexrelease>\long\def \IfFileExists@#1#2#3{%
460 <latexrelease> \openin\@inputcheck"#1" %
461 <latexrelease> \ifeof\@inputcheck
462 <latexrelease> \ifx\input@path\@undefined
463 <latexrelease> \def\reserved@a{#3}%
464 <latexrelease> \else
465 <latexrelease> \def\reserved@a{\@iffileonpath{#1}{#2}{#3}}%
466 <latexrelease> \fi
467 <latexrelease> \else
468 <latexrelease> \closein\@inputcheck
469 <latexrelease> \edef\@filef@und{"#1" }%
470 <latexrelease> \def\reserved@a{#2}%
471 <latexrelease> \fi
472 <latexrelease> \reserved@a}
473 <latexrelease>\EndIncludeInRelease
474 <latexrelease>\IncludeInRelease{0000/00/00}%
475 <latexrelease>{\IfFileExists@}{manage unbalanced conditionals}
476 <latexrelease>
477 <latexrelease>\let\IfFileExists@\@undefined
478 <latexrelease>
479 <latexrelease>
480 <latexrelease>\EndIncludeInRelease
481 <*2ekernel>

```

(End definition for \IfFileExists@.)

`\@iffileonpath` If the file is not found by `\openin`, and `\input@path` is defined, look in all the directories specified in `\input@path`.

```

482 </2ekernel>
483 <*2ekernel | latexrelease>
484 <latexrelease> \IncludeInRelease{2019/10/01}%
485 <latexrelease>           {\@iffileonpath}{Quote file names}
486 \long\def\@iffileonpath#1{%
487   \let\reserved@a\@secondoftwo
488   \expandafter\@tfor\expandafter\reserved@b\expandafter
489     :\expandafter=\input@path\do{%
490     \openin\@inputcheck\expandafter\quote@name\expandafter{\reserved@b#1} %
491     \ifeof\@inputcheck\else
492       \edef\@filef@und{\expandafter\quote@name\expandafter{\reserved@b#1} }%
493       \let\reserved@a\@firstoftwo%
494       \closein\@inputcheck
495       \@break@tfor
496       \fi}%
497   \reserved@a}

```

(End definition for \@iffileonpath.)

```

498 </2ekernel | latexrelease>
499 <latexrelease> \EndIncludeInRelease
500 <latexrelease> \IncludeInRelease{0000/00/00}%
501 <latexrelease>           {\quote@name}{Quote file names}
502 <latexrelease>
503 <latexrelease> \long\def\@iffileonpath#1{%
504 <latexrelease>   \let\reserved@a\@secondoftwo
505 <latexrelease>   \expandafter\@tfor\expandafter\reserved@b\expandafter
506 <latexrelease>     :\expandafter=\input@path\do{%
507 <latexrelease>     \openin\@inputcheck\reserved@b#1 %
508 <latexrelease>     \ifeof\@inputcheck\else
509 <latexrelease>       \edef\@filef@und{\reserved@b#1 }%
510 <latexrelease>       \let\reserved@a\@firstoftwo%
511 <latexrelease>       \closein\@inputcheck
512 <latexrelease>       \@break@tfor
513 <latexrelease>       \fi}%
514 <latexrelease>   \reserved@a}
515 <latexrelease>
516 <latexrelease> \EndIncludeInRelease
517 <*2ekernel>

```

`\InputIfFileExists` Now define `\InputIfFileExists` to input #1 if it seems to exist. Immediately prior to the input, #2 is executed. If the file #1 does not exist, execute ‘#3’.

This here is a temporary definition for the kernel. The real one comes somewhat later in the file `ltxfilehook.dtx`.

```

518 \DeclareRobustCommand \InputIfFileExists[2]{%
519   \IfFileExists{#1}%
520   {%
521     \expandafter\@swaptwoargs\expandafter
522     {\@filef@und}{#2\@addtofilelist{#1}\@input}}

```

(End definition for \InputIfFileExists.)

`\@swaptwoargs` Swap two arguments and return them unbraced (like `\@firstoftwo` etc).

```

523 </2ekernel>
524 <*2ekernel | latexrelease>
525 <latexrelease> \IncludeInRelease{2019/10/01}%
526 <latexrelease> {\@swaptwoargs}{Don't lose the file name}%
527 \long\def\@swaptwoargs#1#2{#2#1}

528 </2ekernel | latexrelease>
529 <latexrelease> \EndIncludeInRelease
530 <latexrelease> \IncludeInRelease{0000/00/00}%
531 <latexrelease> {\@swaptwoargs}{Don't lose the file name}%
532 <latexrelease> \let\@swaptwoargs\undefined
533 <latexrelease> \EndIncludeInRelease
534 <*2ekernel>

```

(End definition for `\@swaptwoargs`.)

`\input` Input a file: if the argument is given in braces use safe input macros, otherwise use TeX's primitive `\input` command (which is called `\@@input` in L^AT_EX).

```

535 \def\input{\@ifnextchar\bggroup\iinput\@@input}

```

(End definition for `\input`.)

`\iinput` Define `\@iinput` (i.e., `\input`) in terms of `\InputIfFileExists`.
Changes to `\iinput`: adapt to the changes to `\@missingfileerror`.

```

536 </2ekernel>
537 <*2ekernel | latexrelease>
538 <latexrelease> \IncludeInRelease{2020/10/01}%
539 <latexrelease> {\@iinput}{Change in file error handling}%
540 \def\@iinput#1{%
541   \InputIfFileExists{#1}{}%
542   {\filename@parse\@curr@file
543     \edef\reserved@a{\noexpand\@missingfileerror
544       {\filename@area\filename@base}%
545       {\ifx\filename@ext\relax tex\else\filename@ext\fi}}%

```

This line now just sets `\@missingfile@<part>`:

```

546   \reserved@a

```

Now here we have to use it. The file here is guaranteed to exist, because `\@missingfileerror` ensures so, but we have to use `\InputIfFileExists` because it executes the file hooks.

```

547   \edef\reserved@a{\noexpand\@iinput{%
548     \@missingfile@area\@missingfile@base.\@missingfile@ext}}%
549   \reserved@a}}
550 </2ekernel | latexrelease>
551 <latexrelease> \EndIncludeInRelease

552 <latexrelease> \IncludeInRelease{2019/10/01}%
553 <latexrelease> {\@iinput}{Quote file names}%
554 <latexrelease>
555 <latexrelease> \def\@iinput#1{%
556 <latexrelease>   \InputIfFileExists{#1}{}%
557 <latexrelease>   {\filename@parse\@curr@file
558 <latexrelease>     \edef\reserved@a{\noexpand\@missingfileerror
559 <latexrelease>       {\filename@area\filename@base}%
560 <latexrelease>       {\ifx\filename@ext\relax tex\else\filename@ext\fi}}%

```

```

561 <latexrelease> \reserved@a}}
562 <latexrelease>\EndIncludeInRelease
563 <latexrelease>\IncludeInRelease{0000/00/00}%
564 <latexrelease> {\@input}{Quote file names}%
565 <latexrelease>\def\@input#1{%
566 <latexrelease> \InputIfFileExists{#1}{}%
567 <latexrelease> {\filename@parse{#1}%
568 <latexrelease> \edef\reserved@a{\noexpand\@missingfileerror
569 <latexrelease> {\filename@area\filename@base}%
570 <latexrelease> {\ifx\filename@ext\relax tex\else\filename@ext\fi}}%
571 <latexrelease> \reserved@a}}
572 <latexrelease>\EndIncludeInRelease
573 <*2ekernel>

```

(End definition for \@input.)

\@input Define \@input in terms of \IfFileExists. So this is a ‘safe input’ command, but the files input are not listed by \listfiles.

We don’t want .aux, .toc files etc be listed by \listfiles. However, something like .bbl probably should be listed and thus should be implemented not by \@input.

```

574 \def\@input#1{%
575 \IfFileExists{#1}{\@input\@filef@und}{\typeout{No file #1.}}}

```

(End definition for \@input.)

\@input@ Version of \@input that does add the file to \@filelist.

```

576 \def\@input@#1{\InputIfFileExists{#1}{\typeout{No file #1.}}}

```

(End definition for \@input@.)

\@missingfileerror This ‘error’ command avoids T_EX’s primitive missing file loop.

Missing file error. Prompt for a new filename, offering a default extension.

Changes to \@missingfileerror: rather than trying to input the file by force, now \@missingfileerror just returns three \@missingfile@{part} and the caller macro is responsible for doing the right thing with it.

```

577 </2ekernel>
578 <*2ekernel | latexrelease>
579 <latexrelease>\IncludeInRelease{2020/10/01}%
580 <latexrelease> {\@missingfileerror}{Do not load missing file immediately}%
581 \gdef\@missingfileerror#1#2{%
582 \typeout{^^J! LaTeX Error: File ‘#1.#2’ not found.^^J^^J%
583 Type X to quit or <RETURN> to proceed,^^J%
584 or enter new name. (Default extension: #2)^^J}%
585 \message{Enter file name: }%
586 {\endlinechar\m@ne
587 \global\read\m@ne to\@gtempa}%
588 \ifx\@gtempa\@empty

```

If the user answers with <return>, fallback to the .tex file (previously it did nothing).

```

589 \let\@missingfile@area\@empty
590 \let\@missingfile@base\@empty
591 \def\@missingfile@ext{tex}%
592 \else

```

Use `\batchmode\read-1 to <tl>` to end the T_EX run, same as `expl3` does (it was `\batchmode\@@end before`).

```

593 \def\reserved@b{\batchmode\read-1 to \reserved@a}%
594 \def\reserved@a{x}\ifx\reserved@a\@gtempa\reserved@b\fi
595 \def\reserved@a{X}\ifx\reserved@a\@gtempa\reserved@b\fi
596 \filename@parse\@gtempa
597 \edef\filename@ext{%
598 \ifx\filename@ext\relax#2\else\filename@ext\fi}%
599 \edef\reserved@a{%

```

Only check `\IfFileExists` (it was `\InputIfFileExists`).

```

600 \noexpand\IfFileExists
601 {\filename@area\filename@base.\filename@ext}%

```

If the file exists, define `\@missingfile@<part>`.

```

602 {\def\noexpand\@missingfile@area{\filename@area}%
603 \def\noexpand\@missingfile@base{\filename@base}%
604 \def\noexpand\@missingfile@ext {\filename@ext}}%
605 {\noexpand\@missingfileerror
606 {\filename@area\filename@base}{\filename@ext}}}%
607 \reserved@a
608 \fi
609 }
610 </2ekernel | latexrelease>
611 <latexrelease>\EndIncludeInRelease
612 <latexrelease>\IncludeInRelease{0000/00/00}%
613 <latexrelease> {\@missingfileerror}{Do not load missing file immediately}%
614 <latexrelease>
615 <latexrelease>\gdef\@missingfileerror#1#2{%
616 <latexrelease> \typeout{^^J! LaTeX Error: File ‘#1.#2’ not found.^^J^^J%
617 <latexrelease> Type X to quit or <RETURN> to proceed,^^J%
618 <latexrelease> or enter new name. (Default extension: #2)^^J}%
619 <latexrelease> \message{Enter file name: }%
620 <latexrelease> {\endlinechar\m@ne
621 <latexrelease> \global\read\m@ne to\@gtempa}%
622 <latexrelease> \ifx\@gtempa\@empty
623 <latexrelease> \else
624 <latexrelease> \def\reserved@a{x}\ifx\reserved@a\@gtempa\batchmode\@@end\fi
625 <latexrelease> \def\reserved@a{X}\ifx\reserved@a\@gtempa\batchmode\@@end\fi
626 <latexrelease> \filename@parse\@gtempa
627 <latexrelease> \edef\filename@ext{%
628 <latexrelease> \ifx\filename@ext\relax#2\else\filename@ext\fi}%
629 <latexrelease> \edef\reserved@a{%
630 <latexrelease> \noexpand\InputIfFileExists
631 <latexrelease> {\filename@area\filename@base.\filename@ext}%
632 <latexrelease> {}%
633 <latexrelease> {\noexpand\@missingfileerror
634 <latexrelease> {\filename@area\filename@base}{\filename@ext}}}%
635 <latexrelease> \reserved@a
636 <latexrelease> \fi}
637 <latexrelease>
638 <latexrelease>\EndIncludeInRelease
639 <*2ekernel>

```

(End definition for \@missingfileerror.)

`\@obsoletefile` For compatibility with L^AT_EX 2.09 document styles, we distribute files called `article.sty`, `book.sty`, `report.sty`, `slides.sty` and `letter.sty`. These use the command `\@obsoletefile`, which produces a warning message.

```
640 \def\@obsoletefile#1#2{%
641   \@latex@warning@no@line{inputting ‘#1’ instead of obsolete ‘#2’}}
642 \@onlypreamble\@obsoletefile
```

1.2 Listing files

A list of files input so far. The initial value of `\@gobble` eats the comma before the first file name.

```
\@filelist
643 \let\@filelist\@gobble
```

Add to the list of files input so far. This ‘real’ definition is only used for ‘cfg’ files during initex. An initial definition of `\@gobble` has already been set.

```
\@addtofilelist
644 %\def\@addtofilelist#1{\xdef\@filelist{\@filelist,#1}}
```

A preamble command to cause `\end{document}` to list files input from the main file.

```
\listfiles
645 \def\listfiles{%
646   \let\listfiles\relax
647   \def\@listfiles##1##2##3##4##5##6##7##8##9\@@{%
648     \def\reserved@d{\}%
649     \@tfor\reserved@c:=##1##2##3##4##5##6##7##8\do{%
650       \ifx\reserved@c\reserved@d
651         \edef\filename@area{ \filename@area}%
652       \fi}}%

653   \def\@dofilelist{%
654     \typeout{^^J *File List*}%
655     \@for\@currname:=\@filelist\do{%
656       \filename@parse\@currname
657       \edef\reserved@a{%
658         \filename@base.%
659         \ifx\filename@ext\relax tex\else\filename@ext\fi}%
660       \expandafter\let\expandafter\reserved@b
661         \csname ver@\reserved@a\endcsname
662       \expandafter\expandafter\expandafter\@listfiles\expandafter
663         \filename@area\filename@base\@@\@@\@@\@@\@@\@@\@@\@@\@@\@@
664       \typeout{%
665         \filename@area\reserved@a
666         \ifx\reserved@b\relax\else\@spaces\reserved@b\fi}}%
667     \typeout{ *****^^J}}
```

The `\@filelist` will be de-activated if `\listfiles` does not appear in the preamble. `\begin{document}` contains code equivalent to the following:

```
\AtBeginDocument{%
  \ifx\@listfiles\@undefined
    \let\@filelist\relax
    \let\@addtofilelist\@gobble
  \fi}

668 \@onlypreamble\listfiles
```

```

\@dofilelist 669 \let\@dofilelist\relax
670 \</2kernel>
(End definition for \@obsoletefile and others.)

```

File r

ltoutenc.dtx

1 Font encodings

This section of the kernel contains commands for declaring encoding-specific commands, such as accents. It also contains the code for some of the encoding files, including `omlenc.def`, `omsenc.def`, `t1enc.def` and `ot1enc.def` files, which define the OLM, OMS, T1 and OT1 encodings, and the `fontenc` package for selecting encodings.

The `fontenc` package has options for encodings, of which the last option is the default encoding. For example, to use the OT2, OT3 and T1 encodings, with T1 as the default, you say:

```
\usepackage[OT2,OT3,T1]{fontenc}
```

The standard kernel set-up loads font encoding files and selects an encoding as follows.

```
\input {omlenc.def}
\input {t1enc.def}
\input {ot1enc.def}
\input {omsenc.def}
\fontencoding{OT1}
```

Note that the files in the standard `inputenc` package depend on this behaviour of the kernel.

The syntax for declaring encoding-specific commands is:

```
\DeclareTextCommand{<command>}{<encoding>}
                               [<number>] [<default>]{<commands>}
```

This command is like `\newcommand`, except that it defines a command which is specific to one encoding. The resulting command is always robust, even if its definition is fragile. For example, the definition of `\l` in the OT1 encoding is:

```
\DeclareTextCommand{\l}{OT1}{\@xxxii l}
```

`\DeclareTextCommand` takes the same optional arguments as `\newcommand`.

```
\ProvideTextCommand{<command>}{<encoding>}
                               [<number>] [<default>]{<commands>}
```

This acts like `\DeclareTextCommand`, but does nothing if the command is already defined.

```
\DeclareTextSymbol{<command>}{<encoding>}{<slot>}
```

This command defines a text symbol, with a particular slot in that encoding. The commands:

```
\DeclareTextSymbol{\ss}{OT1}{25}
\DeclareTextCommand{\ss}{OT1}{\char25 }
```

have the same effect, but the `\DeclareTextSymbol` is faster.

```
\DeclareTextAccent{<command>}{<encoding>}{<slot>}
```

This command declares a text accent. The commands:

```
\DeclareTextAccent{"}{OT1}{127}
\DeclareTextCommand{"}{OT1}{\add@accent {127}}
```

have the same effect.

```
\DeclareTextComposite{<command>}
                               {<encoding>}{<argument>}{<slot>}
```

This command declares a composite letter, for example in the T1 encoding `\'a` is slot 225, which is declared by:

```
\DeclareTextComposite{\'}{T1}{a}{225}
```

The *command* will normally have been declared with `\DeclareTextAccent`, or as a one-argument `\DeclareTextCommand`.

`\DeclareTextComposite` is the most common example of using the more general declaration `\DeclareTextCompositeCommand`, which can define a composite to be an arbitrary piece of text.

```
\DeclareTextCompositeCommand{<command>}
                               {<encoding>}{<argument>}{<text>}
```

For example, in the OT1 encoding Å has a hand-crafted definition this is declared as follows

```
\DeclareTextCompositeCommand{\r}{OT1}{A}
{\leavevmode\setbox\z@\hbox{!}\dimen@ht\z@\advance\dimen@-1ex%
 \rlap{\raise.67\dimen@\hbox{\char23}}A}
```

The *command* will normally have been declared with `\DeclareTextAccent`, or as a one-argument `\DeclareTextCommand`.

The commands defined using the above declarations can be used in two ways. Normally they are used by just calling the command in the appropriate encoding, for example `\ss`. However, sometimes you may wish to use a command in an encoding where it is not defined. If the command has no arguments, then you can use it in another encoding by calling `\UseTextSymbol`:

```
\UseTextSymbol{<encoding>}{<command>}
```

For example, `\UseTextSymbol{OT1}{\ss}` has the same effect as:

```
{\fontencoding{OT1}\selectfont\ss}
```

If the command has one argument then you can use it in another encoding by calling `\UseTextAccent`:

```
\UseTextAccent{<encoding>}{<command>}{<text>}
```

For example, if the current encoding is OT2 then `\UseTextAccent{OT1}{\'}{a}` has the same effect as:

```
{\fontencoding{OT1}\selectfont\'{\fontencoding{OT2}\selectfont a}}
```


You can also declare a default definition for a text command, which will be used if the current encoding has no appropriate definition. Such use will also set the definition for this command in the current encoding to equal this default definition; this makes subsequent uses of the command much faster.

```
\DeclareTextCommandDefault{<command>}{<definition>}
```

For example, the default definition of the command `\textonequarter` (which produces the fraction $\frac{1}{4}$) could be built using math mode:

```
\DeclareTextCommandDefault{\textonequarter}{\ensuremath {\frac{1}{4}}}
```

There is a matching `\Provide` command which will not override an existing default definition:

```
\ProvideTextCommandDefault{<command>}{<definition>}
```

The most common use for these commands is to use symbols from other encodings, so there are some optimizations provided:

```
\DeclareTextSymbolDefault{<command>}{<encoding>}
\DeclareTextAccentDefault{<command>}{<encoding>}
```

are short for:

```
\DeclareTextCommandDefault{<command>}
      {\UseTextSymbol{<encoding>}{<command>}}
\DeclareTextCommandDefault[1]{<command>}
      {\UseTextAccent{<encoding>}{<command>}{#1}}
```

For example, to make OT1 the default encoding for `\ss` and `\'` you say:

```
\DeclareTextSymbolDefault{\ss}{OT1}
\DeclareTextAccentDefault{\'}{OT1}
```

Note that you can use these commands on any zero- or one-argument commands declared with `\DeclareText*` or `\ProvideText*`, not just those defined using `\DeclareTextSymbol` or `\DeclareTextAccent`.

1.1 Removing encoding-specific commands

In some cases encoding definitions are given to provide some limited support since nothing better is available, for example, the definition for `\textdollar` in OT1 is a hack since \$ and £ actually share the same slot in this encoding. Thus if such a glyph becomes available in a different encoding (e.g., TS1) one would like to get rid of the flaky one and make the default definition point to the new encoding. In such a case defining

```
\DeclareTextSymbol{\textdollar}{TS1}{36}
\DeclareTextSymbolDefault{\textdollar}{TS1}
```

is not enough since if typesetting in OT1 L^AT_EX will still find the encoding specific-definition for OT1 and therefore ignore the new default. Therefore to ensure that in this case the TS1 version is used we have to remove the OT1 declaration:

```
\UndeclareTextCommand{\textdollar}{OT1}
```

Since the \$ sign is a proper glyph in the T1 encoding there is no point removing its definition and forcing L^AT_EX to pick up the TS1 version if typesetting in this encoding. However, assume you want to use the variant dollar sign, i.e., \$ for your dollars. In that case you have to get rid of the T1 declaration as well, e.g., the following would do that for you:

```
\UndeclareTextCommand{\textdollar}{OT1}
\UndeclareTextCommand{\textdollar}{T1}
\DeclareTextCommandDefault{\textdollar}
{\UseTextSymbol{TS1}\textdollaroldstyle}
```

1.2 The order of declarations

If an encoding-specific command is defined for more than one encoding, then it will execute fastest in the encoding in which it was defined last since its top-level definition will be set up to execute in that encoding without any overhead.

For this reason the file `fonttext.ltx` currently first loads the definitions for the T1 encoding and then those for the OT1 encoding so that typesetting in OT1 is optimized since that is (still) the default. However, when T1 is explicitly requested (via `\usepackage[T1]{fontenc}`) the top-level definitions are automatically changed to favour T1 since its declarations are reloaded in the process.

For the same reason default declarations should never come last since they are implemented as a special encoding themselves (with the name ?). Specifying them last would simply mean to make those encoding-specific commands equally inefficient in all encodings. Therefore the `textcomp` package, for example, first sets up all defaults to point to TS1 and then declares the commands in the TS1 encoding.

1.3 Docstrip modules

This `.dtx` file is be used to generate several related files containing font encoding definitions. The mutually exclusive docstrip options are listed here.

T1	generates <code>t1enc.def</code> for the Cork encoding.
TS1	generates <code>ts1enc.def</code> for the Text Companion encoding.
TS1sty	generates <code>textcomp.sty</code> , package that sets up use of the Text Companion encoding.
OT1	generates <code>ot1enc.def</code> for Knuth's CM encoding.
OMS	generates <code>omsenc.def</code> for Knuth's math symbol encoding.
OML	generates <code>omlenc.def</code> for Knuth's math letters encoding.
OT4	generates <code>ot4enc.def</code> for the Polish extension to the OT1 encoding, created by B. Jackowski and M. Ryćko for use with the Polish version of Computer Modern and Computer Concrete.
TU	generates <code>tuenc.def</code> for Unicode font encoding.
package	generates <code>fontenc.sty</code> for selecting encodings.
2ekernel	for the kernel commands.

1.4 Definitions for the kernel

1.4.1 Declaration commands

This section contains definitions for commands such as accents which depend on the current encoding. These commands will usually be kept in `.def` files, for example `ot1enc.def` contains the definitions for the OT1 encoding.

```
1 <*2ekernel>
2 \message{font encodings,}

Far too many macros in one block here!
```

```
\DeclareTextCommand
\ProvideTextCommand
\DeclareTextSymbol
  \@dec@text@cmd
\chardef@text@cmd
  \@changed@cmd
  \@changed@x
\TextSymbolUnavailable
  \@inmathwarn

If you say:
  \DeclareTextCommand{\foo}{T1}...

then \foo is defined to be \T1-cmd \foo \T1\foo, where \T1\foo is one control sequence, not two! We then call \newcommand to define \T1\foo.

3 \def\DeclareTextCommand{%
4   \@dec@text@cmd\newcommand}
5 \def\ProvideTextCommand{%
6   \@dec@text@cmd\providecommand}
7 \def\@dec@text@cmd#1#2#3{%
8   \expandafter\def\expandafter#2%
9     \expandafter{%
10      \csname#3-cmd\expandafter\endcsname
11      \expandafter#2%
12      \csname#3\string#2\endcsname
13    }%
14   \let\@ifdefinable\@rc@ifdefinable
15   \expandafter#1\csname#3\string#2\endcsname}
```

This command was introduced to fix a major bug in `\@dec@text@cmd` without changing that command itself. This was thought to be necessary because it is defined in more than one package. (Perhaps the more serious bug is to put complex low-level commands like this in packages?)

The problem it solves is that whereas both `\newcommand` and `\providecommand` (used just above) both handle the resetting of `\@ifdefinable` (following its disabling in `\@dec@text@cmd`), the primitive `\chardef` neither needs the disabling, nor does the resetting.

```
16 \def\chardef@text@cmd{%
17   \let\@ifdefinable\@@ifdefinable
18   \chardef
19 }
20 \def\DeclareTextSymbol#1#2#3{%
21   \@dec@text@cmd\chardef@text@cmd#1{#2}#3\relax
22 }
```

The declarations are only available before `\begin{document}`.

```
23 \@onlypreamble\DeclareTextCommand
24 \@onlypreamble\DeclareTextSymbol
```

The sneaky bit in all this is what `\T1-cmd \foo \T1\foo` does. There are five possibilities, depending on the current values of `\protect`, `\cf@encoding` and `\ifmmode`:

- If `\protect` is `\@typeset@protect` and `\cf@encoding` is T1, then we execute `\T1\foo`. This should be the normal behaviour, and is optimized for speed.
- If `\protect` is `\@typeset@protect`, `\cf@encoding` is (say) OT1, and `\OT1\foo` is defined, then we execute `\OT1\foo`.
- If `\protect` is `\@typeset@protect`, `\cf@encoding` is (say) OT1, we're in text mode, and `\OT1\foo` is undefined, then we define `\OT1\foo` to be the default value of `\foo`, and execute `\OT1\foo`.
- If `\protect` is `\@typeset@protect`, `\cf@encoding` is (say) OT1, we're in math mode, and `\OT1\foo` is undefined, then we execute the default value of `\foo`. (This is necessary so that things like `X_\copyright` work properly.)
- If `\protect` is not `\@typeset@protect` then we execute `\noexpand\foo`. For example, if we are writing to a file, then this results in `\foo` being written. If we are in a `\mark`, then `\foo` will be put in the mark—since `\foo` is robust, it will then survive all the things which may happen to it whilst it's a `\mark`.

So after all that, we will either execute the appropriate definition of `\foo` for the current encoding, or we will execute `\noexpand\foo`.

The default value of `\foo` is `\?\foo` if it is defined, and an error message otherwise.

When the encoding is changed from T1 to OT1, `\T1-cmd` is defined to be `\@changed@cmd` and `\OT1-cmd` is defined to be `\@current@cmd`. This means that the test for what the current encoding is can be performed quickly.

```

25 \def\@current@cmd#1{%
26     \ifx\protect\@typeset@protect
27         \@inmathwarn#1%
28     \else
29         \noexpand#1\expandafter\@gobble
30     \fi}

31 \def\@changed@cmd#1#2{%
32     \ifx\protect\@typeset@protect
33         \@inmathwarn#1%
34         \expandafter\ifx\csname\cf@encoding\string#1\endcsname\relax
35             \expandafter\ifx\csname ?\string#1\endcsname\relax
36                 \expandafter\def\csname ?\string#1\endcsname{%
37                     \TextSymbolUnavailable#1%
38                 }%
39             \fi
40             \global\expandafter\let
41                 \csname\cf@encoding\string#1\expandafter\endcsname
42                 \csname ?\string#1\endcsname
43             \fi
44             \csname\cf@encoding\string#1%
45                 \expandafter\endcsname
46     \else
47         \noexpand#1%
48     \fi}

49 \gdef\TextSymbolUnavailable#1{%
50     \@latex@error{%
51         Command \protect#1 unavailable in encoding \cf@encoding%
52     }\@eha}

```

The command `\@inmathwarn` produces a warning message if we are currently in math mode. Note that since this command is used inside text commands, it can't call `\relax` before the `\ifmmode`. This means that it is possible for the warning to fail to be issued at the beginning of a row of an `halign` whose template enters math mode. This is probably a bad feature, but there's not much that can be done about it, since adding a `\relax` would break ligatures and kerning between text symbols.

A more efficient solution would be to make `\@inmathwarn` and `\@inmatherr` equal to `\@empty` and `\relax` by default, and to have `\everymath` reset them to their usual definitions. This is left for future investigation (for example it may break some third party code).

```
53 \def\@inmathwarn#1{%
54   \ifmmode
55     \@latex@warning{Command \protect#1 invalid in math mode}%
56   \fi}
```

(End definition for \DeclareTextCommand and others.)

`\DeclareTextCommandDefault`
`\ProvideTextCommandDefault`

These define commands with encoding ?.

Note that `\DeclareTextCommandDefault` can only be used in the preamble, but that the `\Provide` version is allowed in inputenc .def files, so is allowed anywhere.

```
57 \def\DeclareTextCommandDefault#1{%
58   \DeclareTextCommand#1?}

59 \def\ProvideTextCommandDefault#1{%
60   \ProvideTextCommand#1?}

61 \@onlypreamble\DeclareTextCommandDefault
62 %\@onlypreamble\ProvideTextCommandDefault
```

They require `\? -cmd` to be initialized as `\@changed@cmd`.

```
63 \expandafter\let\csname?-cmd\endcsname\@changed@cmd
```

(End definition for \DeclareTextCommandDefault and \ProvideTextCommandDefault.)

`\DeclareTextAccent`

This is just a disguise for defining a T_EX `\accent` command.

```
64 \def\DeclareTextAccent#1#2#3{%
65   \DeclareTextCommand#1{#2}{\add@accent{#3}}

66 \@onlypreamble\DeclareTextAccent
```

(End definition for \DeclareTextAccent.)

`\add@accent`

To save space this code is shared between all text accents that are set using the `\accent` primitive. The argument is pre-set in a box so that any font loading that is needed is already done within the box. This is needed because font-loading involves grouping and that would prevent the accent mechanism from working so that the accent would not be positioned over the argument. Declarations that change the font should be allowed (only low-level ones are at present) inside the argument of an accent command, but not size changes, as they involve `\setbox` operations which also inhibit the mechanism of the `\accent` primitive.

Note that the whole process is within a group. For a detailed discussion of this reimplementaion and its deficiencies, see pr/3160.

```
67 \def\add@accent#1#2{\hmode@bgroup
```

Turn off the group in `\UseTextSymbol` in case this is used inside the argument of `\add@accent`.

```
68 \let\hmode@start@before@group\@firstofone
69 \setbox\@tempboxa\hbox{#2}
```

When presetting the argument in a box we record its `\spacefactor` for later use after the accent got typeset. This way something like `\‘A` gets the spacefactor of `A` (i.e., 999) rather than the default value of 1000.

```
70 \global\mathchardef\accent@spacefactor\spacefactor}%
```

The accent primitive doesn't allow things `\begingroup` to interfere between accent and base character. Therefore we need to avoid that (they are some hidden inside `\maybe@load@fontshape`). As we don't have to load the fontshape in this case (as that happened in the box above if necessary, we simply disable that part of the code temporarily. We also ignore `\ignorespaces` which has the same issue and may show up as part of `\normalfont` if that is used.

```
71 \let\maybe@load@fontshape\relax
72 \let\ignorespaces\relax
73 \accent#1 #2\egroup\spacefactor\accent@spacefactor}
```

Default definition for `\accent@spacefactor` prevents a horrible death of the above macro inside an unprotected `\edef`.

```
74 \let\accent@spacefactor\relax
```

(End definition for `\add@accent`.)

`\hmode@bgroup`

```
75 \def\hmode@bgroup{\leavevmode\bgroup}
```

(End definition for `\hmode@bgroup`.)

`\DeclareTextCompositeCommand` Another amusing game to play with `\expandafter`, `\csname`, and `\string`. When you say `\DeclareTextCompositeCommand{\foo}{T1}{a}{bar}`, we look to see if the expansion of `\T1\foo` begins with `\@text@composite`, and if it doesn't, we redefine `\T1\foo` to be:

```
\@text@composite
\@text@composite@x
\@strip@args
#1 -> \@text@composite \T1\foo #1\@empty \@text@composite {...}
```

where `...` is the previous definition of `\T1\foo`. Finally, we define `\T1\foo-a` to expand to `bar`.

```
76 </2kernel>
77 <latexrelease>\IncludeInRelease{2017/04/15}{\DeclareTextCompositeCommand}
78 <latexrelease> {test for undeclared accent}%
79 <*2kernel | latexrelease>
80 \def\DeclareTextCompositeCommand#1#2#3#4{%
81 \expandafter\let\expandafter\reserved@a\csname#2\string#1\endcsname
82 \ifx\reserved@a\relax
83 \DeclareTextCommand#1{#2}{%
84 \@latex@error{\string#1 undeclared in encoding #2}\@eha}%
85 \@latex@info{Composite with undeclared \string#1 in encoding #2}%
86 \expandafter\let\expandafter\reserved@a\csname#2\string#1\endcsname
87 \fi
88 \expandafter\expandafter\expandafter\ifx
89 \expandafter\@car\reserved@a\relax\relax\@nil \@text@composite \else
90 \edef\reserved@b##1{%
```

```

91         \def\expandafter\noexpand
92         \csname#2\string#1\endcsname###1{%
93         \noexpand\@text@composite
94         \expandafter\noexpand\csname#2\string#1\endcsname
95         ###1\noexpand\@empty\noexpand\@text@composite
96         {##1}}}%
97     \expandafter\reserved@b\expandafter{\reserved@a{##1}}%
98 \fi
99 \expandafter\def\csname\expandafter\string\csname
100     #2\endcsname\string#1-\string#3\@empty\endcsname{##4}%
101 }
102 </2ekernel | latexrelease>
103 <latexrelease>\EndIncludeInRelease
104 <latexrelease>\IncludeInRelease{0000/00/00}{\DeclareTextCompositeCommand}
105 <latexrelease>                                     {test for undeclared accent}%
106 <latexrelease>\def\DeclareTextCompositeCommand#1#2#3#4{%
107 <latexrelease> \expandafter\let\expandafter\reserved@a
108 <latexrelease>                                     \csname#2\string#1\endcsname
109 <latexrelease> \expandafter\expandafter\expandafter\ifx
110 <latexrelease> \expandafter\@car\reserved@a\relax\relax\@nil
111 <latexrelease>                                     \@text@composite \else
112 <latexrelease> \edef\reserved@b##1{%
113 <latexrelease> \def\expandafter\noexpand
114 <latexrelease> \csname#2\string#1\endcsname###1{%
115 <latexrelease> \noexpand\@text@composite
116 <latexrelease> \expandafter\noexpand\csname#2\string#1\endcsname
117 <latexrelease> ###1\noexpand\@empty\noexpand\@text@composite
118 <latexrelease> {##1}}}%
119 <latexrelease> \expandafter\reserved@b\expandafter{\reserved@a{##1}}%
120 <latexrelease> \fi
121 <latexrelease> \expandafter\def\csname\expandafter\string\csname
122 <latexrelease>     #2\endcsname\string#1-\string#3\@empty\endcsname{##4}}
123 <latexrelease>\EndIncludeInRelease
124 <*2ekernel>
125 \@onlypreamble\DeclareTextCompositeCommand

```

This all works because:

```
\@text@composite \T1\foo A\@empty \@text@composite {...}
```

expands to `\T1\foo-A` if `\T1\foo-A` has been defined, and `{...}` otherwise.

Note that `\@text@composite` grabs the first token of the argument and puts just that in the csname. This is so that `\'\textit{e}` will work—it checks whether `\T1\'-\textit` is defined (which presumably it isn't) and so expands to `\accent 1 \textit{e}`.

This trick won't always work, for example `\'\itshape e` will expand to (with spaces added for clarity):

```
\csname \string \T1\' - \string {\itshape e} \@empty \endcsname
```

which will die pretty horribly. Unfortunately there's not much can be done about this if we're going to use `\csname` lookups as a fast way of accessing composites.

This has an unfortunate 'misfeature' though, which is that in the T1 encoding, `\'aa` produces á. This is not the expected behaviour, and should perhaps be fixed if the fix doesn't affect performance too badly.

Finally, it's worth noting that the `\@empty` is used in `\@text@composite` so that accents will work even when the argument is empty. If you say `\'{}` then this looks up `\T1\'-\@empty`, which ought to be `\relax`, and so all is well. If we didn't include the `\@empty`, then `\'{}` would expand to:

```
\csname \string \T1\' - \string \endcsname
```

so the `\endcsname` would be `\string'ed` and the whole of the rest of the document would be put inside the `\csname`. This would not be good.

```
126 \def\@text@composite#1#2#3\@text@composite{%
127   \expandafter\@text@composite@x
128     \csname\string#1-\string#2\endcsname}
```

Originally the `\@text@composite@x` macro had two arguments and if `#1` was not `\relax` it was executed, otherwise `#2` was executed. All this happened within the `\ifx` code so that neither `#1` nor `#2` could have picked up any additional arguments from the input stream. This has now been changed using the typical `\@firstoftwo / \@secondoftwo` coding. This way the final expansion will happen without any `\else` or `\fi` intervening in the case that we need to get a further token from the input stream.

```
129 \def\@text@composite@x#1{%
130   \ifx#1\relax
131     \expandafter\@secondoftwo
132   \else
133     \expandafter\@firstoftwo
134   \fi
135   #1}
```

The command `\DeclareTextComposite` uses `\DeclareTextCompositeCommand` to declare a command which expands out to a single glyph.

```
136 \catcode\z@=11\relax
137 \def\DeclareTextComposite#1#2#3#4{%
138   \def\reserved@a{\DeclareTextCompositeCommand#1{#2}{#3}}%
139   \bgroup
140     \lccode\z@#4%
141     \lowercase{%
142   \egroup
143     \reserved@a ^^@}}
144 \catcode\z@=15\relax
145 \@onlypreamble\DeclareTextComposite
(End definition for \DeclareTextCompositeCommand and others.)
146 </2ekernel>
147 <*2ekernel | latexrelease>
148 <latexrelease>\IncludeInRelease{2019/10/01}%
149 <latexrelease>           {\UseTextAccent}{Make commands robust}}%
```

`\UseTextAccent` These fragile commands access glyphs from different encodings. They use grotty low-level calls to the font selection scheme for speed, and in order to make sure that `\UseTextSymbol` doesn't do anything which you're not allowed to do between an `\accent` and its glyph.

`\@use@text@encoding`

For a detailed discussion of this reimplementaion and its deficiencies, see pr/3160.


```

150 \DeclareRobustCommand*\UseTextAccent[3]{%
151   \hmode@start@before@group
152   {%

```

Turn off the group in `\UseTextSymbol` in case this is used inside the arguments of `\UseTextAccent`.

```

153   \let\hmode@start@before@group\@firstofone
154   \let\@curr@enc\cf@encoding
155   \@use@text@encoding{#1}%
156   #2{\@use@text@encoding\@curr@enc#3}%
157   }}

```

```

158 \DeclareRobustCommand*\UseTextSymbol[2]{%
159   \hmode@start@before@group
160   {%
161     \def\@wrong@font@char{\MessageBreak
162       for \noexpand\symbol'\string#2'}%
163     \@use@text@encoding{#1}%
164     #2%
165   }%
166   }

167 </2ekernel | latexrelease>
168 <latexrelease>\EndIncludeInRelease
169 <latexrelease>\IncludeInRelease{0000/00/00}%
170 <latexrelease>          {\UseTextAccent}{Make commands robust}%
171 <latexrelease>
172 <latexrelease>\kernel@make@fragile\UseTextAccent
173 <latexrelease>\kernel@make@fragile\UseTextSymbol
174 <latexrelease>
175 <latexrelease>\EndIncludeInRelease
176 <*2ekernel>

```

Switch to a different text encoding without any grouping for use in `\UseTextAccent` or `\UseTextSymbol` (and for `\oldstylenums`).

```

177 \def\@use@text@encoding#1{%
178   \edef\f@encoding{#1}%
179   \xdef\font@name{%
180     \csname\curr@fontshape/\f@size\endcsname}%
181   \pickup@font
182   \font@name
183   \@@enc@update}

```

(End definition for `\UseTextAccent`, `\UseTextSymbol`, and `\@use@text@encoding`.)

`\hmode@start@before@group` The `\hmode@start@before@group` starts hmode and should be immediately followed by an explicit `{...}`. Its purpose is to ensure that hmode is started before this group is opened. Inside `\add@accent` and `\UseTextAccent` it is redefined to remove this group so that it doesn't conflict with the `\accent` primitive.

For a detailed discussion see pr/3160.

```

184 \let\hmode@start@before@group\leavevmode

```

(End definition for `\hmode@start@before@group`.)

`\DeclareTextSymbolDefault` Some syntactic sugar. Again, these should probably be optimized for speed.

```
\DeclareTextAccentDefault
185 \def\DeclareTextSymbolDefault#1#2{%
186   \DeclareTextCommandDefault#1{\UseTextSymbol{#2}#1}}
187 \def\DeclareTextAccentDefault#1#2{%
188   \DeclareTextCommandDefault#1{\UseTextAccent{#2}#1}}
189 \@onlypreamble\DeclareTextSymbolDefault
190 \@onlypreamble\DeclareTextAccentDefault
```

(End definition for \DeclareTextSymbolDefault and \DeclareTextAccentDefault.)

`\UndeclareTextCommand` This command safely removes an encoding specific declaration for a given encoding. It is helpful if one intends to use the default definition always and therefore wants to get rid of a declaration for some specific encoding.

```
191 \def\UndeclareTextCommand#1#2{%
If there is no declaration for the current encoding do nothing. (This makes a hash table
entry but without eTeX we can't do anything about that).
192   \expandafter\ifx\csname#2\string#1\endcsname\relax
193   \else
```

Else: throw away that declaration.

```
194     \global\expandafter\let\csname#2\string#1\endcsname
195     \undefined
```

But this is unfortunately not enough, we have to take a look at the top-level definition of the encoding specific command which for a command `\foo` would look similar to `\T1-cmd \foo \T1\foo` (three tokens).

Of course, instead of `T1` one could see a different encoding name; which one depends the encoding for which `\foo` was declared last.

Now assume we have just removed the declaration for `\foo` in `T1` and the top-level of `\foo` expands to the above. Then we better change that pretty fast otherwise we do get an “undefined csname error” when we try to typeset `\foo` within `T1` instead of getting the default definition for `\foo`. And what is the best way to change that top-level definition? Well, the only “encoding” we know for sure will still be around is the default encoding denoted by `?`.

Thus in case the last token of the top-level expansion is now undefined we change the declaration to look like `\?-cmd \foo \?\foo` which is done by the following (readable?) code:

```
196   \expandafter\expandafter\expandafter
197   \ifx\expandafter\@thirdofthree#1\@undefined
198     \expandafter\gdef\expandafter#1\expandafter
199     {\csname ?-cmd\expandafter\endcsname\expandafter
200      #1\csname?\string#1\endcsname}%
201   \fi
202   \fi
203 }
```

```
204 \@onlypreamble\UndeclareTextCommand
```

(End definition for \UndeclareTextCommand.)

1.4.2 Hyphenation

`\patterns` We redefine `\patterns` and `\hyphenation` to allow the use of commands declared with
`\@@patterns` `\DeclareText*` to be used inside them.
`\hyphenation`
`\@@hyphenation`

```
205 %\let\@@patterns\patterns
206 %\let\@@hyphenation\hyphenation
207 %\def\patterns{%
208 %   \bgroup
209 %     \let\protect\@empty
210 %     \let\@typeset@protect\@empty
211 %     \let\@changed@x\@changed@x@mouth
212 %   \afterassignment\egroup
213 %   \@@patterns
214 %}
215 %\def\hyphenation{%
216 %   \bgroup
217 %     \let\protect\@empty
218 %     \let\@typeset@protect\@empty
219 %     \let\@changed@x\@changed@x@mouth
220 %   \afterassignment\egroup
221 %   \@@hyphenation
222 %}
```

(End definition for `\patterns` and others.)

1.4.3 Miscellania

`\a` The `\a` command is used to access the accent commands even when they have been redefined (for example by the `\tabbing` environment). Its internal name is `\@tabacckludge`.
The `\string` within the `\csname` guards against something like `'` being active at the point of use.

```
223 \def\@tabacckludge#1{\expandafter\@changed@cmd
224                                     \csname\string#1\endcsname\relax}
225 \let\a=\@tabacckludge
```

(End definition for `\a`.)

1.4.4 Default encodings

We define the default encodings for most commands to be either OT1, OML or OMS. These defaults are in the kernel and therefore fonts with these encodings must be available unless these defaults are redefined elsewhere. Recall that the standard kernel loads the encoding files for these encodings, and also that for the T1 encoding.

The naming conventions in the kernel are not what we would use if we were starting from scratch... Those defined by DEK (like `\ae` and `\ss`) or by the T_EX Users Group Technical Working Group on multi-lingual typesetting (like `\th` and `\ng`) have short names. Those which were added to the kernel in 1993 and early 1994 are named after their Adobe glyph names (like `\guillemotleft` and `\quotedblbase`). Unfortunately, this naming scheme won't work for all glyphs, since some names (like `\space`) are already used, and some (like `\endash`) are very likely to be defined by users. So we're now using the naming scheme of `\text` followed by the Adobe name, (like `\textendash` and `\textsterling`). Except that some glyphs don't have Adobe names, so we're using the names used by fontinst for those (like `\textcompwordmark`). Sigh.

Some accents from OT1:

```
226 \DeclareTextAccentDefault{"}{OT1}
227 \DeclareTextAccentDefault{'}{OT1}
228 \DeclareTextAccentDefault{.}{OT1}
229 \DeclareTextAccentDefault{=}{OT1}
230 \DeclareTextAccentDefault{H}{OT1}
231 \DeclareTextAccentDefault{^}{OT1}
232 \DeclareTextAccentDefault{'}{OT1}
233 \DeclareTextAccentDefault{b}{OT1}
234 \DeclareTextAccentDefault{c}{OT1}
235 \DeclareTextAccentDefault{d}{OT1}
236 \DeclareTextAccentDefault{r}{OT1}
237 \DeclareTextAccentDefault{u}{OT1}
238 \DeclareTextAccentDefault{v}{OT1}
239 \DeclareTextAccentDefault{~}{OT1}
```

Some symbols from OT1:

```
240 %\DeclareTextSymbolDefault{AA}{OT1}
241 \DeclareTextSymbolDefault{AE}{OT1}
242 \DeclareTextSymbolDefault{L}{OT1}
243 \DeclareTextSymbolDefault{OE}{OT1}
244 \DeclareTextSymbolDefault{O}{OT1}
245 %\DeclareTextSymbolDefault{aa}{OT1}
246 \DeclareTextSymbolDefault{ae}{OT1}
247 \DeclareTextSymbolDefault{i}{OT1}
248 \DeclareTextSymbolDefault{j}{OT1}

249 \DeclareTextSymbolDefault{ij}{OT1}
250 \DeclareTextSymbolDefault{IJ}{OT1}

251 \DeclareTextSymbolDefault{l}{OT1}
252 \DeclareTextSymbolDefault{oe}{OT1}
253 \DeclareTextSymbolDefault{o}{OT1}
254 \DeclareTextSymbolDefault{ss}{OT1}
255 \DeclareTextSymbolDefault{textdollar}{OT1}
256 \DeclareTextSymbolDefault{textemdash}{OT1}
257 \DeclareTextSymbolDefault{textendash}{OT1}
258 \DeclareTextSymbolDefault{textexclamdown}{OT1}
259 %\DeclareTextSymbolDefault{texthyphenchar}{OT1}
260 %\DeclareTextSymbolDefault{texthyphen}{OT1}
261 \DeclareTextSymbolDefault{textquestiondown}{OT1}
262 \DeclareTextSymbolDefault{textquotedblleft}{OT1}
263 \DeclareTextSymbolDefault{textquotedblright}{OT1}
264 \DeclareTextSymbolDefault{textquoteleft}{OT1}
265 \DeclareTextSymbolDefault{textquoteright}{OT1}
266 \DeclareTextSymbolDefault{textsterling}{OT1}
```

Some symbols from OMS:

```
267 \DeclareTextSymbolDefault{textasteriskcentered}{OMS}
268 \DeclareTextSymbolDefault{textbackslash}{OMS}
269 \DeclareTextSymbolDefault{textbar}{OMS}
270 \DeclareTextSymbolDefault{textbardbl}{OMS}
271 \DeclareTextSymbolDefault{textbraceleft}{OMS}
272 \DeclareTextSymbolDefault{textbraceright}{OMS}
273 \DeclareTextSymbolDefault{textbullet}{OMS}
```

```

274 \DeclareTextSymbolDefault{\textdaggerdbl}{OMS}
275 \DeclareTextSymbolDefault{\textdagger}{OMS}
276 \DeclareTextSymbolDefault{\textparagraph}{OMS}
277 \DeclareTextSymbolDefault{\textperiodcentered}{OMS}
278 \DeclareTextSymbolDefault{\textsection}{OMS}
279 \DeclareTextAccentDefault{\textcircled}{OMS}

```

Some symbols from OML:

```

280 \DeclareTextSymbolDefault{\textless}{OML}
281 \DeclareTextSymbolDefault{\textgreater}{OML}
282 \DeclareTextAccentDefault{\t}{OML}

```

Some defaults we can fake.

The interface for defining `\copyright` changed, it used to use `\expandafter` to add braces at the appropriate points.

```

283 \DeclareTextCommandDefault{\textcopyright}{\textcircled{c}}
284 % \expandafter\def\expandafter
285 % \copyright\expandafter{\expandafter{\copyright}}

286 \DeclareTextCommandDefault{\textasciicircum}{\~{}}
287 \DeclareTextCommandDefault{\textasciitilde}{\~{}}
288 \DeclareTextCommandDefault{\textunderscore}{%
289 \leavevmode \kern.06em\vbox{\hrule\@width.3em}}

```

There is no good reason anymore to fake `\textcompwordmark`.

```

290 %\DeclareTextCommandDefault{\textcompwordmark}{\leavevmode\kern\z@}
291 \DeclareTextSymbolDefault{\textcompwordmark}{T1}

292 \DeclareTextCommandDefault{\textvisiblespace}{%
293 \mbox{\kern.06em\vrule \@height.3ex}%
294 \vbox{\hrule \@width.3em}%
295 \hbox{\vrule \@height.3ex}}

```

Using `\fontdimen3` in the next definition is some sort of a kludge (since it is the interword stretch) but it makes the ellipsis come out right in mono-spaced fonts too (since there it is zero).

```

296 \DeclareTextCommandDefault{\textellipsis}{%
297 .\kern\fontdimen3\font
298 .\kern\fontdimen3\font
299 .\kern\fontdimen3\font}

300 %\DeclareTextCommandDefault{\textregistered}{\textcircled{\scshape r}}
301 \DeclareTextCommandDefault{\textregistered}{\textcircled{%
302 \check@mathfonts\fontsize\sf@size\z@\math@fontfalse\selectfont R}}
303 \DeclareTextCommandDefault{\texttrademark}{\textsuperscript{TM}}
304 \DeclareTextCommandDefault{\SS}{SS}

305 \DeclareTextCommandDefault{\textordfeminine}{\textsuperscript{a}}
306 \DeclareTextCommandDefault{\textordmasculine}{\textsuperscript{o}}

```

1.4.5 Math material

Some commands can be used in both text and math mode:

```
307 \DeclareRobustCommand{\$}{\ifmmode\mathdollar\else\textdollar\fi}
```

We use `\protected` not `\DeclareRobustCommand` so that `\bigl\{` etc. works inside `\protected@edef`.

```
308 \protected\def{{\ifmmode\lbrace\else\textbraceleft\fi}
```

```
309 \protected\def\}{\ifmmode\rbrace\else\textbraceright\fi}
```

```
310 \DeclareRobustCommand{\P}{\ifmmode\mathparagraph\else\textparagraph\fi}
```

```
311 \DeclareRobustCommand{\S}{\ifmmode\mathsection\else\textsection\fi}
```

```
312 \DeclareRobustCommand{\dag}{\ifmmode{\dagger}\else\textdagger\fi}
```

```
313 \DeclareRobustCommand{\ddag}{\ifmmode{\ddagger}\else\textdaggerdbl\fi}
```

For historical reasons `\copyright` needs `{}` around the definition in maths.

```
314 \DeclareRobustCommand{\_}{%
```

```
315   \ifmmode\nfss@text{\textunderscore}\else\textunderscore\fi}
```

```
316 \DeclareRobustCommand{\copyright}{%
```

```
317   \ifmmode{\nfss@text{\textcopyright}}\else\textcopyright\fi}
```

```
318 \DeclareRobustCommand{\pounds}{%
```

```
319   \ifmmode\mathsterling\else\textsterling\fi}
```

```
320 \DeclareRobustCommand{\dots}{%
```

```
321   \ifmmode\mathellipsis\else\textellipsis\fi}
```

```
322 \let\ldots\dots
```

Default definition of the `commabelow` accent.

```
323 </2ekernel>
```

```
324 <latexrelease>\IncludeInRelease{2015/10/01}{\textcommabelow}{comma accent}%
```

```
325 <*2ekernel | latexrelease>
```

```
326 \DeclareTextCommandDefault\textcommabelow[1]
```

```
327   {\hmode\bgroup\ooalign{\null#1\cr\hidewidth\raise-.31ex
```

```
328     \hbox{\check@mathfonts\fontsize\ssf@size\z@
```

```
329     \math@fontsfalse\selectfont,}\hidewidth}\egroup}
```

```
330 <latexrelease>\EndIncludeInRelease
```

```
331 </2ekernel | latexrelease>
```

```
332 <latexrelease>\IncludeInRelease{0000/00/00}{\textcommabelow}{comma accent}%
```

```
333 <latexrelease>\let\textcommabelow@undefined
```

```
334 <latexrelease>\expandafter
```

```
335 <latexrelease> \let\csname\string\T1\string\c-G\endcsname@undefined
```

```
336 <latexrelease>\expandafter
```

```
337 <latexrelease> \let\csname\string\T1\string\c-K\endcsname@undefined
```

```
338 <latexrelease>\expandafter
```

```
339 <latexrelease> \let\csname\string\T1\string\c-k\endcsname@undefined
```

```
340 <latexrelease>\expandafter
```

```
341 <latexrelease> \let\csname\string\T1\string\c-L\endcsname@undefined
```

```
342 <latexrelease>\expandafter
```

```
343 <latexrelease> \let\csname\string\T1\string\c-l\endcsname@undefined
```

```
344 <latexrelease>\expandafter
```

```
345 <latexrelease> \let\csname\string\T1\string\c-N\endcsname@undefined
```

```
346 <latexrelease>\expandafter
```

```
347 <latexrelease> \let\csname\string\T1\string\c-n\endcsname@undefined
```

```
348 <latexrelease>\expandafter
```

```
349 <latexrelease> \let\csname\string\T1\string\c-R\endcsname@undefined
```

```

350 <latexrelease>\expandafter
351 <latexrelease> \let\cename\string\T1\string\c-r\endcsname\@undefined
352 <latexrelease>\EndIncludeInRelease

    Default definition of the commaabove accent(E.G.).

353 <latexrelease>\IncludeInRelease{2016/02/01}{\textcommaabove}{comma above}%
354 <*2ekernel | latexrelease>
355 \DeclareTextCommandDefault\textcommaabove[1]{%
356   \hmode\bgroup
357   \ooalign{%
358     \hidewidth
359     \raise.7ex\hbox{%
360       \check@mathfonts\fontsize\ssf@size\z@\math@fontsfalse\selectfont‘%
361     }%
362     \hidewidth\crrcr
363     \null#1\crrcr
364   }%
365   \egroup
366 }
367 <latexrelease>\EndIncludeInRelease
368 </2ekernel | latexrelease>
369 <latexrelease>\IncludeInRelease{0000/00/00}{\textcommaabove}{comma above}%
370 <latexrelease>\let\textcommaabove\@undefined
371 <latexrelease>\expandafter
372 <latexrelease> \let\cename\string\OT1\string\c-g\endcsname\@undefined
373 <latexrelease>\expandafter
374 <latexrelease> \let\cename\string\T1\string\c-g\endcsname\@undefined
375 <latexrelease>\EndIncludeInRelease

```

1.5 Definitions for the OT1 encoding

The definitions for the ‘TEX text’ (OT1) encoding.

Declare the encoding.

```

376 <*OT1>
377 \DeclareFontEncoding{OT1}{-}{-}

```

Declare the accents.

```

378 \DeclareTextAccent{"}{OT1}{127}
379 \DeclareTextAccent{'}{OT1}{19}
380 \DeclareTextAccent{.}{OT1}{95}
381 \DeclareTextAccent{=}{OT1}{22}
382 \DeclareTextAccent{^}{OT1}{94}
383 \DeclareTextAccent{'}{OT1}{18}
384 \DeclareTextAccent{\~}{OT1}{126}
385 \DeclareTextAccent{\H}{OT1}{125}
386 \DeclareTextAccent{\u}{OT1}{21}
387 \DeclareTextAccent{\v}{OT1}{20}
388 \DeclareTextAccent{\r}{OT1}{23}

```

Some accents have to be built by hand: Note that `\ooalign` and `\o@lign` must be inside a group. In these definitions we no longer use the helper function `\sh@ft` from `plain.tex` since that now has two incompatible definitions.

```

389 \DeclareTextCommand{\b}{OT1}[1]
390   {\hmode\bgroup\o@lign{\relax#1\crrcr\hidewidth\ltx@sh@ft{-3ex}}%

```

```

391 \vbox to.2ex{\hbox{\char22}\vss}\hidewidth}\egroup}
392 \DeclareTextCommand{\c}{OT1}[1]
393 {\leavevmode\setbox\z@\hbox{#1}\ifdim\ht\z@=1ex\accent24 #1%
394 \else\oalign{\unhbox\z@\crrc\hidewidth\char24\hidewidth}}\fi}
395 \DeclareTextCommand{\d}{OT1}[1]
396 {\hmode\bgroup
397 \oalign{\relax#1\crrc\hidewidth\ltx@sh@ft{-1ex}.\hidewidth}\egroup}

```

Declare the text symbols.

```

398 \DeclareTextSymbol{\AE}{OT1}{29}
399 \DeclareTextSymbol{\OE}{OT1}{30}
400 \DeclareTextSymbol{\O}{OT1}{31}
401 \DeclareTextSymbol{\ae}{OT1}{26}
402 \DeclareTextSymbol{\i}{OT1}{16}
403 \DeclareTextSymbol{\j}{OT1}{17}
404 \DeclareTextSymbol{\oe}{OT1}{27}
405 \DeclareTextSymbol{\o}{OT1}{28}
406 \DeclareTextSymbol{\ss}{OT1}{25}
407 \DeclareTextSymbol{\textendash}{OT1}{124}
408 \DeclareTextSymbol{\textdash}{OT1}{123}

```

The `\nobreak\hskip\z@` is there to prevent a break after the hyphen but allow later breaks in the remainder of the word.

```

409 \DeclareTextCommand{\textnonbreakinghyphen}{OT1}{\mbox{-}\nobreak\hskip\z@}
410 \DeclareTextCommand{\textfiguredash}{OT1}{\textendash}
411 \DeclareTextCommand{\texthorizontalbar}{OT1}{\textdash}

```

Using the ligatures helps with OT1 fonts that have `\textexclamdown` and `\textquestiondown` in unusual positions.

```

412 %\DeclareTextSymbol{\textexclamdown}{OT1}{60}
413 %\DeclareTextSymbol{\textquestiondown}{OT1}{62}
414 \DeclareTextCommand{\textexclamdown}{OT1}{\!'}
415 \DeclareTextCommand{\textquestiondown}{OT1}{\?' }
416 %\DeclareTextSymbol{\texthyphenchar}{OT1}{'\-}
417 %\DeclareTextSymbol{\texthyphen}{OT1}{'\-}
418 \DeclareTextSymbol{\textquotedblleft}{OT1}{92}
419 \DeclareTextSymbol{\textquotedblright}{OT1}{'\'}
420 \DeclareTextSymbol{\textquoteleft}{OT1}{'\'}
421 \DeclareTextSymbol{\textquoteright}{OT1}{'\'}

```

Some symbols which are faked from others:

```

422 % \DeclareTextCommand{\aa}{OT1}
423 % {\accent23a}
424 \DeclareTextCommand{\L}{OT1}
425 {\leavevmode\setbox\z@\hbox{L}\hb@xt@\wd\z@{\hss\@xxxii L}}
426 \DeclareTextCommand{\l}{OT1}
427 {\hmode\bgroup\@xxxii l\egroup}
428 % \DeclareTextCommand{\AA}{OT1}
429 % {\leavevmode\setbox\z@\hbox{h}\dimen@ht\z@\advance\dimen@-1ex%
430 % \rlap{\raise.67\dimen@\hbox{\char23}}A}

```

In the OT1 encoding Å has a hand-crafted definition, so we have here the first recorded explicit use of `\DeclareTextCompositeCommand`.

```

431 \DeclareTextCompositeCommand{\r}{OT1}{A}
432 {\leavevmode\setbox\z@\hbox{!}\dimen@ht\z@\advance\dimen@-1ex%
433 \rlap{\raise.67\dimen@\hbox{\char23}}A}

```


The dutch language uses the letter ‘ij’. It is available in T1 encoded fonts, but not in the OT1 encoded fonts. Therefor we fake it for the OT1 encoding.

```

434 \DeclareTextCommand{\ij}{OT1}{%
435   \nobreak\hskip\z@skip i\kern-0.02em j\nobreak\hskip\z@skip}
436 \DeclareTextCommand{\IJ}{OT1}{%
437   \nobreak\hskip\z@skip I\kern-0.02em J\nobreak\hskip\z@skip}

```

In the OT1 encoding, £ and \$ share a slot.

```

438 \DeclareTextCommand{\textdollar}{OT1}{\hmode@bgroup
439   \ifdim \fontdimen\@ne\font >\z@
440     \slshape
441   \else
442     \upshape
443   \fi
444   \char‘\$ \egroup}

445 \DeclareTextCommand{\textsterling}{OT1}{\hmode@bgroup
446   \ifdim \fontdimen\@ne\font >\z@
447     \itshape
448   \else
449     \fontshape{ui}\selectfont
450   \fi
451   \char‘\$ \egroup}

```

Here we are adding some more composite commands to the OT1 encoding. This makes the use of certain accents with i compatible with their use with the T1 encoding; this enables them to become true L^AT_EX internal representations. However, it will make these accents work a little less fast since a check will always be made for the existence of a composite.

```

452 \DeclareTextComposite{\.}{OT1}{i}{‘\i}
453 \DeclareTextComposite{\.}{OT1}{i}{‘\i}
454 \DeclareTextCompositeCommand{\'}{OT1}{i}{\@tabacckludge‘\i}
455 \DeclareTextCompositeCommand{\'}{OT1}{i}{\@tabacckludge‘\i}
456 \DeclareTextCompositeCommand{\~}{OT1}{i}{\^{\i}}
457 \DeclareTextCompositeCommand{\~}{OT1}{i}{\^{\i}}

```

T1 encoding is given more extensive set of overloads for \c But here we just adjust \c{g}.

```

458 \ifx\textcommaabove\undefined\else
459 \DeclareTextCompositeCommand{\c}{OT1}{g}{\textcommaabove{g}}
460 \fi
461 </OT1>

```

1.6 Definitions for the T1 encoding

The definitions for the ‘Extended T_EX text’ (T1) encoding.

Declare the encoding.

```

462 <*T1>
463 \DeclareFontEncoding{T1}{}{}

```

Declare the accents.

```

464 \DeclareTextAccent{\'}{T1}{0}
465 \DeclareTextAccent{\'}{T1}{1}
466 \DeclareTextAccent{\~}{T1}{2}

```

```

467 \DeclareTextAccent{\~}{T1}{3}
468 \DeclareTextAccent{\"}{T1}{4}
469 \DeclareTextAccent{\H}{T1}{5}
470 \DeclareTextAccent{\r}{T1}{6}
471 \DeclareTextAccent{\v}{T1}{7}
472 \DeclareTextAccent{\u}{T1}{8}
473 \DeclareTextAccent{\=}{T1}{9}
474 \DeclareTextAccent{\.}{T1}{10}

```

Some accents have to be built by hand. Note that `\ooalign` and `\oalign` must be inside a group. In these definitions we no longer use the helper function `\sh@ft` from `plain.tex` since that now has two incompatible definitions.

```

475 \DeclareTextCommand{\b}{T1}[1]
476   {\hmode\bgroup\oalign{\relax#1\crcr\hidewidth\ltx@sh@ft{-3ex}%
477     \vbox to.2ex{\hbox{\char9}\vss}\hidewidth}\egroup}
478 \DeclareTextCommand{\c}{T1}[1]
479   {\leavevmode\setbox\z@\hbox{#1}\ifdim\ht\z@=1ex\accent11 #1%
480     \else{\ooalign{\unhbox\z@\crcr
481       \hidewidth\char11\hidewidth}}\fi}
482 \DeclareTextCommand{\d}{T1}[1]
483   {\hmode\bgroup
484     \oalign{\relax#1\crcr\hidewidth\ltx@sh@ft{-1ex}.\hidewidth}\egroup}
485 \DeclareTextCommand{\k}{T1}[1]
486   {\hmode\bgroup\ooalign{\null#1\crcr\hidewidth\char12}\egroup}
487 \DeclareTextCommand{\textogonekcentered}{T1}[1]
488   {\hmode\bgroup\ooalign{%
489     \null#1\crcr\hidewidth\char12\hidewidth}\egroup}

```

Some symbols are constructed.

Slot 24 contains a small circle intended for construction of these two glyphs.

```

490 \DeclareTextCommand{\textperthousand}{T1}
491   {\%\char 24 } % space or 'relax as delimiter?
492 \DeclareTextCommand{\textpertenthousand}{T1}
493   {\%\char 24\char 24 } % space or 'relax as delimiter?

```

For Maltese, `\Hwithstroke` and `\hwithstroke` are needed.

```

494 \DeclareTextCommand{\Hwithstroke}{T1}
495   {%
496     \hmode\bgroup
497     \vphantom{H}%
498     \sbox\z@{H}%
499     \ooalign{%
500       H\cr
501       \hidewidth
502       \vrule
503         height \dimexpr 0.7\ht\z@+0.1ex\relax
504         depth -0.7\ht\z@
505         width 0.8\wd\z@
506       \hidewidth\cr
507     }%
508     \egroup
509   }
510 \DeclareTextCommand{\hwithstroke}{T1}
511   {%

```

```

512 \hmode@bgroup
513 \vphantom{h}%
514 \sbox\z@{h}%
515 \ooalign{%
516   h\cr
517   \kern0.075\wd\z@
518   \vrule
519     height \dimexpr 0.7\ht\z@+0.1ex\relax
520     depth  -0.7\ht\z@
521     width  0.4\wd\z@
522   \hidewidth\cr
523 }%
524 \egroup
525 }

```

Declare the text symbols.

```

526 %\DeclareTextSymbol{\AA}{T1}{197}
527 \DeclareTextSymbol{\AE}{T1}{198}
528 \DeclareTextSymbol{\DH}{T1}{208}
529 \DeclareTextSymbol{\DJ}{T1}{208}
530 \DeclareTextSymbol{\L}{T1}{138}
531 \DeclareTextSymbol{\NG}{T1}{141}
532 \DeclareTextSymbol{\OE}{T1}{215}
533 \DeclareTextSymbol{\O}{T1}{216}
534 \DeclareTextSymbol{\SS}{T1}{223}
535 \DeclareTextSymbol{\TH}{T1}{222}
536 %\DeclareTextSymbol{\aa}{T1}{229}
537 \DeclareTextSymbol{\ae}{T1}{230}
538 \DeclareTextSymbol{\dh}{T1}{240}
539 \DeclareTextSymbol{\dj}{T1}{158}

540 \DeclareTextSymbol{\guillemetleft}{T1}{19}
541 \DeclareTextSymbol{\guillemetright}{T1}{20}
542 % old Adobe names
543 \DeclareTextSymbol{\guillemotleft}{T1}{19}
544 \DeclareTextSymbol{\guillemotright}{T1}{20}

545 \DeclareTextSymbol{\guilsinglleft}{T1}{14}
546 \DeclareTextSymbol{\guilsinglright}{T1}{15}
547 \DeclareTextSymbol{\i}{T1}{25}
548 \DeclareTextSymbol{\j}{T1}{26}
549 \DeclareTextSymbol{\ij}{T1}{188}
550 \DeclareTextSymbol{\IJ}{T1}{156}
551 \DeclareTextSymbol{\l}{T1}{170}
552 \DeclareTextSymbol{\ng}{T1}{173}
553 \DeclareTextSymbol{\oe}{T1}{247}
554 \DeclareTextSymbol{\o}{T1}{248}
555 \DeclareTextSymbol{\quotedblbase}{T1}{18}
556 \DeclareTextSymbol{\quotesinglbase}{T1}{13}
557 \DeclareTextSymbol{\ss}{T1}{255}
558 \DeclareTextSymbol{\textasciicircum}{T1}{'\^}
559 \DeclareTextSymbol{\textasciitilde}{T1}{'\~}
560 \DeclareTextSymbol{\textbackslash}{T1}{'\}
561 \DeclareTextSymbol{\textbar}{T1}{'|}
562 \DeclareTextSymbol{\textbraceleft}{T1}{'\{ }

```

```

563 \DeclareTextSymbol{\textbraceright}{T1}{\}
564 \DeclareTextSymbol{\textcompwordmark}{T1}{23}
565 \DeclareTextSymbol{\textdollar}{T1}{\$}
566 \DeclareTextSymbol{\textendash}{T1}{22}
567 \DeclareTextSymbol{\textendash}{T1}{21}

```

The `\nobreak\hskip\z@` is there to prevent a break after the hyphen but allow later breaks in the remainder of the word.

```

568 \DeclareTextCommand{\textnonbreakinghyphen}{T1}{\mbox{-}\nobreak\hskip\z@}
569 \DeclareTextCommand{\textfiguredash}{T1}{\textendash}
570 \DeclareTextCommand{\texthorizontalbar}{T1}{\textendash}

571 \DeclareTextSymbol{\textexclamdown}{T1}{189}
572 \DeclareTextSymbol{\textgreater}{T1}{>}
573 %\DeclareTextSymbol{\textthyphenchar}{T1}{127}
574 %\DeclareTextSymbol{\textthyphen}{T1}{-}
575 \DeclareTextSymbol{\textless}{T1}{<}
576 \DeclareTextSymbol{\textquestiondown}{T1}{190}
577 \DeclareTextSymbol{\textquotedblleft}{T1}{16}
578 \DeclareTextSymbol{\textquotedblright}{T1}{17}
579 \DeclareTextSymbol{\textquotedbl}{T1}{"}
580 \DeclareTextSymbol{\textquoteleft}{T1}{'}
581 \DeclareTextSymbol{\textquoteright}{T1}{'}
582 \DeclareTextSymbol{\textsection}{T1}{159}
583 \DeclareTextSymbol{\textsterling}{T1}{191}
584 \DeclareTextSymbol{\textunderscore}{T1}{95}
585 \DeclareTextSymbol{\textvisiblespace}{T1}{32}
586 \DeclareTextSymbol{\th}{T1}{254}

```

Declare the composites.

```

587 \DeclareTextComposite{\.}{T1}{i}{\i}
588 \DeclareTextComposite{\.}{T1}{i}{\i}

```

"80 = 128

```

589 \DeclareTextComposite{\u}{T1}{A}{128}
590 \DeclareTextComposite{\k}{T1}{A}{129}
591 \DeclareTextComposite{\'}{T1}{C}{130}
592 \DeclareTextComposite{\v}{T1}{C}{131}
593 \DeclareTextComposite{\v}{T1}{D}{132}
594 \DeclareTextComposite{\v}{T1}{E}{133}
595 \DeclareTextComposite{\k}{T1}{E}{134}
596 \DeclareTextComposite{\u}{T1}{G}{135}

```

"88 = 136

```

597 \DeclareTextComposite{\'}{T1}{L}{136}
598 \DeclareTextComposite{\v}{T1}{L}{137}
599 \DeclareTextComposite{\'}{T1}{N}{139}
600 \DeclareTextComposite{\v}{T1}{N}{140}
601 \DeclareTextComposite{\H}{T1}{O}{142}
602 \DeclareTextComposite{\'}{T1}{R}{143}

```

"90 = 144

```

603 \DeclareTextComposite{\v}{T1}{R}{144}
604 \DeclareTextComposite{\'}{T1}{S}{145}
605 \DeclareTextComposite{\v}{T1}{S}{146}
606 \DeclareTextComposite{\c}{T1}{S}{147}

```

```

607 \DeclareTextComposite{\v}{T1}{T}{148}
608 \DeclareTextComposite{\c}{T1}{T}{149}
609 \DeclareTextComposite{\H}{T1}{U}{150}
610 \DeclareTextComposite{\r}{T1}{U}{151}
"98 = 152
611 \DeclareTextComposite{\"}{T1}{Y}{152}
612 \DeclareTextComposite{\'}{T1}{Z}{153}
613 \DeclareTextComposite{\v}{T1}{Z}{154}
614 \DeclareTextComposite{\.}{T1}{Z}{155}
615 \DeclareTextComposite{\.}{T1}{I}{157}
"A0 = 160
616 \DeclareTextComposite{\u}{T1}{a}{160}
617 \DeclareTextComposite{\k}{T1}{a}{161}
618 \DeclareTextComposite{\'}{T1}{c}{162}
619 \DeclareTextComposite{\v}{T1}{c}{163}
620 \DeclareTextComposite{\v}{T1}{d}{164}
621 \DeclareTextComposite{\v}{T1}{e}{165}
622 \DeclareTextComposite{\k}{T1}{e}{166}
623 \DeclareTextComposite{\u}{T1}{g}{167}
"A8 = 168
624 \DeclareTextComposite{\'}{T1}{l}{168}
625 \DeclareTextComposite{\v}{T1}{l}{169}
626 \DeclareTextComposite{\'}{T1}{n}{171}
627 \DeclareTextComposite{\v}{T1}{n}{172}
628 \DeclareTextComposite{\H}{T1}{o}{174}
629 \DeclareTextComposite{\'}{T1}{r}{175}
"B0 = 176
630 \DeclareTextComposite{\v}{T1}{r}{176}
631 \DeclareTextComposite{\'}{T1}{s}{177}
632 \DeclareTextComposite{\v}{T1}{s}{178}
633 \DeclareTextComposite{\c}{T1}{s}{179}
634 \DeclareTextComposite{\v}{T1}{t}{180}
635 \DeclareTextComposite{\c}{T1}{t}{181}
636 \DeclareTextComposite{\H}{T1}{u}{182}
637 \DeclareTextComposite{\r}{T1}{u}{183}
"B8 = 184
638 \DeclareTextComposite{\"}{T1}{y}{184}
639 \DeclareTextComposite{\'}{T1}{z}{185}
640 \DeclareTextComposite{\v}{T1}{z}{186}
641 \DeclareTextComposite{\.}{T1}{z}{187}
"C0 = 192
642 \DeclareTextComposite{\'}{T1}{A}{192}
643 \DeclareTextComposite{\'}{T1}{A}{193}
644 \DeclareTextComposite{\^}{T1}{A}{194}
645 \DeclareTextComposite{\~}{T1}{A}{195}
646 \DeclareTextComposite{\"}{T1}{A}{196}
647 \DeclareTextComposite{\r}{T1}{A}{197}
648 \DeclareTextComposite{\c}{T1}{C}{199}

```

"C8 = 200

```

649 \DeclareTextComposite{\'}{T1}{E}{200}
650 \DeclareTextComposite{\'}{T1}{E}{201}
651 \DeclareTextComposite{\^}{T1}{E}{202}
652 \DeclareTextComposite{\"}{T1}{E}{203}
653 \DeclareTextComposite{\'}{T1}{I}{204}
654 \DeclareTextComposite{\'}{T1}{I}{205}
655 \DeclareTextComposite{\^}{T1}{I}{206}
656 \DeclareTextComposite{\"}{T1}{I}{207}

```

"D0 = 208

```

657 \DeclareTextComposite{\~}{T1}{N}{209}
658 \DeclareTextComposite{\'}{T1}{O}{210}
659 \DeclareTextComposite{\'}{T1}{O}{211}
660 \DeclareTextComposite{\^}{T1}{O}{212}
661 \DeclareTextComposite{\~}{T1}{O}{213}
662 \DeclareTextComposite{\"}{T1}{O}{214}

```

"D8 = 216

```

663 \DeclareTextComposite{\'}{T1}{U}{217}
664 \DeclareTextComposite{\'}{T1}{U}{218}
665 \DeclareTextComposite{\^}{T1}{U}{219}
666 \DeclareTextComposite{\"}{T1}{U}{220}
667 \DeclareTextComposite{\'}{T1}{Y}{221}

```

"E0 = 224

```

668 \DeclareTextComposite{\'}{T1}{a}{224}
669 \DeclareTextComposite{\'}{T1}{a}{225}
670 \DeclareTextComposite{\^}{T1}{a}{226}
671 \DeclareTextComposite{\~}{T1}{a}{227}
672 \DeclareTextComposite{\"}{T1}{a}{228}
673 \DeclareTextComposite{\r}{T1}{a}{229}
674 \DeclareTextComposite{\c}{T1}{c}{231}

```

"E8 = 232

```

675 \DeclareTextComposite{\'}{T1}{e}{232}
676 \DeclareTextComposite{\'}{T1}{e}{233}
677 \DeclareTextComposite{\^}{T1}{e}{234}
678 \DeclareTextComposite{\"}{T1}{e}{235}
679 \DeclareTextComposite{\'}{T1}{i}{236}
680 \DeclareTextComposite{\'}{T1}{i}{236}
681 \DeclareTextComposite{\'}{T1}{i}{237}
682 \DeclareTextComposite{\'}{T1}{i}{237}
683 \DeclareTextComposite{\^}{T1}{i}{238}
684 \DeclareTextComposite{\^}{T1}{i}{238}
685 \DeclareTextComposite{\"}{T1}{i}{239}
686 \DeclareTextComposite{\"}{T1}{i}{239}

```

"F0 = 240

```

687 \DeclareTextComposite{\~}{T1}{n}{241}
688 \DeclareTextComposite{\'}{T1}{o}{242}
689 \DeclareTextComposite{\'}{T1}{o}{243}
690 \DeclareTextComposite{\^}{T1}{o}{244}
691 \DeclareTextComposite{\~}{T1}{o}{245}
692 \DeclareTextComposite{\"}{T1}{o}{246}

```

"F8 = 248

```

693 \DeclareTextComposite{\'}{T1}{u}{249}
694 \DeclareTextComposite{\'}{T1}{u}{250}
695 \DeclareTextComposite{\^}{T1}{u}{251}
696 \DeclareTextComposite{\"}{T1}{u}{252}
697 \DeclareTextComposite{\'}{T1}{y}{253}

698 \DeclareTextCompositeCommand{\k}{T1}{o}{\textogonekcentered{o}}
699 \DeclareTextCompositeCommand{\k}{T1}{O}{\textogonekcentered{O}}

700 \ifx\textcommaabove\undefined\else
701 \DeclareTextCompositeCommand{\c}{T1}{g}{\textcommaabove{g}}
702 \fi
703 \ifx\textcommabelow\undefined\else
704 \DeclareTextCompositeCommand{\c}{T1}{G}{\textcommabelow{G}}
705 \DeclareTextCompositeCommand{\c}{T1}{K}{\textcommabelow{K}}
706 \DeclareTextCompositeCommand{\c}{T1}{k}{\textcommabelow{k}}
707 \DeclareTextCompositeCommand{\c}{T1}{L}{\textcommabelow{L}}
708 \DeclareTextCompositeCommand{\c}{T1}{l}{\textcommabelow{l}}
709 \DeclareTextCompositeCommand{\c}{T1}{N}{\textcommabelow{N}}
710 \DeclareTextCompositeCommand{\c}{T1}{n}{\textcommabelow{n}}
711 \DeclareTextCompositeCommand{\c}{T1}{R}{\textcommabelow{R}}
712 \DeclareTextCompositeCommand{\c}{T1}{r}{\textcommabelow{r}}
713 \fi
714 \end{T1}

```

1.7 Definitions for the OMS encoding

The definitions for the ‘TeX math symbol’ (OMS) encoding. Even though this is meant to be a math font, it includes some of the standard L^AT_EX text symbols.

Declare the encoding.

```

715 \*OMS
716 \DeclareFontEncoding{OMS}{}{}

```

Declare the symbols. Note that slot 13 has in places been named \Orb: please root out and destroy this impolity wherever you find it!

```

717 \DeclareTextSymbol{\textasteriskcentered}{OMS}{3} % "03
718 \DeclareTextSymbol{\textbackslash}{OMS}{110} % "6E
719 \DeclareTextSymbol{\textbar}{OMS}{106} % "6A
720 \DeclareTextSymbol{\textbardbl}{OMS}{107} % "6B
721 \DeclareTextSymbol{\textbraceleft}{OMS}{102} % "66
722 \DeclareTextSymbol{\textbraceright}{OMS}{103} % "67
723 \DeclareTextSymbol{\textbullet}{OMS}{15} % "0F
724 \DeclareTextSymbol{\textdaggerdbl}{OMS}{122} % "7A
725 \DeclareTextSymbol{\textdagger}{OMS}{121} % "79
726 \DeclareTextSymbol{\textparagraph}{OMS}{123} % "7B
727 \DeclareTextSymbol{\textperiodcentered}{OMS}{1} % "01
728 \DeclareTextSymbol{\textsection}{OMS}{120} % "78
729 \DeclareTextSymbol{\textbigcircle}{OMS}{13} % "0D
730 \DeclareTextCommand{\textcircled}{OMS}[1]{\hmode@bgroup
731 \oalign{%
732 \hfil \raise .07ex\hbox {\upshape#1}\hfil \crcr
733 \char 13 % "0D

```

```

734     }%
735     \egroup}
736   </OMS>

```

1.8 Definitions for the OML encoding

The definitions for the ‘ $\mathrm{T}_{\mathrm{E}}\mathrm{X}$ math italic’ (OML) encoding. Even though this is meant to be a math font, it includes some of the standard $\mathrm{L}^{\mathrm{A}}\mathrm{T}_{\mathrm{E}}\mathrm{X}$ text symbols.

Declare the encoding.

```

737 <*OML>
738 \DeclareFontEncoding{OML}{-}{-}

```

Declare the symbols.

```

739 \DeclareTextSymbol{\textless}{OML}{'\<}
740 \DeclareTextSymbol{\textgreater}{OML}{'\>}
741 \DeclareTextAccent{\t}{OML}{127} % "7F
742 </OML>

```

1.9 Definitions for the OT4 encoding

These definitions are for the Polish extension to the ‘ $\mathrm{T}_{\mathrm{E}}\mathrm{X}$ text’ (OT1) encoding. This encoding was created by B. Jackowski and M. Ryćko for use with the Polish version of Computer Modern and Computer Concrete. In positions 0–127 it is identical to OT1 but it contains some additional characters in the upper half. The $\mathrm{L}^{\mathrm{A}}\mathrm{T}_{\mathrm{E}}\mathrm{X}$ support was developed by Mariusz Olko.

The PL fonts that use it are available as follows:

Metafont sources <ftp://ftp.gust.org.pl/TeX/language/polish/pl-mf.zip>;

Font files <ftp://ftp.gust.org.pl/TeX/language/polish/pl-tfm.zip>.

Declare the encoding.

```

743 <*OT4>
744 \DeclareFontEncoding{OT4}{-}{-}
745 \DeclareFontSubstitution{OT4}{cmr}{m}{n}

```

Declare the accents.

```

746 \DeclareTextAccent{"}{OT4}{127}
747 \DeclareTextAccent{'}{OT4}{19}
748 \DeclareTextAccent{.}{OT4}{95}
749 \DeclareTextAccent{=}{OT4}{22}
750 \DeclareTextAccent{~}{OT4}{94}
751 \DeclareTextAccent{'}{OT4}{18}
752 \DeclareTextAccent{~}{OT4}{126}
753 \DeclareTextAccent{H}{OT4}{125}
754 \DeclareTextAccent{u}{OT4}{21}
755 \DeclareTextAccent{v}{OT4}{20}
756 \DeclareTextAccent{r}{OT4}{23}

```

The ogonek accent is available only under a e A & E. But we have to provide some definition for \k. Some other accents have to be built by hand as in OT1:

```

757 \DeclareTextCommand{\k}{OT4}[1]{%
758   \TextSymbolUnavailable{\k{#1}}#1}

```


In these definitions we no longer use the helper function `\sh@ft` from `plain.tex` since that now has two incompatible definitions.

```

759 \DeclareTextCommand{\b}{OT4}[1]
760   {\hmode\bgroup\o@lign{\relax#1\cr\hidewidth\ltx@sh@ft{-3ex}%
761     \vbox to.2ex{\hbox{\char22}\vss}\hidewidth}\egroup}
762 \DeclareTextCommand{\c}{OT4}[1]
763   {\leavevmode\setbox\z@\hbox{#1}\ifdim\ht\z@=1ex\accent24 #1%
764     \else\oalign{\unhbox\z@\cr\hidewidth\char24\hidewidth}}\fi}
765 \DeclareTextCommand{\d}{OT4}[1]
766   {\hmode\bgroup
767     \o@lign{\relax#1\cr\hidewidth\ltx@sh@ft{-1ex}.\hidewidth}\egroup}

```

Declare the text symbols.

```

768 \DeclareTextSymbol{\AE}{OT4}{29}
769 \DeclareTextSymbol{\OE}{OT4}{30}
770 \DeclareTextSymbol{\O}{OT4}{31}
771 \DeclareTextSymbol{\L}{OT4}{138}
772 \DeclareTextSymbol{\ae}{OT4}{26}

773 \DeclareTextSymbol{\guillemetleft}{OT4}{174}
774 \DeclareTextSymbol{\guillemetright}{OT4}{175}
775 % old Adobe names
776 \DeclareTextSymbol{\guillemotleft}{OT4}{174}
777 \DeclareTextSymbol{\guillemotright}{OT4}{175}

778 \DeclareTextSymbol{\i}{OT4}{16}
779 \DeclareTextSymbol{\j}{OT4}{17}
780 \DeclareTextSymbol{\l}{OT4}{170}
781 \DeclareTextSymbol{\o}{OT4}{28}
782 \DeclareTextSymbol{\oe}{OT4}{27}
783 \DeclareTextSymbol{\quotedblbase}{OT4}{255}
784 \DeclareTextSymbol{\ss}{OT4}{25}
785 \DeclareTextSymbol{\textendash}{OT4}{124}
786 \DeclareTextSymbol{\textendash}{OT4}{123}
787 \DeclareTextSymbol{\textexclamdown}{OT4}{60}
788 %\DeclareTextSymbol{\textthyphenchar}{OT4}{'\-}
789 %\DeclareTextSymbol{\textthyphen}{OT4}{'\-}
790 \DeclareTextSymbol{\textquestiondown}{OT4}{62}
791 \DeclareTextSymbol{\textquotedblleft}{OT4}{92}
792 \DeclareTextSymbol{\textquotedblright}{OT4}{'\'}
793 \DeclareTextSymbol{\textquoteleft}{OT4}{'\'}
794 \DeclareTextSymbol{\textquoteright}{OT4}{'\'}

```

Definition for Å as in OT1:

```

795 \DeclareTextCompositeCommand{\r}{OT4}{A}
796   {\leavevmode\setbox\z@\hbox{!}\dimen@ht\z@\advance\dimen@-1ex%
797     \rlap{\raise.67\dimen@\hbox{\char23}}A}

```

In the OT4 encoding, £ and \$ share a slot.

```

798 \DeclareTextCommand{\textdollar}{OT4}{\hmode\bgroup
799   \ifdim \fontdimen\@ne\font >\z@
800     \slshape
801   \else
802     \upshape
803   \fi
804   \char'\$\egroup}

```

```

805 \DeclareTextCommand{\textsterling}{OT4}{\hmode@bgroup
806   \ifdim \fontdimen\@ne\font >\z@
807     \itshape
808   \else
809     \fontshape{ui}\selectfont
810   \fi
811   \char'\$ \egroup}

```

Declare the composites.

```

812 \DeclareTextComposite{\k}{OT4}{A}{129}
813 \DeclareTextComposite{\'}{OT4}{C}{130}
814 \DeclareTextComposite{\k}{OT4}{E}{134}
815 \DeclareTextComposite{\'}{OT4}{N}{139}
816 \DeclareTextComposite{\'}{OT4}{S}{145}
817 \DeclareTextComposite{\'}{OT4}{Z}{153}
818 \DeclareTextComposite{\.}{OT4}{Z}{155}
819 \DeclareTextComposite{\k}{OT4}{a}{161}
820 \DeclareTextComposite{\'}{OT4}{c}{162}
821 \DeclareTextComposite{\k}{OT4}{e}{166}
822 \DeclareTextComposite{\'}{OT4}{n}{171}
823 \DeclareTextComposite{\'}{OT4}{s}{177}
824 \DeclareTextComposite{\'}{OT4}{z}{185}
825 \DeclareTextComposite{\.}{OT4}{z}{187}
826 \DeclareTextComposite{\'}{OT4}{o}{211}
827 \DeclareTextComposite{\'}{OT4}{o}{243}
828 \end{OT4}

```

1.10 Definitions for the TS1 encoding

```

829 \langle *TS1 \rangle
830 \DeclareFontEncoding{TS1}{}{}
831 \DeclareFontSubstitution{TS1}{cmr}{m}{n}

```

Some accents have to be built by hand. Note that `\ooalign` and `\o@lign` must be inside a group.

```

832 \DeclareTextCommand{\capitalcedilla}{TS1}[1]
833   {\hmode@bgroup
834     \ooalign{\null#1\crrcr\hidewidth\char11\hidewidth}\egroup}
835 \DeclareTextCommand{\capitalogonek}{TS1}[1]
836   {\hmode@bgroup
837     \ooalign{\null#1\crrcr\hidewidth\char12\hidewidth}\egroup}

```

Accents for capital letters.

These commands can be used by the end user either directly or through definitions of the type

```
\DeclareTextCompositeCommand{\'}{T1}{X}{\capitalacute X}
```

None of the latter definitions are provided by default, since they are probably rarely used.

"00 = 0

```

838 \DeclareTextAccent{\capitalgrave}{TS1}{0}
839 \DeclareTextAccent{\capitalacute}{TS1}{1}
840 \DeclareTextAccent{\capitalcircumflex}{TS1}{2}
841 \DeclareTextAccent{\capitaltilde}{TS1}{3}
842 \DeclareTextAccent{\capitaldieresis}{TS1}{4}
843 \DeclareTextAccent{\capitalhungarumlaut}{TS1}{5}

```

```

844 \DeclareTextAccent{\capitalring}{TS1}{6}
845 \DeclareTextAccent{\capitalcaron}{TS1}{7}
"08 = 8
846 \DeclareTextAccent{\capitalbreve}{TS1}{8}
847 \DeclareTextAccent{\capitalmacron}{TS1}{9}
848 \DeclareTextAccent{\capitaldotaccent}{TS1}{10}

```

Tie accents.

The tie accent was borrowed from the `cmmi` font. The `tc` fonts now provide four tie accents, the first two are done in the classical way with asymmetric glyphs hanging out of their boxes; the new ties are centered in their boxes like all other accents. They need a name: please tell us if you know what to call them.

" =

```

849 \DeclareTextAccent{\t}{TS1}{26}
850 \DeclareTextAccent{\capitaltie}{TS1}{27}
851 \DeclareTextAccent{\newtie}{TS1}{28}
852 \DeclareTextAccent{\capitalnewtie}{TS1}{29}

```

Compound word marks.

The text companion fonts contain two compound word marks of different heights, one has `cap_height`, the other `asc_height`.

```

853 \DeclareTextSymbol{\textcapitalcompwordmark}{TS1}{23}
854 \DeclareTextSymbol{\textascendercompwordmark}{TS1}{31}

```

The text companion symbols.

```

855 \DeclareTextSymbol{\textquotestraightbase}{TS1}{13}

```

"10 = 16

```

856 \DeclareTextSymbol{\textquotestraightdblbase}{TS1}{18}
857 \DeclareTextSymbol{\texttwelveudash}{TS1}{21}
858 \DeclareTextSymbol{\textthreequartersemdash}{TS1}{22}

```

"18 = 24

```

859 \DeclareTextSymbol{\textleftarrow}{TS1}{24}
860 \DeclareTextSymbol{\textrightarrow}{TS1}{25}

```

"20 = 32

```

861 \DeclareTextSymbol{\textblank}{TS1}{32}
862 \DeclareTextSymbol{\textdollar}{TS1}{36}
863 \DeclareTextSymbol{\textquotesingle}{TS1}{39}

```

"28 = 40

The symbol `\textasteriskcentered` “*” is supposed to be always available in `TS1` and that is important as it is used in footnote symbols. However, in a few fonts it is missing even though they are otherwise fairly complete. We therefore use a rather elaborate method and check if the slot has a glyph and if not produce a poor man’s version by using a normal “*” slightly enlarged and somewhat lowered. The main application for this symbol is in footnote symbols and there it should produce a comparable size and show a similar placement.

```

864 %\DeclareTextSymbol{\textasteriskcentered}{TS1}{42} % that's wanted
865 \DeclareTextCommand \textasteriskcentered{TS1}{%    % and that's needed
866   \iffontchar\font 42 \char42 \else
867   \begingroup\fontencoding{T1}%
868     \fontsize

```

```

869      {\the\dimexpr1.3\dimexpr\fontsize pt\relax}%
870      {\font@baselineskip}%
871      \selectfont
872      \raisebox{-0.7ex}{\dimexpr\height-0.7ex}[0pt]{*}%
873    \endgroup
874    \fi
875  }

```

Note that '054 is a comma and '056 is a full stop: these make numbers using oldstyle digits easier to input.

```

876 \DeclareTextSymbol{\textdblhyphen}{TS1}{45}
877 \DeclareTextSymbol{\textfractionsolidus}{TS1}{47}

```

Oldstyle digits.

"30 = 48

```

878 \DeclareTextSymbol{\textzerooldstyle}{TS1}{48}
879 \DeclareTextSymbol{\textoneoldstyle}{TS1}{49}
880 \DeclareTextSymbol{\texttwooldstyle}{TS1}{50}
881 \DeclareTextSymbol{\textthreeoldstyle}{TS1}{51}
882 \DeclareTextSymbol{\textfouroldstyle}{TS1}{52}
883 \DeclareTextSymbol{\textfiveoldstyle}{TS1}{53}
884 \DeclareTextSymbol{\textsixoldstyle}{TS1}{54}
885 \DeclareTextSymbol{\textsevenoldstyle}{TS1}{55}

```

"38 = 56

```

886 \DeclareTextSymbol{\texteightoldstyle}{TS1}{56}
887 \DeclareTextSymbol{\textnineoldstyle}{TS1}{57}

```

More text companion symbols.

```

888 \DeclareTextSymbol{\textlangle}{TS1}{60}
889 \DeclareTextSymbol{\textminus}{TS1}{61}
890 \DeclareTextSymbol{\textrangle}{TS1}{62}

```

"48 = 72

```

891 \DeclareTextSymbol{\textmho}{TS1}{77}

```

The big circle is here to define the command `\textcircled`. Formerly it was taken from the `cmsy` font.

```

892 \DeclareTextSymbol{\textbigcircle}{TS1}{79}
893 \DeclareTextCommand{\textcircled}{TS1}[1]{\hmode\bgroup
894   \ooalign{%
895     \hfil \raise .07ex\hbox {\upshape#1}\hfil \cr
896     \char 79 % '117 = "4F
897   }%
898 \egroup}

```

More text companion symbols.

"50 = 80

```

899 \DeclareTextSymbol{\textohm}{TS1}{87}

```

"58 = 88

```

900 \DeclareTextSymbol{\textlbrackdbl}{TS1}{91}
901 \DeclareTextSymbol{\textrbrackdbl}{TS1}{93}
902 \DeclareTextSymbol{\textuparrow}{TS1}{94}
903 \DeclareTextSymbol{\textdownarrow}{TS1}{95}

```

"60 = 96

```
904 \DeclareTextSymbol{\textasciigrave}{TS1}{96}  
905 \DeclareTextSymbol{\textborn}{TS1}{98}  
906 \DeclareTextSymbol{\textdivorced}{TS1}{99}  
907 \DeclareTextSymbol{\textdied}{TS1}{100}
```

"68 = 104

```
908 \DeclareTextSymbol{\textleaf}{TS1}{108}  
909 \DeclareTextSymbol{\textmarried}{TS1}{109}  
910 \DeclareTextSymbol{\textmusicalnote}{TS1}{110}
```

"78 = 120

```
911 \DeclareTextSymbol{\texttildelow}{TS1}{126}
```

This glyph, `\textdblhyphenchar` is hanging, like the hyphenchar of the ec fonts.

```
912 \DeclareTextSymbol{\textdblhyphenchar}{TS1}{127}
```

"80 = 128

```
913 \DeclareTextSymbol{\textasciibreve}{TS1}{128}  
914 \DeclareTextSymbol{\textasciicaron}{TS1}{129}
```

This next glyph is *not* the same as `\textquotedbl`.

```
915 \DeclareTextSymbol{\textacutedbl}{TS1}{130}  
916 \DeclareTextSymbol{\textgravedbl}{TS1}{131}  
917 \DeclareTextSymbol{\textdagger}{TS1}{132}  
918 \DeclareTextSymbol{\textdaggerdbl}{TS1}{133}  
919 \DeclareTextSymbol{\textbardbl}{TS1}{134}  
920 \DeclareTextSymbol{\textperthousand}{TS1}{135}
```

"88 = 136

```
921 \DeclareTextSymbol{\textbullet}{TS1}{136}  
922 \DeclareTextSymbol{\textcelsius}{TS1}{137}  
923 \DeclareTextSymbol{\textdollaroldstyle}{TS1}{138}  
924 \DeclareTextSymbol{\textcentoldstyle}{TS1}{139}  
925 \DeclareTextSymbol{\textflorin}{TS1}{140}  
926 \DeclareTextSymbol{\textcolonmonetary}{TS1}{141}  
927 \DeclareTextSymbol{\textwon}{TS1}{142}  
928 \DeclareTextSymbol{\textnaira}{TS1}{143}
```

"90 = 144

```
929 \DeclareTextSymbol{\textguarani}{TS1}{144}  
930 \DeclareTextSymbol{\textpeso}{TS1}{145}  
931 \DeclareTextSymbol{\textlira}{TS1}{146}  
932 \DeclareTextSymbol{\textrecipe}{TS1}{147}  
933 \DeclareTextSymbol{\textinterrobang}{TS1}{148}  
934 \DeclareTextSymbol{\textinterrobangdown}{TS1}{149}  
935 \DeclareTextSymbol{\textdong}{TS1}{150}  
936 \DeclareTextSymbol{\texttrademark}{TS1}{151}
```

"98 = 152

```
937 \DeclareTextSymbol{\textpertenthousand}{TS1}{152}  
938 \DeclareTextSymbol{\textpilcrow}{TS1}{153}  
939 \DeclareTextSymbol{\textbaht}{TS1}{154}  
940 \DeclareTextSymbol{\textnumero}{TS1}{155}
```

This next name may change. For the following sign we know only a german name, which is abzüglich. The meaning is something like “commercial minus”. An ASCII ersatz is ./ (dot slash dot). The temporary English name is `\textdiscount`.

```

941 \DeclareTextSymbol{\textdiscount}{TS1}{156}
942 \DeclareTextSymbol{\textestimated}{TS1}{157}
943 \DeclareTextSymbol{\textopenbullet}{TS1}{158}
944 \DeclareTextSymbol{\textservicemark}{TS1}{159}

```

"A0 = 160

```

945 \DeclareTextSymbol{\textlquill}{TS1}{160}
946 \DeclareTextSymbol{\textrquill}{TS1}{161}
947 \DeclareTextSymbol{\textcent}{TS1}{162}
948 \DeclareTextSymbol{\textsterling}{TS1}{163}
949 \DeclareTextSymbol{\textcurrency}{TS1}{164}
950 \DeclareTextSymbol{\textyen}{TS1}{165}
951 \DeclareTextSymbol{\textbrokenbar}{TS1}{166}
952 \DeclareTextSymbol{\textsection}{TS1}{167}

```

"A8 = 168

```

953 \DeclareTextSymbol{\textasciidieresis}{TS1}{168}
954 \DeclareTextSymbol{\textcopyright}{TS1}{169}
955 \DeclareTextSymbol{\textordfeminine}{TS1}{170}
956 \DeclareTextSymbol{\textcopyleft}{TS1}{171}
957 \DeclareTextSymbol{\textlnot}{TS1}{172}

```

The meaning of the circled-P is “sound recording copyright”.

```

958 \DeclareTextSymbol{\textcircledP}{TS1}{173}
959 \DeclareTextSymbol{\textregistered}{TS1}{174}
960 \DeclareTextSymbol{\textasciimacron}{TS1}{175}

```

"B0 = 176

```

961 \DeclareTextSymbol{\textdegree}{TS1}{176}
962 \DeclareTextSymbol{\textpm}{TS1}{177}
963 \DeclareTextSymbol{\texttwosuperior}{TS1}{178}
964 \DeclareTextSymbol{\textthreesuperior}{TS1}{179}
965 \DeclareTextSymbol{\textasciicute}{TS1}{180}
966 \DeclareTextSymbol{\textmu}{TS1}{181} % micro sign
967 \DeclareTextSymbol{\textparagraph}{TS1}{182}
968 \DeclareTextSymbol{\textperiodcentered}{TS1}{183}

```

"B8 = 184

```

969 \DeclareTextSymbol{\textreferencemark}{TS1}{184}
970 \DeclareTextSymbol{\textonesuperior}{TS1}{185}
971 \DeclareTextSymbol{\textordmasculine}{TS1}{186}
972 \DeclareTextSymbol{\textsurd}{TS1}{187}
973 \DeclareTextSymbol{\textonequarter}{TS1}{188}
974 \DeclareTextSymbol{\textonehalf}{TS1}{189}
975 \DeclareTextSymbol{\textthreequarters}{TS1}{190}
976 \DeclareTextSymbol{\texteuro}{TS1}{191}

```

"E0 = 208

```

977 \DeclareTextSymbol{\texttimes}{TS1}{214}

```

"F0 = 240

```

978 \DeclareTextSymbol{\textdiv}{TS1}{246}
979 \</TS1>

```

1.11 Definitions for the TU encoding

The TU encoding was originally introduced in the contributed package `fontspec` as a Unicode encoding for XeTeX and LuaTeX.

Normally for these engines, the input consists of Unicode characters encoded in UTF-8. There is therefore little need to use the traditional (ASCII) encoding-specific commands

However, sometimes (e.g. for backwards compatibility) it can be useful to access these Unicode characters via such ASCII-based markup. The commands provided here cover the characters in the T1 and TS1 encodings, but specified in Unicode position. Almost all the command names have been mechanically extracted from the `inputenc` UTF-8 support, which is essentially doing a reverse mapping from UTF-8 data to L^AT_EX LICR commands.

A few additional names for character which were supported in the original `fontspec` version of this file have also been added, even though they are not currently in the default `inputenc` UTF-8 declarations.

```
980 ⟨*TU⟩
```

In the base interface the Unicode encoding is always known as TU. But we parameterize the encoding name to allow for modelling differences in Unicode support by different fonts.

```
981 \providecommand\UnicodeEncodingName{TU}
```

As the Unicode encoding, TU, is only currently available with XeTeX or LuaTeX, we detect these engines first, and make adjustments for the differing font loading syntax. For other engines, we issue a warning then abort this file, switching back to T1 encoding.

```
982 \begingroup\expandafter\expandafter\expandafter\endgroup
```

```
983 \expandafter\ifx\csname XeTeXrevision\endcsname\relax
```

```
984 \begingroup\expandafter\expandafter\expandafter\endgroup
```

```
985 \expandafter\ifx\csname directlua\endcsname\relax
```

Not LuaTeX or XeTeX, abort with a warning.

```
986 \PackageWarningNoLine{fontenc}
```

```
987 {\UnicodeEncodingName\space
```

```
988 encoding is only available with XeTeX and LuaTeX.\MessageBreak
```

```
989 Defaulting to T1 encoding}
```

```
990 \def\encodingdefault{T1}
```

```
991 \expandafter\expandafter\expandafter\endinput
```

```
992 \else
```

LuaTeX. For LuaTeX 1.10+, define a Lua function to disable any handing by the font code. Otherwise we reload the font without TeX ligatures.

```
993 \def\UnicodeFontTeXLigatures{+tlig;}
```

```
994 \ifnum\luatexversion<110
```

```
995 \def\reserved@a#1{%
```

```
996 \def\@remove@tlig##1{\@remove@tlig@##1\@nil#1\@nil\relax}
```

```
997 \def\@remove@tlig@##1#1{\@remove@tlig@##1}}
```

```
998 \edef\reserved@b{\detokenize{+tlig;}}
```

```
999 \expandafter\reserved@a\expandafter{\reserved@b}
```

```
1000 \def\@remove@tlig@##1\@nil#2\relax{#1}
```

```

1001 \def\remove@tlig#1{%
1002 \begingroup
1003 \font\remove@tlig
1004 \expandafter\@remove@tlig\expandafter{\fontname\font}%
1005 \remove@tlig
1006 \char#1\relax
1007 \endgroup
1008 }
1009 \else
1010 \newluafunction\@remove@tlig@@@

```

Now we can define the function. Mostly we just have to insert a protected glyph node, which is a glyph node with subtype 256. But we have to keep track of the current mode to avoid inserting the glyph into a vlist.

```

1011 \now@and@everyjob{\directlua{
1012   local rawchar_func = token.create'\@remove@tlig@@@'.index
1013   local forcehmode = tex.forcehmode
1014   local put_next = token.put_next
1015   local glyph_id = node.id'glyph'
1016   local rawchar_token = token.new(rawchar_func, token.command_id'lua_call')
1017   lua.get_functions_table()[rawchar_func] = function()
1018     local mode = tex.nest.top.mode
1019     if mode == 1 or mode == -1 then
1020       put_next(rawchar_token)
1021       return forcehmode(true)
1022     end
1023     local n = node.new(glyph_id, 256)
1024     n.font = font.current()
1025     n.char = token.scan_int()
1026     return node.write(n)
1027   end
1028   token.set_lua('\@remove@tlig@@@', rawchar_func, 'global', 'protected')
1029 }}

```

Now `\remove@tlig` can be implemented almost as in XeTeX.

```

1030 \def\remove@tlig#1{\@remove@tlig@@@#1\relax}
1031 \fi
1032 \fi
1033 \else
1034 XeTeX
1035 \def\UnicodeFontTeXLigatures{mapping=tex-text;}
1036 \def\remove@tlig#1{\XeTeXglyph\numexpr\XeTeXcharglyph#1\relax}
1037 \fi
1038 \def\UnicodeFontFile#1#2{"[#1]:#2"}
1039 \def\UnicodeFontName#1#2{"#1:#2"}
1040 Declare the encoding
1041 \DeclareFontEncoding\UnicodeEncodingName{}{}
1042 Declare accent command to use a postpended combining character rather than the
TeX \accent primitive
1043 \def\add@unicode@accent#1#2{%
1044 \if\relax\detokenize{#2}\relax~^a0\else#2\fi
1045 \char#1\relax}

```



```

1043 \def\DeclareUnicodeAccent#1#2#3{%
1044   \DeclareTextCommand{#1}{#2}{\add@unicode@accent{#3}}%
1045 }

    Wrapper around \DeclareTextCompositeCommand that uses the declared composite
    if it exists in the current font or falls back to the default definition for the TU accent if
    not.

1046 {
1047   \catcode\z@=11\relax
1048   \gdef\DeclareUnicodeComposite#1#2#3{%
1049     \def\reserved@a##1##2{%
1050       \DeclareTextCompositeCommand#1\UnicodeEncodingName{#2}{%
1051         \iffontchar\font#3 ##2%
1052         \else ##1\fi}}%
1053     \expandafter\expandafter\expandafter\extract@default@composite
1054     \csname\UnicodeEncodingName\string#1\endcsname{#2}\@nil
1055     \bgroup
1056       \lccode\z@#3 %
1057       \lowercase{\egroup
1058         \expandafter\reserved@a\expandafter{\reserved@b}{^^@}}}%
1059   }

1060   \def\extract@default@composite#1{%
1061     \ifx\@text@composite#1%
1062       \expandafter\extract@default@composite@a
1063     \else
1064       \expandafter\extract@default@composite@b\expandafter#1%
1065     \fi}

1066   \def\extract@default@composite@a#1\@text@composite#2\@nil{%
1067     \def\reserved@b{#2}}
1068   \def\extract@default@composite@b#1#2\@nil{%
1069     \def\reserved@b{#1#2}}

1070   \DeclareTextCommand\textquotesingle \UnicodeEncodingName{%
1071     \remove@tlig{"0027}}
1072   \DeclareTextCommand\textasciigrave \UnicodeEncodingName{%
1073     \remove@tlig{"0060}}
1074   \DeclareTextCommand\textquotedbl \UnicodeEncodingName{%
1075     \remove@tlig{"0022}}

1076   \DeclareTextSymbol{\textdollar} \UnicodeEncodingName{"0024}
1077   \DeclareTextSymbol{\textless} \UnicodeEncodingName{"003C}
1078   \DeclareTextSymbol{\textgreater} \UnicodeEncodingName{"003E}
1079   \DeclareTextSymbol{\textbackslash} \UnicodeEncodingName{"005C}
1080   \DeclareTextSymbol{\textasciicircum} \UnicodeEncodingName{"005E}
1081   \DeclareTextSymbol{\textunderscore} \UnicodeEncodingName{"005F}
1082   \DeclareTextSymbol{\textbraceleft} \UnicodeEncodingName{"007B}
1083   \DeclareTextSymbol{\textbar} \UnicodeEncodingName{"007C}
1084   \DeclareTextSymbol{\textbraceright} \UnicodeEncodingName{"007D}
1085   \DeclareTextSymbol{\textasciitilde} \UnicodeEncodingName{"007E}
1086   \DeclareTextSymbol{\textexclamdown} \UnicodeEncodingName{"00A1}
1087   \DeclareTextSymbol{\textcent} \UnicodeEncodingName{"00A2}
1088   \DeclareTextSymbol{\textsterling} \UnicodeEncodingName{"00A3}
1089   \DeclareTextSymbol{\textcurrency} \UnicodeEncodingName{"00A4}
1090   \DeclareTextSymbol{\textyen} \UnicodeEncodingName{"00A5}

```

1091	\DeclareTextSymbol{\textbrokenbar}	\UnicodeEncodingName{"00A6}
1092	\DeclareTextSymbol{\textsection}	\UnicodeEncodingName{"00A7}
1093	\DeclareTextSymbol{\textasciidieresis}	\UnicodeEncodingName{"00A8}
1094	\DeclareTextSymbol{\textcopyright}	\UnicodeEncodingName{"00A9}
1095	\DeclareTextSymbol{\textordfeminine}	\UnicodeEncodingName{"00AA}
1096	\DeclareTextSymbol{\guillemetleft}	\UnicodeEncodingName{"00AB}
1097	% old Adobe name	
1098	\DeclareTextSymbol{\guillemotleft}	\UnicodeEncodingName{"00AB}
1099	\DeclareTextSymbol{\textlnot}	\UnicodeEncodingName{"00AC}
1100	\DeclareTextSymbol{\textregistered}	\UnicodeEncodingName{"00AE}
1101	\DeclareTextSymbol{\textasciimacron}	\UnicodeEncodingName{"00AF}
1102	\DeclareTextSymbol{\textdegree}	\UnicodeEncodingName{"00B0}
1103	\DeclareTextSymbol{\textpm}	\UnicodeEncodingName{"00B1}
1104	\DeclareTextSymbol{\texttwosuperior}	\UnicodeEncodingName{"00B2}
1105	\DeclareTextSymbol{\textthreesuperior}	\UnicodeEncodingName{"00B3}
1106	\DeclareTextSymbol{\textasciacute}	\UnicodeEncodingName{"00B4}
1107	\DeclareTextSymbol{\textmu}	\UnicodeEncodingName{"00B5}
1108	\DeclareTextSymbol{\textparagraph}	\UnicodeEncodingName{"00B6}
1109	\DeclareTextSymbol{\textperiodcentered}	\UnicodeEncodingName{"00B7}
1110	\DeclareTextSymbol{\textonesuperior}	\UnicodeEncodingName{"00B9}
1111	\DeclareTextSymbol{\textordmasculine}	\UnicodeEncodingName{"00BA}
1112	\DeclareTextSymbol{\guillemetright}	\UnicodeEncodingName{"00BB}
1113	% old Adobe name	
1114	\DeclareTextSymbol{\guillemotright}	\UnicodeEncodingName{"00BB}
1115	\DeclareTextSymbol{\textonequarter}	\UnicodeEncodingName{"00BC}
1116	\DeclareTextSymbol{\textonehalf}	\UnicodeEncodingName{"00BD}
1117	\DeclareTextSymbol{\textthreequarters}	\UnicodeEncodingName{"00BE}
1118	\DeclareTextSymbol{\textquestiondown}	\UnicodeEncodingName{"00BF}
1119	\DeclareTextSymbol{\AE}	\UnicodeEncodingName{"00C6}
1120	\DeclareTextSymbol{\DH}	\UnicodeEncodingName{"00D0}
1121	\DeclareTextSymbol{\texttmes}	\UnicodeEncodingName{"00D7}
1122	\DeclareTextSymbol{\O}	\UnicodeEncodingName{"00D8}
1123	\DeclareTextSymbol{\TH}	\UnicodeEncodingName{"00DE}
1124	\DeclareTextSymbol{\ss}	\UnicodeEncodingName{"00DF}
1125	\DeclareTextSymbol{\ae}	\UnicodeEncodingName{"00E6}
1126	\DeclareTextSymbol{\dh}	\UnicodeEncodingName{"00F0}
1127	\DeclareTextSymbol{\textdiv}	\UnicodeEncodingName{"00F7}
1128	\DeclareTextSymbol{\o}	\UnicodeEncodingName{"00F8}
1129	\DeclareTextSymbol{\th}	\UnicodeEncodingName{"00FE}
1130	\DeclareTextSymbol{\DJ}	\UnicodeEncodingName{"0110}
1131	\DeclareTextSymbol{\dj}	\UnicodeEncodingName{"0111}
1132	\DeclareTextSymbol{\i}	\UnicodeEncodingName{"0131}
1133	\DeclareTextSymbol{\IJ}	\UnicodeEncodingName{"0132}
1134	\DeclareTextSymbol{\ij}	\UnicodeEncodingName{"0133}
1135	\DeclareTextSymbol{\L}	\UnicodeEncodingName{"0141}
1136	\DeclareTextSymbol{\l}	\UnicodeEncodingName{"0142}
1137	\DeclareTextSymbol{\NG}	\UnicodeEncodingName{"014A}
1138	\DeclareTextSymbol{\ng}	\UnicodeEncodingName{"014B}
1139	\DeclareTextSymbol{\OE}	\UnicodeEncodingName{"0152}
1140	\DeclareTextSymbol{\oe}	\UnicodeEncodingName{"0153}
1141	\DeclareTextSymbol{\textflorin}	\UnicodeEncodingName{"0192}
1142	\DeclareTextSymbol{\j}	\UnicodeEncodingName{"0237}

```

1143 \DeclareTextSymbol{\textasciicaron} \UnicodeEncodingName{"02C7}
1144 \DeclareTextSymbol{\textasciibreve} \UnicodeEncodingName{"02D8}
1145 \DeclareTextSymbol{\textacutedbl} \UnicodeEncodingName{"02DD}
1146 \DeclareTextSymbol{\textgravedbl} \UnicodeEncodingName{"02F5}
1147 \DeclareTextSymbol{\texttildelow} \UnicodeEncodingName{"02F7}
1148 \DeclareTextSymbol{\textbaht} \UnicodeEncodingName{"0E3F}
1149 \DeclareTextSymbol{\SS} \UnicodeEncodingName{"1E9E}
1150 \DeclareTextSymbol{\textcompwordmark} \UnicodeEncodingName{"200C}

1151 %\DeclareTextSymbol{\textnonbreakinghyphen} \UnicodeEncodingName{"2011}
1152 %\DeclareTextSymbol{\textfiguredash} \UnicodeEncodingName{"2012}
1153 \DeclareTextSymbol{\textendash} \UnicodeEncodingName{"2013}
1154 \DeclareTextSymbol{\textemdash} \UnicodeEncodingName{"2014}
1155 %\DeclareTextSymbol{\texthorizontalbar} \UnicodeEncodingName{"2015}

Unfortunately some fonts do not implement "2011, "2012 and/or "2015 (including the
LATEX default fonts for Unicode engines) so we provide some approximations if the glyph
is missing, like we do for OT1 and T1.

The \nobreak\hskip\z@ is there to prevent a break after the hyphen but allow later
breaks in the remainder of the word.

1156 \DeclareTextCommand{\textnonbreakinghyphen} \UnicodeEncodingName
1157 {\iffontchar\font "2011 \char "2011 \else \mbox{-}\nobreak\hskip\z@ \fi}
1158 \DeclareTextCommand{\textfiguredash} \UnicodeEncodingName
1159 {\iffontchar\font "2012 \char "2012 \else \char "2013 \fi}
1160 \DeclareTextCommand{\texthorizontalbar} \UnicodeEncodingName
1161 {\iffontchar\font "2015 \char "2015 \else \char "2014 \fi}

1162 \DeclareTextSymbol{\textbardbl} \UnicodeEncodingName{"2016}
1163 \DeclareTextSymbol{\textquoteleft} \UnicodeEncodingName{"2018}
1164 \DeclareTextSymbol{\textquoteright} \UnicodeEncodingName{"2019}
1165 \DeclareTextSymbol{\quotesinglbase} \UnicodeEncodingName{"201A}
1166 \DeclareTextSymbol{\textquotedblleft} \UnicodeEncodingName{"201C}
1167 \DeclareTextSymbol{\textquotedblright} \UnicodeEncodingName{"201D}
1168 \DeclareTextSymbol{\quotedblbase} \UnicodeEncodingName{"201E}
1169 \DeclareTextSymbol{\textdagger} \UnicodeEncodingName{"2020}
1170 \DeclareTextSymbol{\textdaggerdbl} \UnicodeEncodingName{"2021}
1171 \DeclareTextSymbol{\textbullet} \UnicodeEncodingName{"2022}
1172 \DeclareTextSymbol{\textellipsis} \UnicodeEncodingName{"2026}
1173 \DeclareTextSymbol{\textperthousand} \UnicodeEncodingName{"2030}
1174 \DeclareTextSymbol{\textpertenthousand} \UnicodeEncodingName{"2031}
1175 \DeclareTextSymbol{\guilsinglleft} \UnicodeEncodingName{"2039}
1176 \DeclareTextSymbol{\guilsinglright} \UnicodeEncodingName{"203A}
1177 \DeclareTextSymbol{\textreferencemark} \UnicodeEncodingName{"203B}
1178 \DeclareTextSymbol{\textinterrobang} \UnicodeEncodingName{"203D}
1179 \DeclareTextSymbol{\textfractionsolidus} \UnicodeEncodingName{"2044}
1180 \DeclareTextSymbol{\textlquill} \UnicodeEncodingName{"2045}
1181 \DeclareTextSymbol{\textrquill} \UnicodeEncodingName{"2046}
1182 \DeclareTextSymbol{\textdiscount} \UnicodeEncodingName{"2052}
1183 \DeclareTextSymbol{\textcolonmonetary} \UnicodeEncodingName{"20A1}
1184 \DeclareTextSymbol{\textlira} \UnicodeEncodingName{"20A4}
1185 \DeclareTextSymbol{\textnaira} \UnicodeEncodingName{"20A6}
1186 \DeclareTextSymbol{\textwon} \UnicodeEncodingName{"20A9}
1187 \DeclareTextSymbol{\textdong} \UnicodeEncodingName{"20AB}
1188 \DeclareTextSymbol{\texteuro} \UnicodeEncodingName{"20AC}
1189 \DeclareTextSymbol{\textpeso} \UnicodeEncodingName{"20B1}

```

```

1190 \DeclareTextSymbol{\textcelsius} \UnicodeEncodingName{"2103}
1191 \DeclareTextSymbol{\textnumero} \UnicodeEncodingName{"2116}
1192 \DeclareTextSymbol{\textcircledP} \UnicodeEncodingName{"2117}
1193 \DeclareTextSymbol{\textrecipe} \UnicodeEncodingName{"211E}
1194 \DeclareTextSymbol{\textservicemark} \UnicodeEncodingName{"2120}
1195 \DeclareTextSymbol{\texttrademark} \UnicodeEncodingName{"2122}
1196 \DeclareTextSymbol{\textohm} \UnicodeEncodingName{"2126}
1197 \DeclareTextSymbol{\textmho} \UnicodeEncodingName{"2127}
1198 \DeclareTextSymbol{\textestimated} \UnicodeEncodingName{"212E}
1199 \DeclareTextSymbol{\textleftarrow} \UnicodeEncodingName{"2190}
1200 \DeclareTextSymbol{\textuparrow} \UnicodeEncodingName{"2191}
1201 \DeclareTextSymbol{\textrightarrow} \UnicodeEncodingName{"2192}
1202 \DeclareTextSymbol{\textdownarrow} \UnicodeEncodingName{"2193}
1203 \DeclareTextSymbol{\textminus} \UnicodeEncodingName{"2212}
1204

```

```

1205 \DeclareTextSymbol{\Hwithstroke} \UnicodeEncodingName{"0126}
1206 \DeclareTextSymbol{\hwithstroke} \UnicodeEncodingName{"0127}

```

Not all fonts have U+2217 but using U+002A requires some adjustment.

```

1207 \DeclareTextCommand{\textasteriskcentered}\UnicodeEncodingName{%
1208   \iffontchar\font"2217 \char"2217 \else
1209     \begingroup
1210       \fontsize
1211       {\the\dimexpr1.3\dimexpr\fontsize pt\relax}%
1212       {\f@baselineskip}%
1213       \selectfont
1214       \raisebox{-0.7ex}{\dimexpr\height-0.7ex}[0pt]{*}%
1215     \endgroup
1216   \fi
1217 }
1218 \DeclareTextSymbol{\textsurd} \UnicodeEncodingName{"221A}
1219 \DeclareTextSymbol{\textlangle} \UnicodeEncodingName{"2329}
1220 \DeclareTextSymbol{\textrangle} \UnicodeEncodingName{"232A}
1221 \DeclareTextSymbol{\textblank} \UnicodeEncodingName{"2422}
1222 \DeclareTextSymbol{\textvisiblespace} \UnicodeEncodingName{"2423}
1223 \DeclareTextSymbol{\textopenbullet} \UnicodeEncodingName{"25E6}
1224 \DeclareTextSymbol{\textbigcircle} \UnicodeEncodingName{"25EF}
1225 \DeclareTextSymbol{\textmusicalnote} \UnicodeEncodingName{"266A}
1226 \DeclareTextSymbol{\textmarried} \UnicodeEncodingName{"26AD}
1227 \DeclareTextSymbol{\textdivorced} \UnicodeEncodingName{"26AE}
1228 \DeclareTextSymbol{\textinterrobangdown} \UnicodeEncodingName{"2E18}

```

Accents must be declared before the composites that use them.

```

1229 \DeclareUnicodeAccent{\` } \UnicodeEncodingName{"0300}
1230 \DeclareUnicodeAccent{\' } \UnicodeEncodingName{"0301}
1231 \DeclareUnicodeAccent{\^ } \UnicodeEncodingName{"0302}
1232 \DeclareUnicodeAccent{\~ } \UnicodeEncodingName{"0303}
1233 \DeclareUnicodeAccent{\=} \UnicodeEncodingName{"0304}
1234 \DeclareUnicodeAccent{\u } \UnicodeEncodingName{"0306}
1235 \DeclareUnicodeAccent{\. } \UnicodeEncodingName{"0307}
1236 \DeclareUnicodeAccent{\" } \UnicodeEncodingName{"0308}
1237 \DeclareUnicodeAccent{\r } \UnicodeEncodingName{"030A}
1238 \DeclareUnicodeAccent{\H } \UnicodeEncodingName{"030B}

```

```

1239 \DeclareUnicodeAccent{\v}          \UnicodeEncodingName{"030C}
1240 \DeclareUnicodeAccent{\b}          \UnicodeEncodingName{"0332}
1241 \DeclareUnicodeAccent{\d}          \UnicodeEncodingName{"0323}
1242 \DeclareUnicodeAccent{\c}          \UnicodeEncodingName{"0327}
1243 \DeclareUnicodeAccent{\k}          \UnicodeEncodingName{"0328}
1244 \DeclareTextCommand\textcommabelow \UnicodeEncodingName[1]
1245   {\hmode\bgroup\oalign{\null#1\cr\hidewidth\raise-.31ex
1246     \hbox{\check@mathfonts\fontsize\ssf@size\z@
1247       \math@fontsfalse\selectfont,}\hidewidth}\egroup}

1248 \DeclareUnicodeComposite{\^}        {}{"005E}
1249 \DeclareUnicodeComposite{\~}        {}{"007E}

1250 \DeclareUnicodeComposite{\'}        {A}{"00C0}
1251 \DeclareUnicodeComposite{\'}        {A}{"00C1}
1252 \DeclareUnicodeComposite{\^}        {A}{"00C2}
1253 \DeclareUnicodeComposite{\~}        {A}{"00C3}
1254 \DeclareUnicodeComposite{"}         {A}{"00C4}
1255 \DeclareUnicodeComposite{\r}        {A}{"00C5}
1256 \DeclareUnicodeComposite{\c}        {C}{"00C7}
1257 \DeclareUnicodeComposite{\'}        {E}{"00C8}
1258 \DeclareUnicodeComposite{\'}        {E}{"00C9}
1259 \DeclareUnicodeComposite{\^}        {E}{"00CA}
1260 \DeclareUnicodeComposite{"}         {E}{"00CB}
1261 \DeclareUnicodeComposite{\'}        {I}{"00CC}
1262 \DeclareUnicodeComposite{\'}        {I}{"00CD}
1263 \DeclareUnicodeComposite{\^}        {I}{"00CE}
1264 \DeclareUnicodeComposite{"}         {I}{"00CF}
1265 \DeclareUnicodeComposite{\~}        {N}{"00D1}
1266 \DeclareUnicodeComposite{\'}        {O}{"00D2}
1267 \DeclareUnicodeComposite{\'}        {O}{"00D3}
1268 \DeclareUnicodeComposite{\^}        {O}{"00D4}
1269 \DeclareUnicodeComposite{\~}        {O}{"00D5}
1270 \DeclareUnicodeComposite{"}         {O}{"00D6}
1271 \DeclareUnicodeComposite{\'}        {U}{"00D9}
1272 \DeclareUnicodeComposite{\'}        {U}{"00DA}
1273 \DeclareUnicodeComposite{\^}        {U}{"00DB}
1274 \DeclareUnicodeComposite{"}         {U}{"00DC}
1275 \DeclareUnicodeComposite{\'}        {Y}{"00DD}
1276 \DeclareUnicodeComposite{\'}        {a}{"00E0}
1277 \DeclareUnicodeComposite{\'}        {a}{"00E1}
1278 \DeclareUnicodeComposite{\^}        {a}{"00E2}
1279 \DeclareUnicodeComposite{\~}        {a}{"00E3}
1280 \DeclareUnicodeComposite{"}         {a}{"00E4}
1281 \DeclareUnicodeComposite{\r}        {a}{"00E5}
1282 \DeclareUnicodeComposite{\c}        {c}{"00E7}
1283 \DeclareUnicodeComposite{\'}        {e}{"00E8}
1284 \DeclareUnicodeComposite{\'}        {e}{"00E9}
1285 \DeclareUnicodeComposite{\^}        {e}{"00EA}
1286 \DeclareUnicodeComposite{"}         {e}{"00EB}
1287 \DeclareUnicodeComposite{\'}        \i {"00EC}
1288 \DeclareUnicodeComposite{\'}        {i}{"00EC}
1289 \DeclareUnicodeComposite{\'}        \i {"00ED}
1290 \DeclareUnicodeComposite{\'}        {i}{"00ED}
1291 \DeclareUnicodeComposite{\^}        \i {"00EE}

```

1292	\DeclareUnicodeComposite{\~}	{i}{"00EE}
1293	\DeclareUnicodeComposite{"}	\i {"00EF}
1294	\DeclareUnicodeComposite{"}	{i}{"00EF}
1295	\DeclareUnicodeComposite{\~}	{n}{"00F1}
1296	\DeclareUnicodeComposite{'}	{o}{"00F2}
1297	\DeclareUnicodeComposite{'}	{o}{"00F3}
1298	\DeclareUnicodeComposite{\^}	{o}{"00F4}
1299	\DeclareUnicodeComposite{\~}	{o}{"00F5}
1300	\DeclareUnicodeComposite{"}	{o}{"00F6}
1301	\DeclareUnicodeComposite{'}	{u}{"00F9}
1302	\DeclareUnicodeComposite{'}	{u}{"00FA}
1303	\DeclareUnicodeComposite{\^}	{u}{"00FB}
1304	\DeclareUnicodeComposite{"}	{u}{"00FC}
1305	\DeclareUnicodeComposite{'}	{y}{"00FD}
1306	\DeclareUnicodeComposite{"}	{y}{"00FF}
1307	\DeclareUnicodeComposite{\=}	{A}{"0100}
1308	\DeclareUnicodeComposite{\=}	{a}{"0101}
1309	\DeclareUnicodeComposite{u}	{A}{"0102}
1310	\DeclareUnicodeComposite{u}	{a}{"0103}
1311	\DeclareUnicodeComposite{k}	{A}{"0104}
1312	\DeclareUnicodeComposite{k}	{a}{"0105}
1313	\DeclareUnicodeComposite{'}	{C}{"0106}
1314	\DeclareUnicodeComposite{'}	{c}{"0107}
1315	\DeclareUnicodeComposite{\^}	{C}{"0108}
1316	\DeclareUnicodeComposite{\^}	{c}{"0109}
1317	\DeclareUnicodeComposite{.}	{C}{"010A}
1318	\DeclareUnicodeComposite{.}	{c}{"010B}
1319	\DeclareUnicodeComposite{v}	{C}{"010C}
1320	\DeclareUnicodeComposite{v}	{c}{"010D}
1321	\DeclareUnicodeComposite{v}	{D}{"010E}
1322	\DeclareUnicodeComposite{v}	{d}{"010F}
1323	\DeclareUnicodeComposite{\=}	{E}{"0112}
1324	\DeclareUnicodeComposite{\=}	{e}{"0113}
1325	\DeclareUnicodeComposite{u}	{E}{"0114}
1326	\DeclareUnicodeComposite{u}	{e}{"0115}
1327	\DeclareUnicodeComposite{.}	{E}{"0116}
1328	\DeclareUnicodeComposite{.}	{e}{"0117}
1329	\DeclareUnicodeComposite{k}	{E}{"0118}
1330	\DeclareUnicodeComposite{k}	{e}{"0119}
1331	\DeclareUnicodeComposite{v}	{E}{"011A}
1332	\DeclareUnicodeComposite{v}	{e}{"011B}
1333	\DeclareUnicodeComposite{\^}	{G}{"011C}
1334	\DeclareUnicodeComposite{\^}	{g}{"011D}
1335	\DeclareUnicodeComposite{u}	{G}{"011E}
1336	\DeclareUnicodeComposite{u}	{g}{"011F}
1337	\DeclareUnicodeComposite{.}	{G}{"0120}
1338	\DeclareUnicodeComposite{.}	{g}{"0121}
1339	\DeclareUnicodeComposite{c}	{G}{"0122}
1340	\DeclareUnicodeComposite{c}	{g}{"0123}
1341	\DeclareUnicodeComposite{\^}	{H}{"0124}
1342	\DeclareUnicodeComposite{\^}	{h}{"0125}
1343	\DeclareUnicodeComposite{\~}	{I}{"0128}
1344	\DeclareUnicodeComposite{\~}	\i {"0129}
1345	\DeclareUnicodeComposite{\~}	{i}{"0129}

1346	\DeclareUnicodeComposite{=}	{I}-{ "012A}
1347	\DeclareUnicodeComposite{=}	\i { "012B}
1348	\DeclareUnicodeComposite{=}	{i}-{ "012B}
1349	\DeclareUnicodeComposite{\u}	{I}-{ "012C}
1350	\DeclareUnicodeComposite{\u}	\i { "012D}
1351	\DeclareUnicodeComposite{\u}	{i}-{ "012D}
1352	\DeclareUnicodeComposite{\k}	{I}-{ "012E}
1353	\DeclareUnicodeComposite{\k}	\i { "012F}
1354	\DeclareUnicodeComposite{\k}	{i}-{ "012F}
1355	\DeclareUnicodeComposite{.}	{I}-{ "0130}
1356	\DeclareUnicodeComposite{^}	{J}-{ "0134}
1357	\DeclareUnicodeComposite{^}	\j { "0135}
1358	\DeclareUnicodeComposite{^}	{j}-{ "0135}
1359	\DeclareUnicodeComposite{c}	{K}-{ "0136}
1360	\DeclareUnicodeComposite{c}	{k}-{ "0137}
1361	\DeclareUnicodeComposite{'}	{L}-{ "0139}
1362	\DeclareUnicodeComposite{'}	{l}-{ "013A}
1363	\DeclareUnicodeComposite{c}	{L}-{ "013B}
1364	\DeclareUnicodeComposite{c}	{l}-{ "013C}
1365	\DeclareUnicodeComposite{v}	{L}-{ "013D}
1366	\DeclareUnicodeComposite{v}	{l}-{ "013E}
1367	\DeclareUnicodeComposite{'}	{N}-{ "0143}
1368	\DeclareUnicodeComposite{'}	{n}-{ "0144}
1369	\DeclareUnicodeComposite{c}	{N}-{ "0145}
1370	\DeclareUnicodeComposite{c}	{n}-{ "0146}
1371	\DeclareUnicodeComposite{v}	{N}-{ "0147}
1372	\DeclareUnicodeComposite{v}	{n}-{ "0148}
1373	\DeclareUnicodeComposite{=}	{O}-{ "014C}
1374	\DeclareUnicodeComposite{=}	{o}-{ "014D}
1375	\DeclareUnicodeComposite{\u}	{O}-{ "014E}
1376	\DeclareUnicodeComposite{\u}	{o}-{ "014F}
1377	\DeclareUnicodeComposite{H}	{O}-{ "0150}
1378	\DeclareUnicodeComposite{H}	{o}-{ "0151}
1379	\DeclareUnicodeComposite{'}	{R}-{ "0154}
1380	\DeclareUnicodeComposite{'}	{r}-{ "0155}
1381	\DeclareUnicodeComposite{c}	{R}-{ "0156}
1382	\DeclareUnicodeComposite{c}	{r}-{ "0157}
1383	\DeclareUnicodeComposite{v}	{R}-{ "0158}
1384	\DeclareUnicodeComposite{v}	{r}-{ "0159}
1385	\DeclareUnicodeComposite{'}	{S}-{ "015A}
1386	\DeclareUnicodeComposite{'}	{s}-{ "015B}
1387	\DeclareUnicodeComposite{^}	{S}-{ "015C}
1388	\DeclareUnicodeComposite{^}	{s}-{ "015D}
1389	\DeclareUnicodeComposite{c}	{S}-{ "015E}
1390	\DeclareUnicodeComposite{c}	{s}-{ "015F}
1391	\DeclareUnicodeComposite{v}	{S}-{ "0160}
1392	\DeclareUnicodeComposite{v}	{s}-{ "0161}
1393	\DeclareUnicodeComposite{c}	{T}-{ "0162}
1394	\DeclareUnicodeComposite{c}	{t}-{ "0163}
1395	\DeclareUnicodeComposite{v}	{T}-{ "0164}
1396	\DeclareUnicodeComposite{v}	{t}-{ "0165}
1397	\DeclareUnicodeComposite{~}	{U}-{ "0168}
1398	\DeclareUnicodeComposite{~}	{u}-{ "0169}
1399	\DeclareUnicodeComposite{=}	{U}-{ "016A}

1400	\DeclareUnicodeComposite{=}	{u}{"016B}
1401	\DeclareUnicodeComposite{u}	{U}{"016C}
1402	\DeclareUnicodeComposite{u}	{u}{"016D}
1403	\DeclareUnicodeComposite{r}	{U}{"016E}
1404	\DeclareUnicodeComposite{r}	{u}{"016F}
1405	\DeclareUnicodeComposite{H}	{U}{"0170}
1406	\DeclareUnicodeComposite{H}	{u}{"0171}
1407	\DeclareUnicodeComposite{k}	{U}{"0172}
1408	\DeclareUnicodeComposite{k}	{u}{"0173}
1409	\DeclareUnicodeComposite{^}	{W}{"0174}
1410	\DeclareUnicodeComposite{^}	{w}{"0175}
1411	\DeclareUnicodeComposite{^}	{Y}{"0176}
1412	\DeclareUnicodeComposite{^}	{y}{"0177}
1413	\DeclareUnicodeComposite{"}	{Y}{"0178}
1414	\DeclareUnicodeComposite{'}	{Z}{"0179}
1415	\DeclareUnicodeComposite{'}	{z}{"017A}
1416	\DeclareUnicodeComposite{.}	{Z}{"017B}
1417	\DeclareUnicodeComposite{.}	{z}{"017C}
1418	\DeclareUnicodeComposite{v}	{Z}{"017D}
1419	\DeclareUnicodeComposite{v}	{z}{"017E}
1420	\DeclareUnicodeComposite{v}	{A}{"01CD}
1421	\DeclareUnicodeComposite{v}	{a}{"01CE}
1422	\DeclareUnicodeComposite{v}	{I}{"01CF}
1423	\DeclareUnicodeComposite{v}	{i}{"01D0}
1424	\DeclareUnicodeComposite{v}	{i}{"01D0}
1425	\DeclareUnicodeComposite{v}	{O}{"01D1}
1426	\DeclareUnicodeComposite{v}	{o}{"01D2}
1427	\DeclareUnicodeComposite{v}	{U}{"01D3}
1428	\DeclareUnicodeComposite{v}	{u}{"01D4}
1429	\DeclareUnicodeComposite{'}	\AE{"01FC}
1430	\DeclareUnicodeComposite{'}	{Æ}{"01FC}
1431	\DeclareUnicodeComposite{'}	\ae{"01FD}
1432	\DeclareUnicodeComposite{'}	{æ}{"01FD}
1433	\DeclareUnicodeComposite{=}	\AE{"01E2}
1434	\DeclareUnicodeComposite{=}	{Æ}{"01E2}
1435	\DeclareUnicodeComposite{=}	\ae{"01E3}
1436	\DeclareUnicodeComposite{=}	{æ}{"01E3}
1437	\DeclareUnicodeComposite{v}	{G}{"01E6}
1438	\DeclareUnicodeComposite{v}	{g}{"01E7}
1439	\DeclareUnicodeComposite{v}	{K}{"01E8}
1440	\DeclareUnicodeComposite{v}	{k}{"01E9}
1441	\DeclareUnicodeComposite{k}	{O}{"01EA}
1442	\DeclareUnicodeComposite{k}	{o}{"01EB}
1443	\DeclareUnicodeComposite{v}	\j {"01F0}
1444	\DeclareUnicodeComposite{v}	{j}{"01F0}
1445	\DeclareUnicodeComposite{'}	{G}{"01F4}
1446	\DeclareUnicodeComposite{'}	{g}{"01F5}
1447	\DeclareUnicodeComposite{\textcommabelow}	{S}{"0218}
1448	\DeclareUnicodeComposite{\textcommabelow}	{s}{"0219}
1449	\DeclareUnicodeComposite{\textcommabelow}	{T}{"021A}
1450	\DeclareUnicodeComposite{\textcommabelow}	{t}{"021B}
1451	\DeclareUnicodeComposite{=}	{Y}{"0232}
1452	\DeclareUnicodeComposite{=}	{y}{"0233}


```

1453 \DeclareUnicodeComposite{\.}      {B}{"1E02}
1454 \DeclareUnicodeComposite{\.}      {b}{"1E03}
1455 \DeclareUnicodeComposite{\d}      {B}{"1E04}
1456 \DeclareUnicodeComposite{\d}      {b}{"1E05}
1457 \DeclareUnicodeComposite{\d}      {D}{"1E0C}
1458 \DeclareUnicodeComposite{\d}      {d}{"1E0D}
1459 \DeclareUnicodeComposite{\=}      {G}{"1E20}
1460 \DeclareUnicodeComposite{\=}      {g}{"1E21}
1461 \DeclareUnicodeComposite{\d}      {H}{"1E24}
1462 \DeclareUnicodeComposite{\d}      {h}{"1E25}
1463 \DeclareUnicodeComposite{\d}      {K}{"1E32}
1464 \DeclareUnicodeComposite{\d}      {k}{"1E33}
1465 \DeclareUnicodeComposite{\d}      {L}{"1E36}
1466 \DeclareUnicodeComposite{\d}      {l}{"1E37}
1467 \DeclareUnicodeComposite{\d}      {M}{"1E42}
1468 \DeclareUnicodeComposite{\d}      {m}{"1E43}
1469 \DeclareUnicodeComposite{\d}      {N}{"1E46}
1470 \DeclareUnicodeComposite{\d}      {n}{"1E47}
1471 \DeclareUnicodeComposite{\d}      {R}{"1E5A}
1472 \DeclareUnicodeComposite{\d}      {r}{"1E5B}
1473 \DeclareUnicodeComposite{\d}      {S}{"1E62}
1474 \DeclareUnicodeComposite{\d}      {s}{"1E63}
1475 \DeclareUnicodeComposite{\d}      {T}{"1E6C}
1476 \DeclareUnicodeComposite{\d}      {t}{"1E6D}
1477 \DeclareUnicodeComposite{\d}      {V}{"1E7E}
1478 \DeclareUnicodeComposite{\d}      {v}{"1E7F}
1479 \DeclareUnicodeComposite{\d}      {W}{"1E88}
1480 \DeclareUnicodeComposite{\d}      {w}{"1E89}
1481 \DeclareUnicodeComposite{\d}      {Z}{"1E92}
1482 \DeclareUnicodeComposite{\d}      {z}{"1E93}
1483 \DeclareUnicodeComposite{\d}      {A}{"1EA0}
1484 \DeclareUnicodeComposite{\d}      {a}{"1EA1}
1485 \DeclareUnicodeComposite{\d}      {E}{"1EB8}
1486 \DeclareUnicodeComposite{\d}      {e}{"1EB9}
1487 \DeclareUnicodeComposite{\d}      {I}{"1ECA}
1488 \DeclareUnicodeComposite{\d}      {i}{"1ECB}
1489 \DeclareUnicodeComposite{\d}      {O}{"1ECC}
1490 \DeclareUnicodeComposite{\d}      {o}{"1ECD}
1491 \DeclareUnicodeComposite{\d}      {U}{"1EE4}
1492 \DeclareUnicodeComposite{\d}      {u}{"1EE5}
1493 \DeclareUnicodeComposite{\d}      {Y}{"1EF4}
1494 \DeclareUnicodeComposite{\d}      {y}{"1EF5}
1495 </TU>

```

2 Package files

This file now also contains some packages that provide access to the more specialised encodings.

2.1 The fontenc package

This package allows authors to specify which encodings they will use. For each encoding FOO, the package looks to see if the encoding FOO has already been declared. If it has not, the file `fooenc.def` is loaded. The default encoding is set to be FOO.

In addition the package at the moment contains extra code to extend the `\@uclclist` (list of upper/lower case pairs) for encodings that involve cyrillic characters. THIS IS A TEMPORARY SOLUTION and will not stay this way forever (or so we hope) but right now we are missing a proper interface for this and didn't wanted to rush it.

```
1496 (*package)
```

Here we define a macro that extends the `\@uclclist` if needed and afterwards turns itself in a noop.

```
1497 \def\update@uclc@with@cyrillic{%
1498 \expandafter\def\expandafter\@uclclist\expandafter
1499 {\@uclclist
1500 \cyr\CYRA\cyrahch\CYRABHCH\cyrahchdsc\CYRABHCHDSC\cyrahhdze
1501 \CYRABHDZE\cyrahbha\CYRABHHA\cyrac\CYRAE\cyrb\CYRB\cyrbys
1502 \CYRBYUS\cyrc\CYRC\cyrch\CYRCH\cyrchldsc\CYRCHLDSC\cyrchrdsc
1503 \CYRCHRDSC\cyrchvcrs\CYRCHVCRS\cyrd\CYRD\cyrdelta\CYRDELTA
1504 \cyrdje\CYRDJE\cyrdze\CYRDZE\cyrdzhe\CYRDZHE\cyre\CYRE\cyreps
1505 \CYREPS\cyrerev\CYREREV\cyrery\CYRERY\cyrf\CYRF\cyrfita
1506 \CYRFITA\cyrg\CYRG\cyrgdsc\CYRGDSC\cyrgdschcrs\CYRGDSCHCRS
1507 \cyrgchcrs\CYRGHCRS\cyrgkh\CYRGHK\cyrgup\CYRGUP\cyrh\CYRH
1508 \cyrhdsdsc\CYRHDS\cyrhchcrs\CYRHHCRS\cyrhkh\CYRHHK\cyrhdsn
1509 \CYRHDSN\cyri\CYRI\cyrie\CYRIE\cyrii\CYRII\cyrishrt\CYRISHRT
1510 \cyrishrtdsc\CYRISHRTDSC\cyrizh\CYRIZH\cyrje\CYRJE\cyrk\CYRK
1511 \cyrkbeak\CYRKBEAK\cyrkdsdsc\CYRKDSC\cyrkchcrs\CYRKHCRS\cyrkhh
1512 \CYRKHK\cyrkvcrs\CYRKVCRS\cyrld\CYRL\cyrldsc\CYRLDSC\cyrllhk
1513 \CYRLHK\cyrllje\CYRLJE\cyrm\CYRM\cyrmdsc\CYRMDSC\cyrmhk\CYRMHK
1514 \cyrn\CYRN\cyrndsc\CYRNDSC\cyrng\CYRNG\cyrnkh\CYRNHK\cyrnje
1515 \CYRNJE\cyrnkh\CYRNLHK\cyro\CYRO\cyrotld\CYROTLD\cyrp\CYRP
1516 \cyrphk\CYRPHK\cyrq\CYRQ\cyrr\CYRR\cyrrdsdsc\CYRRDSC\cyrrhk
1517 \CYRRHK\cyrrtick\CYRRTICK\cyrs\CYRS\cyrsacrs\CYRSACRS
1518 \cyrschwa\CYRSCHWA\cyrsdsdsc\CYRSDSC\cyrsemisftsn\CYRSEMISFTSN
1519 \cyrsftsn\CYRSFTSN\cyrsh\CYRSH\cyrshch\CYRSHCH\cyrshha\CYRSHHA
1520 \cyrtd\CYRT\cyrtddsc\CYRTDSC\cyrtetse\CYRTETSE\cyrtshe\CYRTSHE
1521 \cyrtd\CYRU\cyrushrt\CYRUSHRT\cyrv\CYRV\cyrw\CYRW\cyrz\CYRZ
1522 \cyrza\CYRYA\cyrzat\CYRYAT\cyrzchcrs\CYRYHCRS\cyrzi\CYRYI\cyrzo
1523 \CYRYO\cyrzu\CYRYU\cyrz\CYRZ\cyrzdsdsc\CYRZDSC\cyrzh\CYRZH
1524 \cyrzhdsc\CYRZHDSC}%
1525 \let\update@uclc@with@cyrillic\relax
1526 }
```

Here we process each option:

```
1527 \DeclareOption*{%
1528 \let\encodingdefault\CurrentOption
```

From 2020/02/02 release onward we only load the encoding files if they haven't be loaded already. To check this we look if `\T@encoding` is already defined. If not we load (indicated by setting the switch `@tempwa` to true and we always load if we run in an older format (or rather in a rollback situation).

```
1529 \@tempwafalse
1530 \@ifl@t@r\fmtversion{2020/02/02}%
```

```

1531     {\expandafter\ifx\csname T@\CurrentOption\endcsname\relax
1532     \@tempswatrue\fi}%
1533     {\@tempswatrue}%

```

Load if necessary:

```

1534     \if@tempswa
1535         \edef\reserved@f{%
1536             \lowercase{\def\noexpand\reserved@f{\CurrentOption enc.def}}}%
1537         \reserved@f
1538         \InputIfFileExists\reserved@f
1539         {\PackageError{fontenc}%
1540         {Encoding file ‘\reserved@f’ not found.%
1541         \MessageBreak
1542         You might have misspelt the name of the encoding}%
1543         {Necessary code for this encoding was not
1544         loaded.\MessageBreak
1545         Thus calling the encoding later on will
1546         produce further error messages.}}%
1547     \let\reserved@f\relax

```

In case the current encoding is one of a list of known cyrillic ones we extend the `\@uclclist`:

```

1548     \expandafter\in@\expandafter{\CurrentOption}%
1549                                     {T2A,T2B,T2C,X2,LCY,OT2}%
1550     \ifin@

```

But only if it hasn't already been extended. This might happen if there are several calls to `fontenc` loading one of the above encodings. If we don't do this check the `\@uclclist` gets unnecessarily big, slowing down the processing at runtime.

```

1551     \expandafter\in@\expandafter\cyra\expandafter
1552                                     {\@uclclist}%
1553     \ifin@
1554     \else
1555         \update@uclc@with@cyrillic
1556     \fi
1557 \fi
1558 \fi
1559 }
1560 \ProcessOptions*

```

We select the new font encoding default (i.e., the last encoding specified in the option list. But this encoding may not work with the current `\f@shape`, e.g., `LY1` is not defined for `cmr` and therefore packages switching to `LY1` usually also change `\rmdefault`. But that only applies at `\begin{document}` so we get a spurious warning if we use what \LaTeX previously used:

```

1561 %\fontencoding\encodingdefault\selectfont

```

So instead we do this here:

```

1562 \usefont\encodingdefault\familydefault\seriesdefault\shapedefault

```

To save some space we get rid of the macro extending the `\@uclclist` (might have happened already).

```

1563 \let\update@uclc@with@cyrillic\relax

```

Finally we pretend that the fontenc package wasn't read in. This allows for using it several times, e.g., in a class file and in the preamble (at the cost of not getting any version info). That kind of hackery shows that using a general purpose package just for loading an encoding is not the right kind of interface for setting up encodings — it will get replaced at some point in the future.

```

1564 \let\@elt\relax
1565 \xdef\@fontenc@load@list{\@fontenc@load@list
1566   \@elt{\csname opt@fontenc.sty\endcsname}}

1567 \global\expandafter\let\csname ver@fontenc.sty\endcsname\relax
1568 \global\expandafter\let\csname opt@fontenc.sty\endcsname\relax
1569 \global\let\@ifl@ter@@\@ifl@ter
1570 \def\@ifl@ter#1#2#3#4#5{\global\let\@ifl@ter\@ifl@ter@@}
1571 \endpackage

```

File s

ltcounts.dtx

1 Counters and Lengths

Commands for defining and using counters. This file defines:

<code>\newcounter</code>	To define a new counter.
<code>\setcounter</code>	To set the value of counters.
<code>\addtocounter</code>	Increase the counter <code>#1</code> by the number <code>#2</code> .
<code>\stepcounter</code>	Increase a counter by one.
<code>\refstepcounter</code>	Increase a counter by one, also setting the value used by <code>\label</code> .
<code>\value</code>	For accessing the value of the counter as a TeX number (as opposed to <code>\the<counter></code> which expands to the <i>printed</i> representation of <code><counter></code>)
<code>\arabic</code>	<code>\arabic{<counter>}</code> : 1, 2, 3, ...
<code>\roman</code>	<code>\roman{<counter>}</code> : i, ii, iii, ...
<code>\Roman</code>	<code>\Roman{<counter>}</code> : I, II, III, ...
<code>\alph</code>	<code>\alph{<counter>}</code> : a, b, c, ...
<code>\Alph</code>	<code>\Alph{<counter>}</code> : A, B, C, ...
<code>\fnsymbol</code>	<code>\fnsymbol{<counter>}</code> : *, †, ‡, ...
<code>\counterwithin</code>	<code>\counterwithin{<counter>}{<within-counter>}</code> : Resets <code><counter></code> whenever <code><within-counter></code> is stepped. Also redefines <code>\the<counter></code> command to produce <code>\the<within-counter>.\arabic{<counter>}</code> . Star form omits redefining the print representation.
<code>\counterwithout</code>	<code>\counterwithout{<counter>}{<within-counter>}</code> : Removes <code><counter></code> from the reset list of <code><within-counter></code> . Also redefines <code>\the<counter></code> command to produce <code>\arabic{<counter>}</code> . Star form omits redefining the print representation.

1 (*2ekernel)

1.1 Environment Counter Macros

An environment `foo` has an associated counter defined by the following control sequences:

<code>\c@foo</code>	Contains the counter's numerical value. It is defined by <code>\newcount\foocounter</code> .
<code>\thefoo</code>	Macro that expands to the printed value of <code>\foocounter</code> . For example, if sections are numbered within chapters, and section headings look like Section II-3. The Nature of Counters then <code>\thesection</code> might be defined by: <code>\def\thesection</code> <code>{\@Roman{\c@chapter}-\@arabic{\c@section}}</code>
<code>\p@foo</code>	Macro that expands to a printed 'reference prefix' of counter <code>foo</code> . Any <code>\ref</code> to a value created by counter <code>foo</code> will produce the expansion of <code>\p@foo\thefoo</code> when the <code>\label</code> command is executed. See file <code>ltxref.dtx</code> for an extension of this mechanism.
<code>\cl@foo</code>	List of counters to be reset when <code>foo</code> stepped. Has format <code>\@elt{countera}\@elt{counterb}\@elt{counterc}</code> .

NOTE:

`\thefoo` and `\p@foo` *must* be defined in such a way that `\edef\bar{\thefoo}` or `\edef\bar{\p@foo}` defines `\bar` so that it will evaluate to the counter value at the

time of the `\edef`, even after `\foocounter` and any other counters have been changed. This will happen if you use the standard commands `\@arabic`, `\@Roman`, etc.

The following commands are used to define and modify counters.

`\refstepcounter{<foo>}`

Same as `\stepcounter`, but it also defines `\@currentreference` so that a subsequent `\label{<bar>}` command causes `\ref{<bar>}` to generate the current value of counter `<foo>`.

`\@definecounter{<foo>}`

Initializes counter `{<foo>}` (with empty reset list), defines `\p@foo` and `\thefoo` to be null. Also adds `<foo>` to `\cl@ckpt` – the reset list of a dummy counter `@ckpt` used for taking checkpoints for the `\include` system.

`\@addtoreset{<foo>}{<bar>}` : Adds counter `<foo>` to the list of counters `\cl@bar` to be reset when counter `<bar>` is stepped.

`\@removefromreset{<foo>}{<bar>}` : Removes counter `<foo>` to the list of counters `\cl@bar` to be reset when counter `<bar>` is stepped.

`\setcounter` `\setcounter{<foo>}{<val>}` : Globally sets `\foocounter` equal to `<val>`.

```
2 \def\setcounter#1#2{%
3   \@ifundefined{c@#1}%
4     {\@nocounterr{#1}}%
5     {\global\csname c@#1\endcsname#2\relax}}
```

(End definition for `\setcounter`.)

`\addtocounter` `\addtocounter{<foo>}{<val>}` Globally increments `\foocounter` by `<val>`.

```
6 \def\addtocounter#1#2{%
7   \@ifundefined{c@#1}%
8     {\@nocounterr{#1}}%
9     {\global\advance\csname c@#1\endcsname #2\relax}}
```

(End definition for `\addtocounter`.)

`\newcounter` `\newcounter{<newctr>}[<oldctr>]` Defines `<newctr>` to be a counter, which is reset when counter `<oldctr>` is stepped. If `<newctr>` already defined produces ‘`c@newctr` already defined’ error.

```
10 \def\newcounter#1{%
11   \expandafter\@ifdefinable \csname c@#1\endcsname
12     {\@definecounter{#1}}%
13   \@ifnextchar[{\@newctr{#1}}{-}]}
```

(End definition for `\newcounter`.)

`\value` `\value{<ctr>}` produces the value of counter `<ctr>`, for use with a `\setcounter` or `\addtocounter` command.

```
14 \def\value#1{\csname c@#1\endcsname}
```

(End definition for `\value`.)

`\@newctr`

```
15 \def\@newctr#1[#2]{%
16   \@ifundefined{c@#2}{\@nocounterr{#2}}{\@addtoreset{#1}{#2}}}
```

(End definition for `\@newctr`.)

`\stepcounter` `\stepcounterfoo` Globally increments counter `\c@F00` and resets all subsidiary counters.

```

17 \def\stepcounter#1{%
18   \addtocounter{#1}\@ne
19   \begingroup
20     \let\@elt\@stpelt
21     \csname cl@#1\endcsname
22   \endgroup}

```

(End definition for \stepcounter.)

`\@stpelt` Rather than resetting the “within” counter to zero we set it to -1 and then run `\stepcounter` that moves it to 0 and also initiates resetting the next level down.

```

23 \</2ekernel>
24 \<latexrelease>\IncludeInRelease{2015/01/01}{\@stpelt}
25 \<latexrelease>                                     {Reset nested counters}%
26 \<*2ekernel | latexrelease>
27 \def\@stpelt#1{\global\csname c@#1\endcsname \m@ne\stepcounter{#1}}%
28 \<latexrelease>\EndIncludeInRelease
29 \</2ekernel | latexrelease>
30 \<latexrelease>\IncludeInRelease{0000/00/00}{\@stpelt}
31 \<latexrelease>                                     {Reset nested counters}%%
32 \<latexrelease>\def\@stpelt#1{\global\csname c@#1\endcsname \z@}%
33 \<latexrelease>\EndIncludeInRelease
34 \<*2ekernel>

```

(End definition for \@stpelt.)

`\cl@@ckpt`

```

35 \def\cl@@ckpt{\@elt{page}}

```

(End definition for \cl@@ckpt.)

`\@definecounter`

```

36 \def\@definecounter#1{\expandafter\newcount\csname c@#1\endcsname
37   \setcounter{#1}\z@
38   \global\expandafter\let\csname cl@#1\endcsname\@empty
39   \@addtoreset{#1}{@ckpt}%
40   \global\expandafter\let\csname p@#1\endcsname\@empty
41   \expandafter
42   \gdef\csname the#1\expandafter\endcsname\expandafter
43     {\expandafter\@arabic\csname c@#1\endcsname}}

```

(End definition for \@definecounter.)

`\@addtoreset`

```

44 \def\@addtoreset#1#2{\expandafter\@cons\csname cl@#2\endcsname {\{#1\}}}

```

(End definition for \@addtoreset.)

```

45 \</2ekernel>

```

`\@removefromreset`

```

46 \<latexrelease>\IncludeInRelease{2018-04-01}
47 \<latexrelease>                                     {\@removefromreset}{Add interfaces}%
48 \<*2ekernel | latexrelease>

```

```
49 \def\@removefromreset#1#2{%
```

Even through this is internal and the programmer should know what he/she is doing we test here if counter #2 is defined. If not, the execution would run into a tight loop.

```
50 \ifundefined{c@#2}\relax
51 {\begingroup
52 \expandafter\let\csname c@#1\endcsname\@removefromreset
53 \def\@elt##1{%
54 \expandafter\ifx\csname c@##1\endcsname\@removefromreset
55 \else
56 \noexpand\@elt{##1}%
57 \fi}%
58 \expandafter\xdef\csname cl@#2\endcsname
59 {\csname cl@#2\endcsname}%
60 \endgroup}}
```

(End definition for \@removefromreset.)

`\ifbothcounters` Test if arg #1 and #2 are counters and if so execute #3.

```
61 \def\@ifbothcounters#1#2#3{%
62 \ifundefined{c@#1}{\@nocounterr{#1}}%
63 {% else counter is defined
64 \ifundefined{c@#2}{\@nocounterr{#2}}%
65 {% else both counter and within are defined
66 #3}}}
```

(End definition for \@ifbothcounters.)

`\counterwithout`

```
67 \def\counterwithout {\@ifstar\counterwithout@s\counterwithout@x}
68 \def\counterwithout@s#1#2{%
69 \@ifbothcounters{#1}{#2}{\@removefromreset{#1}{#2}}}
70 \def\counterwithout@x#1#2{%
71 \@ifbothcounters{#1}{#2}%
72 {\@removefromreset{#1}{#2}%
73 \expandafter
74 \gdef\csname the#1\expandafter\endcsname\expandafter
75 {\expandafter
76 \@arabic\csname c@#1\endcsname}}}
```

(End definition for \counterwithout.)

`\counterwithin`

```
77 \def\counterwithin{\@ifstar\counterwithin@s\counterwithin@x}
78 \def\counterwithin@s#1#2{%
79 \@ifbothcounters{#1}{#2}{\@addtoreset{#1}{#2}}}
80 \def\counterwithin@x#1#2{%
81 \@ifbothcounters{#1}{#2}%
82 {\@addtoreset{#1}{#2}%
83 \expandafter
84 \gdef\csname the#1\expandafter\endcsname\expandafter
85 {\csname the#2\expandafter\endcsname\expandafter
86 .\expandafter
87 \@arabic\csname c@#1\endcsname}}}
```


(End definition for \counterwithin.)

```
88 </2ekernel | latexrelease>
89 <latexrelease>\EndIncludeInRelease
90 <latexrelease>\IncludeInRelease{0000-00-00}
91 <latexrelease>          {\@removefromreset}{Add interfaces}%
92 <latexrelease>\let \@removefromreset \undefined
93 <latexrelease>\let \@ifbothcounters \undefined
94 <latexrelease>\let \counterwithout \undefined
95 <latexrelease>\let \counterwithout@s \undefined
96 <latexrelease>\let \counterwithout@x \undefined
97 <latexrelease>\let \counterwithin \undefined
98 <latexrelease>\let \counterwithin@s \undefined
99 <latexrelease>\let \counterwithin@x \undefined
100 <latexrelease>\EndIncludeInRelease
101 <*2ekernel>
```

Numbering commands for definitions of \theCOUNTER and \list arguments.
All commands can now be used in text and math mode.

\arabic Representation of $\langle counter \rangle$ as arabic numerals. Changed 29 Apr 86 to make it print the obvious thing if COUNTER not positive.

```
102 \def\arabic#1{\expandafter\@arabic\csname c@#1\endcsname}
```

(End definition for \arabic.)

\roman Representation of $\langle counter \rangle$ as lower-case Roman numerals.

```
103 \def\roman#1{\expandafter\@roman\csname c@#1\endcsname}
```

(End definition for \roman.)

\Roman Representation of $\langle counter \rangle$ as upper-case Roman numerals.

```
104 \def\Roman#1{\expandafter\@Roman\csname c@#1\endcsname}
```

(End definition for \Roman.)

\alph Representation of $\langle counter \rangle$ as a lower-case letter: 1 = a, 2 = b, etc.

```
105 \def\alph#1{\expandafter\@alph\csname c@#1\endcsname}
```

(End definition for \alph.)

\Alph Representation of $\langle counter \rangle$ as an upper-case letter: 1 = A, 2 = B, etc.

```
106 \def\Alph#1{\expandafter\@Alph\csname c@#1\endcsname}
```

(End definition for \Alph.)

\fnsymbol Representation of $\langle COUNTER \rangle$ as a footnote symbol: 1 = *, 2 = †, etc.

```
107 \def\fnsymbol#1{\expandafter\@fnsymbol\csname c@#1\endcsname}
```

(End definition for \fnsymbol.)

\@arabic \@arabic\F00counter Representation of \F00counter as arabic numerals.

```
108 \def\@arabic#1{\number #1} %% changed 29 Apr 86
```

(End definition for \@arabic.)

```

\@roman \@roman\F00counter Representation of \F00counter as lower-case Roman numerals.
109 \def\@roman#1{\romannumeral #1}

(End definition for \@roman.)

\@Roman \@Roman\F00counter Representation of \F00counter as upper-case Roman numerals.
110 \def\@Roman#1{\expandafter\@slowromancap\romannumeral #1@}

(End definition for \@Roman.)

\@slowromancap Fully expandable macro to change a roman number to uppercase.
111 \def\@slowromancap#1{\ifx @#1% then terminate
112     \else
113         \if i#1I\else\if v#1V\else\if x#1X\else\if l#1L\else\if
114             c#1C\else\if d#1D\else \if m#1M\else#1\fi\fi\fi\fi\fi\fi
115             \expandafter\@slowromancap
116         \fi
117     }

(End definition for \@slowromancap.)

\@alph \@alph\F00counter Representation of \F00counter as a lower-case letter: 1 = a, 2 =
b, etc.
118 \def\@alph#1{%
119     \ifcase#1\or a\or b\or c\or d\or e\or f\or g\or h\or i\or j\or
120     k\or l\or m\or n\or o\or p\or q\or r\or s\or t\or u\or v\or w\or x\or
121     y\or z\else\@ctrerr\fi}

(End definition for \@alph.)

\@Alph \@Alph\F00counter Representation of \F00counter as an upper-case letter: 1 = A, 2
= B, etc.
122 \def\@Alph#1{%
123     \ifcase#1\or A\or B\or C\or D\or E\or F\or G\or H\or I\or J\or
124     K\or L\or M\or N\or O\or P\or Q\or R\or S\or T\or U\or V\or W\or X\or
125     Y\or Z\else\@ctrerr\fi}

(End definition for \@Alph.)

\@fnsymbol Typesetting old fashioned footnote symbols. This can be done both in text or math mode
now.

This macro is another example of an ever recurring problem in TEX: Determining
if something is text-mode or math-mode. It is imperative for the decision between text
and math to be delayed until the actual typesetting is done as the code in question may
go through an \edef or \write where an \ifmmode test would be executed prematurely.
Hence in the implementation below, \@fnsymbol is not robust in itself but the parts
doing the actual typesetting are.

In the case of \@fnsymbol we make use of the robust command \TextOrMath which
takes two arguments and typesets the first if in text-mode and the second if in math-
mode. Note that in order for this command to make the correct decision, it must insert
a \relax token if run under regular TEX, which ruins any kerning between the preceding
characters and whatever awaits typesetting. If you use eTEX as engine for LATEX (as
recommended) this unfortunate side effect is not present.

```

```

126 </2ekernel>
127 <latexrelease>\IncludeInRelease{2015/01/01}{\@fnsymbol}{Use \TextOrMath}%
128 <*2ekernel | latexrelease>
129 \def\@fnsymbol#1{%
130   \ifcase#1\or \TextOrMath\textasteriskcentered *\or
131   \TextOrMath \textdagger \dagger\or
132   \TextOrMath \textdaggerdbl \ddagger \or
133   \TextOrMath \textsection \mathsection\or
134   \TextOrMath \textparagraph \mathparagraph\or
135   \TextOrMath \textbardbl \|\or
136   \TextOrMath {\textasteriskcentered\textasteriskcentered}{**}\or
137   \TextOrMath {\textdagger\textdagger}{\dagger\dagger}\or
138   \TextOrMath {\textdaggerdbl\textdaggerdbl}{\ddagger\ddagger}\else
139   \@ctrerrr \fi
140 }%
141 </2ekernel | latexrelease>
142 <latexrelease>\EndIncludeInRelease
143 <latexrelease>\IncludeInRelease{0000/00/00}{\@fnsymbol}{Use \TextOrMath}%
144 <latexrelease>\def\@fnsymbol#1{\ensuremath{%
145   <latexrelease> \ifcase#1\or *\or \dagger\or \ddagger\or \mathsection\or
146   <latexrelease> \mathparagraph\or \|\or **\or \dagger\dagger
147   <latexrelease> \or \ddagger\ddagger \else\@ctrerrr\fi}}%
148 <latexrelease>\EndIncludeInRelease
149 <*2ekernel>

```

(End definition for \@fnsymbol.)

`\TextOrMath` When using regular T_EX, we make this command robust so that it always selects the correct branch in an `\ifmmode` switch with the usual disadvantage of ruining kerning. For the application we use it for here that shouldn't matter. The alternative would be to mimic `\IeC` from `inputenc` but then it will have the disadvantage of choosing the wrong branch if appearing at the beginning of an alignment cell. However, users of eT_EX will be pleasantly surprised to get the best of both worlds and no bad side effects.

First some code for checking if we are running eT_EX but making sure not to permanently turn `\protected` into `\relax`.

```

150 </2ekernel>
151 <latexrelease>\IncludeInRelease{2015/01/01}{\TextOrMath}{\TextOrMath}%
152 <*2ekernel | latexrelease>
153 \begingroup\expandafter\expandafter\expandafter\endgroup
154 \expandafter\ifx\csname protected\endcsname\relax

```

In case of ordinary T_EX we define `\TextOrMath` as a robust command but make sure it always grabs its arguments. If we didn't do this it might very well gobble spaces in the input stream.

```

155 \DeclareRobustCommand\TextOrMath{%
156   \ifmmode \expandafter\@secondoftwo
157   \else \expandafter\@firstoftwo \fi
158 \protected@edef\TextOrMath#1#2{\TextOrMath{#1}{#2}}
159 \else

```

For eT_EX the situation is similar. The robust macro is a hidden one so that we again avoid problems of gobbling spaces in the input.

```

160 \protected\expandafter\def\csname TextOrMath\space\endcsname{%
161   \ifmmode \expandafter\@secondoftwo

```

```

162 \else \expandafter\@firstoftwo \fi}
163 \edef\TextOrMath#1#2{%
164 \expandafter\noexpand\csname TextOrMath\space\endcsname
165 {#1}{#2}}
166 \fi
167 </2ekernel | latexrelease>
168 <latexrelease>\EndIncludeInRelease
169 <latexrelease>\IncludeInRelease{0000/00/00}{\TextOrMath}{\TextOrMath}%
170 <latexrelease>\let\TextOrMath\@undefined
171 <latexrelease>\EndIncludeInRelease
172 <*2ekernel>

(End definition for \TextOrMath.)

173 </2ekernel>

```

File t

ltlength.dtx

1 Lengths

```

\newlength Declare #1 to be a new length command.
\setlength Set the length command, #1, to the value #2.
\addtolength Increase the value of the length command, #1, by the value #2.
\settowidth Set the length, #1 to the width of a box containing #2.
\settoheight Set the length, #1 to the height of a box containing #2.
\settodepth Set the length, #1 to the depth of a box containing #2.

1 <*2ekernel>
2 \message{lengths,}

\newlength

3 \def\newlength#1{\@ifdefinable#1{\newskip#1}}

(End definition for \newlength.)

\setlength

4 </2ekernel>
5 <latexrelease>\IncludeInRelease{2015/01/01}%
6 <latexrelease> \setlength}{Using \setlength with \dimen0}%
7 <*2ekernel | latexrelease>

8 \def\setlength#1#2{#1 #2\relax}
9 </2ekernel | latexrelease>
10 <latexrelease>\EndIncludeInRelease
11 <latexrelease>\IncludeInRelease{0000/00/00}%
12 <latexrelease> \setlength}{Using \setlength with \dimen0}%
13 <latexrelease>\def\setlength#1#2{#1#2\relax}
14 <latexrelease>\EndIncludeInRelease
15 <*2ekernel>

(End definition for \setlength.)

\addtolength \relax added 24 Mar 86

16 \def\addtolength#1#2{\advance#1 #2\relax}

(End definition for \addtolength.)

\settoheight The obvious analogs of \settowidth.
\settodepth
17 \def\@settodim#1#2#3{\setbox\@tempboxa\hbox{{#3}}#2#1\@tempboxa
\settowidth
Clear the memory afterwards (which might be a lot).
\@settodim

18 \setbox\@tempboxa\box\voidb@x}
19 \DeclareRobustCommand\settoheight{\@settodim\ht}
20 \DeclareRobustCommand\settodepth {\@settodim\dp}
21 \DeclareRobustCommand\settowidth {\@settodim\wd}

(End definition for \settoheight and others.)

```

`\@settopoint` This macro takes the contents of the skip register that is supplied as its argument and removes the fractional part to make it a whole number of points. This can be used in class files to avoid values like 345.4666666pt when calculating a dimension.

```
22 \def\@settopoint#1{\divide#1\p@\multiply#1\p@}
23 </2ekernel>
```

(End definition for \@settopoint.)

File u

ltfssbas.dtx

This file contains the main implementation of the ‘low level’ font selection commands. See other parts of the L^AT_EX distribution, or *The L^AT_EX Companion* for higher level documentation of the L^AT_EX ‘New’ Font Selection Scheme.

Warning: The macro documentation is still basically the documentation from the first NFSS release and therefore in some cases probably not completely accurate.

1 Preliminary macros

We define a number of macros that will be used later.

`\@nomath` `\@nomath` is used by most macros that will have no effect in math mode. It issues a warning message.

```

1  \<*2kernel>
2  \def\@nomath#1{\relax\ifmmode
3    \@font@warning{Command \noexpand#1invalid in math mode}\fi}

```

(End definition for \@nomath.)

`\no@alphabet@error` The macro `\no@alphabet@error` is called whenever the user requests a math *alphabet* that is not available in the current *version*. In math mode an error message is produced otherwise the command keeps silent. The argument is the name of the control sequence that identifies the math *alphabet*. The `\relax` at the beginning is necessary to prevent T_EX from scanning too far in certain situations.

```

4  \gdef\no@alphabet@error#1{\relax \ifmmode
5    \@latex@error{Math\space alphabet\space identifier\space
6      \noexpand#1is\space undefined\space in\space math\space
7      version\space ‘\math@version’}%
8    {Your\space requested\space math\space alphabet\space
9      is\space undefined\space in\space the\space current\space
10     math\space version.^~JCheck\space the\space spelling\space
11     or\space use\space the\space \noexpand\SetMathAlphabet\space
12     command.}
13    \fi}

```

(End definition for \no@alphabet@error.)

`\new@mathgroup` We also give a new name to `\newfam` and `\fam` to avoid verbal confusion (see the introduction).²²

`\mathgroup`

```

14  %\def\new@mathgroup{\alloc@8\mathgroup\chardef\sixt@@n}
15  \let\mathgroup\fam
16  %\let\newfam\new@mathgroup
17  \@onlypreamble\new@mathgroup

```

(End definition for \new@mathgroup and \mathgroup.)

²²For the same reason it seems advisable to `\let\fam` and `\newfam` equal to `\relax`, but this is commented out to retain compatibility to existing style files.

2 Macros for setting up the tables

`\DeclareFontShape` The macro `\DeclareFontShape` takes 6 arguments:

```
18 \def\DeclareFontShape{\begingroup
```

First we restore the catcodes of all characters used in the syntax.

```
19 \nfss@catcodes
```

We use `\expandafter \endgroup` to restore catcode in case something goes wrong with the argument parsing (suggested by Tim Van Zandt)

```
20 \expandafter\endgroup
```

```
21 \DeclareFontShape@}
```

(End definition for `\DeclareFontShape`.)

`\DeclareFontShape@`

```
22 \</2kernel>
```

```
23 \<*2kernel | latexrelease>
```

```
24 \<latexrelease>\IncludeInRelease{2020/02/02}%
```

```
25 \<latexrelease> \<\DeclareFontShape@>\{Maybe drop one m\}%
```

```
26 \def\DeclareFontShape@#1#2#3#4#5#6{%
```

```
27 \expandafter\ifx\csname #1+#2\endcsname\relax
```

```
28 \<\latex@error{Font family ‘#1+#2’ unknown}\<\@eha
```

```
29 \else
```

If the series value is incorrectly specified with an extra “m”, e.g., “mc” instead of just “c”, drop the surplus “m” but keep the “m” if it is by its own. In that case also issue a warning that the declaration needs correction.

For this we compare the given value #3 with one where we may have dropped an “m”. If nothing has changes, fine. Otherwise there was a wrong value which is now corrected in `\reservedb` so we use that and also issue a warning.

```
30 \edef\reserved@a{#3}%
```

```
31 \series@maybe@drop@one@m\reserved@a\reserved@b
```

```
32 \ifx\reserved@a\reserved@b\else
```

```
33 \<\latex@warning{Font shape declaration has incorrect series
```

```
34 value ‘#3’.\MessageBreak It should not contain an ‘m’!
```

```
35 Please correct it.\MessageBreak Found\}%
```

```
36 \fi
```

```
37 \expandafter
```

```
38 \xdef\csname#1/#2/\reserved@b/#4\endcsname
```

```
39 {\expandafter\noexpand\csname #5\endcsname}%
```

```
40 %
```

Most of the time #6 is empty so using `\let` to `\@empty` saves on space compared to using `\def`. That’s really one of the old space saving techniques and probably not necessary these days.

```
41 \def\reserved@a{#6}%
```

```
42 \global
```

```
43 \expandafter\let\csname#5\endcsname\expandafter\endcsname
```

```
44 \ifx\reserved@a\@empty
```

```
45 \<\@empty
```

```
46 \else
```

```
47 \reserved@a
```

```
48 \fi
```

```
49 \fi
```

```
50 }
```



```

51 </2ekernel | latexrelease>
52 <latexrelease>\EndIncludeInRelease
53 <latexrelease>\IncludeInRelease{0000/00/00}%
54 <latexrelease>          {\DeclareFontShape@}{Maybe drop one m}%
55 <latexrelease>
56 <latexrelease>\def\DeclareFontShape@#1#2#3#4#5#6{%
57 <latexrelease>  \expandafter\ifx\csname #1+#2\endcsname\relax
58 <latexrelease>    \@latex@error{Font family ‘#1+#2’ unknown}\@eha
59 <latexrelease>  \else
60 <latexrelease>    \expandafter
61 <latexrelease>      \xdef\csname#1/#2/#3/#4\endcsname{\expandafter\noexpand
62 <latexrelease>                                     \csname #5\endcsname}%
63 <latexrelease>    \def\reserved@a{#6}%
64 <latexrelease>    \global
65 <latexrelease>    \expandafter\let\csname#5\endcsname\expandafter\endcsname
66 <latexrelease>    \ifx\reserved@a\@empty
67 <latexrelease>      \@empty
68 <latexrelease>    \else
69 <latexrelease>      \reserved@a
70 <latexrelease>    \fi
71 <latexrelease>  \fi
72 <latexrelease> }
73 <latexrelease>\EndIncludeInRelease
74 < *2ekernel>

```

(End definition for \DeclareFontShape@.)

\DeclareFixedFont Define a direct font switch that avoids all overhead.

```

75 \def\DeclareFixedFont#1#2#3#4#5#6{%
76   \begingroup
77     \math@fontsfalse
78     \every@math@size{}%
79     \fontsize{#6}\z@
80     \usefont{#2}{#3}{#4}{#5}%
81     \global\expandafter\let\expandafter#1\the\font
82   \endgroup
83 }

```

(End definition for \DeclareFixedFont.)

\do@subst@correction

```

84 \def\do@subst@correction{%
85   \xdef\subst@correction{%
86     \font@name
87     \global\expandafter\font
88     \csname \curr@fontshape/\f@size\endcsname
89     \noexpand\fontname\font
90     \relax}%

```

Calling `\subst@correction` after the current group means calling it after we have loaded the substitution font which is done inside a group.

```

91     \aftergroup\subst@correction
92 }

```

(End definition for \do@subst@correction.)

`\DeclareFontFamily`

```
93 \def\DeclareFontFamily#1#2#3{%
```

If we want fast checking for the encoding scheme we can just check for `\T@..` being defined.

```
94 % \@tempswafalse
95 % \def\reserved@b{#1}%
96 % \def\cdp@elt##1##2##3##4{\def\reserved@c{##1}%
97 %     \ifx\reserved@b\reserved@c \@tempswatrue\fi}%
98 % \cdp@list
99 % \if@tempswa
100 % \ifundefined{T@#1}%
101 % {
102 %   \latex@error{Encoding scheme ‘#1’ unknown}\@eha
103 % }%
104 % }
```

Now we have to define the macro `\(<#1>+<#2>)` to contain `#3`. But since most of the time `#3` will be empty we use `\let` in a tricky way rather than a simple `\def` since this will save internal memory. We store the argument `#3` in a temporary macro `\reserved@a`.

```
105 \def\reserved@a{#3}%
```

We compare `\reserved@a` with `\@empty`. If these two are the same we `\let` the ‘extra’ macro equal to `\@empty` which is not the same as doing a `\let` to `\reserved@a` — the latter would blow one extra memory location rather than reusing the one from `\@empty`.

```
106 \global
107 \expandafter\let\csname #1+#2\expandafter\endcsname
108 \ifx \reserved@a\@empty
109 \@empty
110 \else \reserved@a
111 \fi
112 }%
113 }
```

(End definition for \DeclareFontFamily.)

`\cdp@list` We initialize the code page list to be empty.

```
114 \let\cdp@list\@empty
115 \@onlypreamble\cdp@list
```

(End definition for \cdp@list.)

`\cdp@elt`

```
116 \let\cdp@elt\relax
117 \@onlypreamble\cdp@elt
```

(End definition for \cdp@elt.)

`\DeclareFontEncoding`

```
118 \def\DeclareFontEncoding{%
```

First we start with ignoring all blanks and newlines since every surplus space in the second or third argument will come out in a weird place in the document.

```

119 \begingroup
120 \nfss@catcodes
121 \expandafter\endgroup
122 \DeclareFontEncoding@
123 \@onlypreamble\DeclareFontEncoding

124 \def\DeclareFontEncoding@#1#2#3{%
125 \expandafter
126 \ifx\csname T@#1\endcsname\relax
127 \def\cdp@elt{\noexpand\cdp@elt}%
128 \xdef\cdp@list{\cdp@list\cdp@elt{#1}%
129 {\default@family}{\default@series}%
130 {\default@shape}}%

```

To support encoding dependent commands (like accents) we initialise the command `\encoding-cmd` to be `\@changed@cmd`. (See `ltoutenc.dtx` for details.)

```

131 \expandafter\let\csname#1-cmd\endcsname\@changed@cmd
132 \else
133 \@font@info{Redefining font encoding #1}%
134 \fi

135 \global\@namedef{T@#1}{#2}%
136 \global\@namedef{M@#1}{\default@M#3}%

```

Keep a record of the last encoding being declared:

```

137 \xdef\LastDeclaredEncoding{#1}%
138 }
139 \@onlypreamble\DeclareFontEncoding@

```

(End definition for `\DeclareFontEncoding`.)

`\LastDeclaredEncoding` The last encoding being declared by `\DeclareFontEncoding`.

```

140 \def\LastDeclaredEncoding{}

```

(End definition for `\LastDeclaredEncoding`.)

`\DeclareFontSubstitution`

```

141 \def\DeclareFontSubstitution#1#2#3#4{%
142 \expandafter
143 \ifx\csname T@#1\endcsname\relax
144 \@latex@error{Encoding scheme ‘#1’ unknown}\@eha
145 \else
146 \begingroup

```

We loop through the `\cdp@list` and rebuild it anew in `\toks@` thereby replacing the defaults for the encoding in question with the new defaults. It is important to store the encoding to test against expanded in `\reserved@a` since it might just be `\LastDeclaredEncoding` that is passed as `#1`.

```

147 \edef\reserved@a{#1}%
148 \toks@{}%
149 \def\cdp@elt##1##2##3##4{%
150 \def\reserved@b{##1}%
151 \ifx\reserved@a\reserved@b

```

Here we use the new defaults but we use `##1` (i.e., the encoding name already stored previously) since we know that it is expanded.

```
152         \addto@hook\toks@{\cdp@elt{##1}{#2}{#3}{#4}}%
153         \else
```

If `\reserved@a` and `\reserved@b` differ then we simply copy from the old list to the new.

```
154         \addto@hook\toks@{\cdp@elt{##1}{##2}{##3}{##4}}%
155         \fi}%
156         \cdp@list
157         \xdef\cdp@list{\the\toks@}%
158     \endgroup
159     \global
160     \@namedef{D@#1}{%
161         \def\default@family{#2}%
162         \def\default@series{#3}%
163         \def\default@shape{#4}%
164         }%
165     \fi
166 }
167 \@onlypreamble\DeclareFontSubstitution
```

(End definition for `\DeclareFontSubstitution`.)

`\DeclareFontEncodingDefaults`

```
168 \def\DeclareFontEncodingDefaults#1#2{%
169     \ifx\relax#1\else
170         \ifx\default@T\@empty\else
171             \@font@info{Overwriting encoding scheme text defaults}%
172             \fi
173             \gdef\default@T{#1}%
174         \fi
175     \ifx\relax#2\else
176         \ifx\default@M\@empty\else
177             \@font@info{Overwriting encoding scheme math defaults}%
178             \fi
179             \gdef\default@M{#2}%
180         \fi
181     }
182 \@onlypreamble\DeclareFontEncodingDefaults
```

(End definition for `\DeclareFontEncodingDefaults`.)

`\default@T`

`\default@M`

```
183 \let\default@T\@empty
184 \let\default@M\@empty
```

(End definition for `\default@T` and `\default@M`.)

`\DeclarePreloadSizes`

```
185 \def\DeclarePreloadSizes#1#2#3#4#5{%
186     \@ifundefined{T@#1}%
187     {\@latex@error{Encoding scheme ‘#1’ unknown}\@eha}%
188     {%
```

Don't know at the moment what this group here does!

```
189 \begingroup
```

We define a macro `\reserved@f`²³ that grabs the next *size* and loads the corresponding font. This is done by delimiting `\reserved@f`'s only argument by the token `,` (comma).

```
190 \def\reserved@f##1,{%
```

The end of the list will be detected when there are no more elements, i.e. when `\reserved@f`'s argument is empty. The trick used here is explained in Appendix D of the *T_EXbook*: if the argument is empty the `\if` will select the first clause and `\let \reserved@f \relax`. (We use the `>` character here since it cannot appear in font file names.)

```
191 \if>##1>%
192 \let\reserved@f\relax
193 \else
```

Otherwise, we define `\font@name` appropriately and call `\pickup@font` to do the work. Note that the requested `\curr@fontshape` combination must have been defined, or you will get an error. The definition of `\font@name` is carried out globally to be consistent with the rest of the code in this file.

```
194 \xdef\font@name{\csname#1/#2/#3/#4/##1\endcsname}%
195 \pickup@font
```

Now we forget the name of the font just loaded. More precisely, we set the corresponding control sequence to `\relax`. This means that later on, when the font is first used, the macro `\define@newfont` is called again to execute the 'extra' macro for this font.

```
196 \global\expandafter\let\font@name\relax
197 \fi
```

Finally we call `\reserved@f` again to process the next *size*. If `\reserved@f` was `\let` equal to `\relax` this will end the macro.

```
198 \reserved@f}%
```

We finish with reinserting the list of sizes after the `\reserved@f` macro and appending an empty element so that the end of the list is recognized properly.

```
199 \reserved@f#5,,%
200 \endgroup
201 }%
202 }
203 \@onlypreamble\DeclarePreloadSizes
```

(End definition for `\DeclarePreloadSizes`.)

`\ifmath@fonts` We need a switch to decide if we have to switch math fonts. For this purpose we provide `\ifmath@fonts` that can be set to true or false by the `\S@...` macros depending on if math fonts are provided for this size or not. The default is of course to switch all fonts.

```
204 \newif\ifmath@fonts \math@fontstrue
```

(End definition for `\ifmath@fonts`.)

²³We cannot use `\@tempa` since it is needed in `\pickup@font`.

`\DeclareMathSizes` `\DeclareMathSizes` takes the text size, math text size, math script size, and math scriptscript size as arguments and defines the right `\S@...` macro.

```

205 \def\DeclareMathSizes{%
206   \@ifstar{\@DeclareMathSizes\math@fontsfalse}%
207   {\@DeclareMathSizes{}}%
208   \@onlypreamble\DeclareMathSizes

```

(End definition for `\DeclareMathSizes` and `\DeclareMathSizes*`.)

`\@DeclareMathSizes` This modification by Michael J. Downes on comp.text.tex on 2002/10/17 allows the user to have settings such as

`\DeclareMathSizes{9.5dd}{9.5dd}{7.4dd}{6.6dd}`.

```

209 \</2kernel>
210 \<latexrelease>\IncludeInRelease{2015/01/01}{\@DeclareMathSizes}%
211 \<latexrelease>          {Arbitrary units in \DeclareMathSizes}%
212 \<*2kernel | latexrelease>
213 \def\@DeclareMathSizes #1#2#3#4#5{%
214   \@defaultunits\dimen@ #2pt\relax\@nnil
215   \if $#3$%
216     \expandafter\let\csname S@\strip@pt\dimen@\endcsname\math@fontsfalse
217   \else
218     \@defaultunits\dimen@ii #3pt\relax\@nnil
219     \@defaultunits\@tempdima #4pt\relax\@nnil
220     \@defaultunits\@tempdimb #5pt\relax\@nnil
221     \toks@{#1}%
222     \expandafter\xdef\csname S@\strip@pt\dimen@\endcsname{%
223       \gdef\noexpand\tf@size{\strip@pt\dimen@ii}%
224       \gdef\noexpand\s@size{\strip@pt\@tempdima}%
225       \gdef\noexpand\ssf@size{\strip@pt\@tempdimb}%
226       \the\toks@
227     }%
228   \fi
229 }%
230 \</2kernel | latexrelease>
231 \<latexrelease>\EndIncludeInRelease
232 \<latexrelease>\IncludeInRelease{0000/00/00}{\@DeclareMathSizes}%
233 \<latexrelease>          {Arbitrary units in \DeclareMathSizes}%
234 \<latexrelease>\def\@DeclareMathSizes#1#2#3#4#5{%
235 \<latexrelease>    \@defaultunits\dimen@#2pt\relax\@nnil
236 \<latexrelease>    \if $#3$%
237 \<latexrelease>      \expandafter \let
238 \<latexrelease>        \csname S@\strip@pt\dimen@\endcsname
239 \<latexrelease>        \math@fontsfalse
240 \<latexrelease>    \else
241 \<latexrelease>      \expandafter \gdef
242 \<latexrelease>        \csname S@\strip@pt\dimen@\endcsname
243 \<latexrelease>          {\gdef\tf@size{#3}\gdef\s@size{#4}%
244 \<latexrelease>                                \gdef\ssf@size{#5}%
245 \<latexrelease>          #1%
246 \<latexrelease>                                }%
247 \<latexrelease>    \fi}%
248 \<latexrelease>\EndIncludeInRelease
249 \<*2kernel>

```

```
250 \@onlypreamble\@DeclareMathSizes
```

(End definition for \@DeclareMathSizes.)

3 Selecting a new font

3.1 Macros for the user

`\fontencoding` As we said in the introduction a font is described by four parameters. We first define macros to specify the wanted *family*, *series*, or *shape*. These are simply recorded in internal macros `\f@family`, `\f@series`, and `\f@shape`, resp. We use `\edef`'s so that the arguments can also be macros.

```
251 \DeclareRobustCommand\fontencoding[1]{%
252   \expandafter\ifx\csname T@#1\endcsname\relax
253   \latexerror{Encoding scheme ‘#1’ unknown}\@eha
254   \else
255   \edef\f@encoding{#1}%
256   \ifx\cf@encoding\f@encoding
```

If the new encoding is the same as the old encoding we have nothing to do. However, in case we had a sequence of several encoding changes without a `\selectfont` in-between we can save processing by making sure that `\enc@update` is `\relax`.

```
257   \let\enc@update\relax
258   \else
```

If current and new encoding differ we define the macro `\enc@update` to contain all updates necessary at `\selectfont` time.

```
259   \let\enc@update\@enc@update
260   \fi
261   \fi
262 }
```

(End definition for `\fontencoding` and `\f@encoding`.)

`\@enc@update`

```
263 \def\@enc@update{%
```

When `\@enc@update` is executed `\f@encoding` holds the encoding name for the new encoding and `\cf@encoding` the name of the last active encoding.

We start by setting the init command for encoding dependent macros to `\@changed@cmd`.

```
264   \expandafter
265   \let
266   \csname\cf@encoding-cmd\endcsname
267   \@changed@cmd
```

Then we turn the one for the new encoding to `\@current@cmd` (see `ltoutenc.dtx` for further explanations).

```
268   \expandafter
269   \let
270   \csname\f@encoding-cmd\endcsname
271   \@current@cmd
```

We execute the default settings `\default@T`, followed by the one for the new encoding.

```
272   \default@T
273   \csname T@\f@encoding\endcsname
```

Finally we change the default substitution values, disable `\enc@update` and make `\f@encoding` officially the current encoding.

```

274         \csname D@\f@encoding\endcsname
275         \let\enc@update\relax
276         \let\cf@encoding\f@encoding
277     }

```

(End definition for \@enc@update.)

`\enc@update` The default action in `\selectfont` is to do nothing.

```

278 \let\enc@update\relax

```

(End definition for \enc@update.)

`\fontfamily`

`\f@family`

```

279 \DeclareRobustCommand\fontfamily[1]{\edef\f@family{#1}}

```

`\fontseries`

There are now defined later (and differently).

`\f@series`

```

280 %\DeclareRobustCommand\fontseries[1]{\edef\f@series{#1}}

```

`\fontshape`

```

281 %\DeclareRobustCommand\fontshape [1]{\edef\f@shape{#1}}

```

`\f@shape`

(End definition for \fontfamily and others.)

`\usefont` Some handy abbreviation if you want to get some particular font in the current size. If also the size should change one has to issue a `\fontsize` command first.

`\fontencoding` needs to do some setup work so we call that, but instead of calling `\fontfamily`, `\fontseries` and `\fontshape` it earlier versions of this code did, we now set `\f@family`, etc. directly. If we would call `\fontseries` or `\fontshape` as it was done in the past, they would now interact with the existing series and shape which is not desired if we intend to use an explicit font shape!

```

282 </2ekernel>
283 <*2ekernel | latexrelease>
284 <latexrelease>\IncludeInRelease{2021/06/01}%
285 <latexrelease>                {\usefont}{Force font face}%
286 \DeclareRobustCommand\usefont[4]{\fontencoding{#1}%
287     \edef\f@family{#2}%
288     \set@target@series{#3}%
289     \edef\f@shape{#4}%

```

Any earlier `\fontseries`, etc. should be canceled and we should switch unconditionally to the requested font face so we drop any code that may have been stored in `\delayed@f@adjustment`.

```

290     \let\delayed@f@adjustment\@empty
291     \selectfont
292     \ignorespaces}
293 </2ekernel | latexrelease>
294 <latexrelease>\EndIncludeInRelease
295 <latexrelease>\IncludeInRelease{2020/02/02}%
296 <latexrelease>                {\usefont}{Drop m in usefont}%
297 <latexrelease>
298 <latexrelease>\DeclareRobustCommand\usefont[4]{\fontencoding{#1}%
299 <latexrelease>     \edef\f@family{#2}%
300 <latexrelease>     \set@target@series{#3}%
301 <latexrelease>     \edef\f@shape{#4}\selectfont

```



```

302 <latexrelease> \ignorespaces}
303 <latexrelease>
304 <latexrelease>\EndIncludeInRelease
305 <latexrelease>\IncludeInRelease{0000/00/00}%
306 <latexrelease> {\usefont}{Drop m in usefont}%
307 <latexrelease>
308 <latexrelease>\DeclareRobustCommand\usefont[4]{\fontencoding{#1}%
309 <latexrelease> \edef\f@family{#2}%
310 <latexrelease> \edef\f@series{#3}%
311 <latexrelease> \edef\f@shape{#4}\selectfont
312 <latexrelease> \ignorespaces}
313 <latexrelease>
314 <latexrelease>\EndIncludeInRelease
315 <*2ekernel>

```

(End definition for \usefont.)

\linespread The command `\linespread` changes the current `\baselinestretch` by calling `\set@fontsize`. The values for `\f@size` and `\f@baselineskip` will be left unchanged.

```

316 \DeclareRobustCommand\linespread[1]
317 {\set@fontsize{#1}\f@size\f@baselineskip}

```

(End definition for \linespread.)

\fontsize We also define a macro that allows to specify a size. In this case, however, we also need the value of `\baselineskip`. As the first argument to `\set@fontsize` we pass the current value of `\baselinestretch`. This will either match the internal value (in which case nothing changes, or it will be an updated value due to a user change of that macro using `\renewcommand`. If we would pass the internal `\f@linespread` such a change would be effectively overwritten by a size change.

```

318 \DeclareRobustCommand\fontsize[2]
319 {\set@fontsize\baselinestretch{#1}{#2}}

```

(End definition for \fontsize.)

\f@linespread This macro holds the current internal value for `\baselinestretch`.

```

320 \let\f@family\@empty
321 \let\f@series\@empty
322 \let\f@shape\@empty
323 \let\f@size\@empty
324 \let\f@baselineskip\@empty
325 \let\f@linespread\@empty

```

(End definition for \f@linespread.)

\cf@encoding

```

326 \let\f@encoding\@empty
327 \let\cf@encoding\@empty

```

(End definition for \cf@encoding.)

`\@defaultunits` The function `\@defaultunits` when wrapped around a `dimen` or `skip` assignment supplies default units. Usage:

```
\@defaultunits\dimen@=#1pt\relax\@nnil
```

Note: the `\relax` is *important*. Other units can be substituted for the ‘pt’ if desired.

We use `\remove@to@nnil` as an auxiliary macros for `\@defaultunits`. It just has to gobble the supplied default unit ‘pt’ or whatever, if it wasn’t used in the assignment.

```
328 \def\@defaultunits{\afterassignment\remove@to@nnil}
```

(End definition for `\@defaultunits`.)

`\strip@pt` This macro strips the characters `pt` produced by using `\the` on a `dimen` register.

`\rem@pt`

```
329 \begingroup
330   \catcode‘P=12
331   \catcode‘T=12
332   \lowercase{
333     \def\x{\def\rem@pt##1.##2PT{##1\ifnum##2>\z@.##2\fi}}
334   \expandafter\endgroup\x
335   \def\strip@pt{\expandafter\rem@pt\the}
```

(End definition for `\strip@pt` and `\rem@pt`.)

`\mathversion` `\mathversion` takes the *math version* name as argument, defines `\math@version` appropriately and switches to the font selected forcing a call to `\glb@settings` if the *version* is known to the system.

```
336 \DeclareRobustCommand\mathversion[1]
337   {\@nomath\mathversion
338     \expandafter\ifx\csname mv@#1\endcsname\relax
339     \latex@error{Math version ‘#1’ is not defined}\@eha\else
340     \edef\math@version{#1}%
```

We need to force a math font setup both now and at the point where we return to the previous math version. Forcing a math font setup can simply be done by setting `\glb@currsz` to an invalid value since this will trigger the setup when the formula starts.

```
341   \gdef\glb@currsz{}%
```

When the scope of the current `\mathversion` ends we need to restore the old setup. However this time we need to force it directly at least if we are inside `math`, otherwise we could wait. Another way to enhance this code here is to do the setting only if the version really has changed after all. This might be interesting in case of `amstext` and `boldsymbol`.

```
342   \aftergroup\glb@settings
343   \fi}
```

(End definition for `\mathversion` and `\math@version`.)

If \TeX would support a hook just before the end of a formula (opposite of `\everymath` so to speak) the implementation of the algorithm would be much simpler because in that case we would set up the correct math fonts at this point without having to worry about incorrect settings due to nesting. The same would be true if in \LaTeX the use of `$` (as the primitive \TeX command) would be impossible and instead only a higher-level interface would be available. Note that this does not mean that a `$` couldn’t be the short-hand

for starting and stopping that higher-level interface, it only means that the direct \TeX function must be hidden.

Anyway, since we don't have this and won't have it in $\text{\LaTeX 2}_{\epsilon}$ we need to implement it in a somewhat slower way.

We test for the current math font setup on entry of a formula, i.e., on the hooks \everymath and \everydisplay . But since these hooks may contain user data we provide ourselves with an internal version of these hooks which stays frozen.

```

\frozen@everymath New internal names for \everymath and \everydisplay.
\frozen@everydisplay 344 \let\frozen@everymath\everymath
                    345 \let\frozen@everydisplay\everydisplay

                    (End definition for \frozen@everymath and \frozen@everydisplay.)

\everymath Now we provide now user hooks that will be called in the frozen internals.
\everydisplay 346 \newtoks\everymath
              347 \newtoks\everydisplay

              (End definition for \everymath and \everydisplay.)

\frozen@everymath Now we define the behaviour of the frozen hooks: first check the math setup then call
                  the user hook.
              348 \frozen@everymath = {\check@mathfonts
              349                  \the\everymath}

              (End definition for \frozen@everymath.)

\frozen@everydisplay Ditto for the display hook.
              350 \frozen@everydisplay = {\check@mathfonts
              351                  \the\everydisplay}

              (End definition for \frozen@everydisplay.)

\curr@math@size This holds locally the current math size.
              352 \let\curr@math@size\@empty

              (End definition for \curr@math@size.)

```

3.2 Macros for loading fonts

```

\pickup@font The macro \pickup@font which is used in \selectfont is very simple: if the font name
              is undefined (i.e. not known yet) it calls \define@newfont to load it.
              353 \def\pickup@font{%
              354     \expandafter \ifx \font@name \relax
              355     \define@newfont
              356     \fi}

              (End definition for \pickup@font.)

```

`\split@name` `\pickup@font` assumes that `\font@name` is set but it is sometimes called when `\f@family`, `\f@series`, `\f@shape`, or `\f@size` may have the wrong settings (see, e.g., the definition of `\getanddefine@fonts`). Therefore we need a macro to extract font *family*, *series*, *shape*, and *size* from the font name. To this end we define `\split@name` which takes the font name as a list of characters of `\catcode 12` (without the backslash at the beginning) delimited by the special control sequence `\@nil`. This is not very complicated: we first ensure that `/` has the right `\catcode`

```
357 {\catcode'\/=12
```

and define `\split@name` so that it will define our private `\f@encoding`, `\f@family`, `\f@series`, `\f@shape`, and `\f@size` macros.

```
358 \gdef\split@name#1/#2/#3/#4/#5\@nil{\def\f@encoding{#1}%
359                                     \def\f@family{#2}%
360                                     \def\f@series{#3}%
361                                     \def\f@shape{#4}%
362                                     \def\f@size{#5}}}
```

(End definition for `\split@name`.)

`\curr@fontshape` Abbreviation which may get removed again for speed.

```
363 \def\curr@fontshape{\f@encoding/\f@family/\f@series/\f@shape}
```

(End definition for `\curr@fontshape`.)

`\define@newfont` Now we can tackle the problem of defining a new font.

```
364 \def\define@newfont{%
```

We have already mentioned that the token list that `\split@name` will get as argument must not start with a backslash. To reach this goal we will set the `\escapechar` to `-1` so that the `\string` primitive will not generate an escape character. To keep this change local we open a group. We use `\begingroup` for this purpose since `\define@newfont` might be called in math mode, and an empty `\bgroup... \egroup` would add an empty Ord atom to the math list and thus affect the spacing.

Also locally redefine `\typeout` so that ‘No file ...fd’ Warnings become Font Info message just sent to the log file.

```
365 \begingroup
366 \let\typeout\@font@info
367 \escapechar\m@ne
```

Then we extract *encoding scheme*, *family*, *series*, *shape*, and *size* from the font name. Note the four `\expandafter`’s so that `\font@name` is expanded first, then `\string`, and finally `\split@name`.

```
368 \expandafter\expandafter\expandafter
369 \split@name\expandafter\string\font@name\@nil
```

If the `\curr@fontshape` combination is not available, (i.e. undefined) we call the macro `\wrong@fontshape` to take care of this case. Otherwise `\extract@font` will load the external font for us.

```
370 % \expandafter\ifx
371 % \csname\curr@fontshape\endcsname \relax
372 \try@load@fontshape % try always
373 % \fi
374 \expandafter\ifx
375 \csname\curr@fontshape\endcsname \relax
376 \wrong@fontshape\else
```

To allow substitution we call the `curr@fontshape` macro which usually will expand to `\relax` but may hold code for substitution (see `\subst@fontshape` definition).

```
377 %      \csname\curr@fontshape\endcsname
378      \extract@font\fi
```

We are nearly finished and must only restore the `\escapechar` by closing the group.

```
379 \endgroup}

380 \def\try@load@fontshape{%
381   \expandafter
382   \ifx\csname \f@encoding+\f@family\endcsname\relax
383   \@font@info{Trying to load font information for
384     \f@encoding+\f@family}%
```

We predefine this combination to be `\@empty` which means that next time we don't try again unnecessary in case we don't find a `.fd` file. If the file contains a `\DeclareFontFamily` command than this setting will be overwritten.

```
385   \global\expandafter\let
386   \csname \f@encoding+\f@family\endcsname\@empty
```

Set the catcodes used in the syntax, but do it only once (this will be restored at the end of the font loading group).

```
387   \nfss@catcodes
388   \let\nfss@catcodes\relax
```

For increased portability make the external filename monospace, but look for the (old style) mixed case filename if the first attempt fails.

On any monospace system this means that the file is looked for twice which takes up time and string space, but at least for this release Check for both names to give people time to re-install their private `fd` files with lowercase names.

```
389   \edef\reserved@a{%
390     \lowercase{%
391       \noexpand\InputIfFileExists{\f@encoding\f@family.fd}}}%
392   \reserved@a\relax
393   {\@input@\f@encoding\f@family.fd}}%
394 \fi}
```

(End definition for `\define@newfont`.)

`\nfss@catcodes` This macro should contain the standard `\catcode` assignments to all characters which are used in the commands found in an `.fd` file and which might have special `\catcodes` in the middle of a document. If necessary, this list can be extended in a package file using a suitable number of `\expandafter`, i.e.,

```
\expandafter\def\expandafter\nfss@catcodes
\expandafter{\nfss@catcodes <additional settings>}
```

Note, that this macro might get executed several times since it is also called by `\DeclareFontShape`, thus it probably should not be misused as a general purpose hook.

```
395 \def\nfss@catcodes{%
```

We start by making `@` a letter and ignoring all blanks and newlines.

```
396   \makeatletter
397   \catcode'\ 9%
398   \catcode'\^^I9%
399   \catcode'\^^M9%
```

Then we set up `\`, `{`, `}`, `#` and `%` in case an `.fd` file is loaded during a verbatim environment.

```

400      \catcode'\z@
401      \catcode'\{ \one
402      \catcode'\} \tw@
403      \catcode'\#6%
404      \catcode'\^7%
405      \catcode'\%14%

```

The we make sure that the important syntax parts have the right `\catcode`.

```

406      \@makeother\<%
407      \@makeother\>%
408      \@makeother\*%
409      \@makeother\.%
410      \@makeother\-%
411      \@makeother\/%
412      \@makeother\[%
413      \@makeother\]%
414      \@makeother\'%
415      \@makeother\'%
416      \@makeother\"%
417 }

```

(End definition for \nfss@catcodes.)

`\LoadFontDefinitionFile` Load and `.fd` files for some encoding and family (if it exists).

```

418 </2ekernel>
419 <*2ekernel | latexrelease>
420 <latexrelease>\IncludeInRelease{2020/02/02}%
421 <latexrelease>          {\LoadFontDefinitionFile}{Loading .fd files}%
422 \def\LoadFontDefinitionFile#1#2{%
423   \begingroup
424     \edef\f@encoding{#1}%
425     \edef\f@family{#2}%
426     \try@load@fontshape
427   \endgroup
428 }
429 </2ekernel | latexrelease>
430 <latexrelease>\EndIncludeInRelease
431 <latexrelease>\IncludeInRelease{0000/00/00}%
432 <latexrelease>          {\LoadFontDefinitionFile}{Loading .fd files}%
433 <latexrelease>
434 <latexrelease>\let\LoadFontDefinitionFile\@undefined
435 <latexrelease>\EndIncludeInRelease
436 <*2ekernel>

```

(End definition for \LoadFontDefinitionFile.)

`\DeclareFontFamilySubstitution` The idea for this macro is stolen from the `substitutefont` package by Günter Milde, with some modifications and a new name.

Its purpose is to provide characters in a special encoding that are not available in the current font family to be taken from a different family that is visually compatible (or not if you choose badly). For example, you can match the GFS Didot Greek characters with T_EX Gyre Pagella (Palatino) by specifying

```
\DeclareFontFamilySubstitution{LGR}{qpl}{udidot}
```

This way if you ask for the LGR encoding in for the qpl family you get the characters from the udidot family substituted.

We need to ensure that the macro is defined with \nfss@catcodes in force (not quite sure why at the moment to be honest).

```
437 \</2>kernel)
438 \<2>kernel | latexrelease)
439 \<latexrelease>\IncludeInRelease{2020/02/02}%
440 \<latexrelease>      {\DeclareFontFamilySubstitution}{Provide family substitution}%
441 \begingroup
442 \nfss@catcodes
443 \gdef\DeclareFontFamilySubstitution#1#2#3{%
```

We only provide a set of silent substitutions. The package also (re)declared the family, but this is incorrect in my eyes and it is better to handle that differently.

Of course the families may still need loading at this point and so we arrange for this. Otherwise we might run into trouble because the necessary \DeclareFontFamily has not been seen.

```
444 \LoadFontDefinitionFile{#1}{#2}%
445 \LoadFontDefinitionFile{#1}{#3}%
446 \DeclareFontShape{#1}{#2}{m}{it}{<->ssub * #3/m/it}{}%
447 \DeclareFontShape{#1}{#2}{m}{n}{<->ssub * #3/m/n}{}%
448 \DeclareFontShape{#1}{#2}{m}{sc}{<->ssub * #3/m/sc}{}%
449 \DeclareFontShape{#1}{#2}{m}{sl}{<->ssub * #3/m/sl}{}%
```

These days a few more shapes might be around, so we declare those too. If they don't exist then after the first substitution normal fallbacks will happen.

```
450 \DeclareFontShape{#1}{#2}{m}{sw}{<->ssub * #3/m/sw}{}%
451 \DeclareFontShape{#1}{#2}{m}{scit}{<->ssub * #3/m/scit}{}%
452 \DeclareFontShape{#1}{#2}{m}{scsl}{<->ssub * #3/m/scsl}{}%
```

Same game with b and bx, for other weights you are on your own:

```
453 \DeclareFontShape{#1}{#2}{b}{it}{<->ssub * #3/b/it}{}%
454 \DeclareFontShape{#1}{#2}{b}{n}{<->ssub * #3/b/n}{}%
455 \DeclareFontShape{#1}{#2}{b}{scit}{<->ssub * #3/b/scit}{}%
456 \DeclareFontShape{#1}{#2}{b}{scsl}{<->ssub * #3/b/scsl}{}%
457 \DeclareFontShape{#1}{#2}{b}{sc}{<->ssub * #3/b/sc}{}%
458 \DeclareFontShape{#1}{#2}{b}{sl}{<->ssub * #3/b/sl}{}%
459 \DeclareFontShape{#1}{#2}{b}{sw}{<->ssub * #3/b/sw}{}%
460 \DeclareFontShape{#1}{#2}{bx}{it}{<->ssub * #3/bx/it}{}%
461 \DeclareFontShape{#1}{#2}{bx}{n}{<->ssub * #3/bx/n}{}%
462 \DeclareFontShape{#1}{#2}{bx}{scit}{<->ssub * #3/bx/scit}{}%
463 \DeclareFontShape{#1}{#2}{bx}{scsl}{<->ssub * #3/bx/scsl}{}%
464 \DeclareFontShape{#1}{#2}{bx}{sc}{<->ssub * #3/bx/sc}{}%
465 \DeclareFontShape{#1}{#2}{bx}{sl}{<->ssub * #3/bx/sl}{}%
466 \DeclareFontShape{#1}{#2}{bx}{sw}{<->ssub * #3/bx/sw}{}%
467 }
468 \endgroup
469 \</2>kernel | latexrelease)
470 \<latexrelease>\EndIncludeInRelease
471 \<latexrelease>\IncludeInRelease{0000/00/00}%
472 \<latexrelease>      {\DeclareFontFamilySubstitution}{Provide family substitution}%
473 \<latexrelease>
```

```

474 <latexrelease>\let\DeclareFontFamilySubstitution\@undefined
475 <latexrelease>\EndIncludeInRelease
476 <*2ekernel>

```

(End definition for \DeclareFontFamilySubstitution.)

\DeclareErrorFont Declare the last resort shape! We assume that in this fontshape there is a 10pt font but it doesn't really matter. We only loose one macro name if the assumption is false. But at least the font should be there!

```

477 </2ekernel>
478 <*2ekernel | latexrelease>
479 <latexrelease>\IncludeInRelease{2019/10/01}%
480 <latexrelease>          {\DeclareErrorFont}{No side effects please}%
481 \def\DeclareErrorFont#1#2#3#4#5{%
482     \xdef\error@fontshape{%
483         \noexpand\expandafter\noexpand\split@name\noexpand\string
484         \expandafter\noexpand\csname#1/#2/#3/#4/#5\endcsname
485         \noexpand\@nil}%

```

Initialize all those internal variables which may or may not have values in the first seconds of NFSS' bootstrapping process. Later on such values will be updated when an encoding is selected, etc.

We definitely don't want to set \f@encoding; we can set all the others since if they are left "blank" any selection would grab "error default values" as well. However, this probably should go also—and now it did.

```

486 %      \gdef\f@encoding{#1}%
487      \gdef\default@family{#2}%
488      \gdef\default@series{#3}%
489      \gdef\default@shape{#4}%
490 }
491 </2ekernel | latexrelease>
492 <latexrelease>\EndIncludeInRelease
493 <latexrelease>\IncludeInRelease{0000/00/00}%
494 <latexrelease>          {\DeclareErrorFont}{No side effects please}%
495 <latexrelease>
496 <latexrelease>\def\DeclareErrorFont#1#2#3#4#5{%
497 <latexrelease>    \xdef\error@fontshape{%
498 <latexrelease>        \noexpand\expandafter\noexpand\split@name\noexpand\string
499 <latexrelease>        \expandafter\noexpand\csname#1/#2/#3/#4/#5\endcsname
500 <latexrelease>        \noexpand\@nil}%
501 <latexrelease>    \gdef\default@family{#2}%
502 <latexrelease>    \gdef\default@series{#3}%
503 <latexrelease>    \gdef\default@shape{#4}%
504 <latexrelease>    \global\let\f@family\default@family
505 <latexrelease>    \global\let\f@series\default@series
506 <latexrelease>    \global\let\f@shape\default@shape
507 <latexrelease>    \gdef\f@size{#5}%
508 <latexrelease>    \gdef\f@baselineskip{#5pt}%
509 <latexrelease>}
510 <latexrelease>\EndIncludeInRelease
511 <*2ekernel>

```

\@onlypreamble\DeclareErrorFont

(End definition for \DeclareErrorFont.)

`\wrong@fontshape` Before we come to the macro `\extract@font` we have to take care of unknown `\curr@fontshape` combinations. The general strategy is to issue a warning and to try a default *shape*, then a default *series*, and finally a default *family*. If this last one also fails T_EX will go into an infinite loop. But if the defaults are set incorrectly one deserves nothing else!

```

513 \endkernel
514 \latexrelease\IncludeInRelease{2015/01/01}{\wrong@fontshape}%
515 \latexrelease{Font substitution in preamble}%
516 \endkernel\latexrelease
517 \def\wrong@fontshape{%
518   \csname D@f@encoding\endcsname % install defaults if in math

```

We remember the wanted `\curr@fontshape` combination which we will need in a moment.

```

519   \edef\reserved@a{\csname\curr@fontshape\endcsname}%
520   \ifx\last@fontshape\reserved@a
521     \errmessage{Corrupted NFSS tables}%
522     \error@fontshape
523   \else

```

Then we warn the user about the mess and set the shape to its default.

```

524   \let\f@shape\default@shape

```

If the combination is not known, try the default *series*.

```

525   \expandafter\ifx\csname\curr@fontshape\endcsname\relax
526     \let\f@series\default@series

```

If this is still undefined, try the default *family*. Otherwise give up. We never try to change the encoding scheme!

```

527   \expandafter
528   \ifx\csname\curr@fontshape\endcsname\relax
529     \let\f@family\default@family

```

If we change the font family and we are in the preamble then the corresponding `.fd` file may not been loaded yet. Therefore we try this now. Otherwise equating the requested font shape with the finally selected fontshape below will fail and can result in “NFSS tables corrupted”. After begin document that will not happen as all `.fd` files involved in substitution are loaded at `\begin{document}`.

```

530   \begingroup
531   \try@load@fontshape
532   \endgroup
533   \fi \fi
534 \fi

```

At this point a valid `\curr@fontshape` combination must have been found. We inform the user about this fact.

The `\expandafter\string` here stops T_EX adding the space that it usually puts after command names in messages. The similar construction with `\@undefined` just produces ‘undefined’, but saves a few tokens.

`\@wrong@font@char` is locally redefined in `\UseTextSymbol` from its normal (empty) definition, to report the symbol generating the font switch.

```

535   \@font@warning{Font shape '\expandafter\string\reserved@a'
536   \expandafter@gobble\string\@undefined\MessageBreak
537   using '\curr@fontshape' instead\@wrong@font@char}%
538   \global\let\last@fontshape\reserved@a

```

We change `\@defaultsubs` to produce a warning at the end of the document. The macro `\@defaultsubs` is initially `\relax` but gets changed here if some default font substitution happens. It is then executed in `\enddocument`.

```
539 \gdef\@defaultsubs{%
540 \font@warning{Some font shapes were not available, defaults
541 substituted.\@gobbletwo}}%
```

If we substitute a `\curr@fontshape` combination by the default one we don't want the warning to be printed out whenever this (unknown) combination is used. Therefore we globally `\let` the macro corresponding to the wanted combination equal to its substitution. This requires the use of four `\expandafter`'s since `\csname...\endcsname` has to be expanded before `\reserved@a` (i.e. the requested combination), and this must happen before the `\let` is executed.

```
542 \global\expandafter\expandafter\expandafter\let
543 \expandafter\reserved@a
544 \csname\curr@fontshape\endcsname
```

Now we can redefine `\font@name` accordingly. This *must* be done globally since it might occur in the group opened by `\define@newfont`. If we would this definition were local the closing `\endgroup` there would restore the old meaning of `\font@name` and then switch to the wrong font at the end of `\selectfont` although the correct font was loaded.

```
545 \xdef\font@name{%
546 \csname\curr@fontshape/\f@size\endcsname}%
```

The last thing this macro does is to call `\pickup@font` again to load the font if it is not defined yet. At this point this code will loop endlessly if the defaults are not well defined.

```
547 \pickup@font}
548 </2ekernel | latexrelease>
549 <latexrelease>\EndIncludeInRelease
550 <latexrelease>\IncludeInRelease{0000/00/00}{\wrong@fontshape}%
551 <latexrelease> \Font substitution in preamble}%
552 <latexrelease>\def\wrong@fontshape{%
553 <latexrelease> \csname D@f@encoding\endcsname
554 <latexrelease> \edef\reserved@a{\csname\curr@fontshape\endcsname}%
555 <latexrelease> \ifx\last@fontshape\reserved@a
556 <latexrelease> \errmessage{Corrupted NFSS tables}%
557 <latexrelease> \error@fontshape
558 <latexrelease> \else
559 <latexrelease> \let\f@shape\default@shape
560 <latexrelease> \expandafter\ifx\csname\curr@fontshape\endcsname\relax
561 <latexrelease> \let\f@series\default@series
562 <latexrelease> \expandafter
563 <latexrelease> \ifx\csname\curr@fontshape\endcsname\relax
564 <latexrelease> \let\f@family\default@family
565 <latexrelease> \fi \fi
566 <latexrelease> \fi
567 <latexrelease> \font@warning{Font shape
568 <latexrelease> '\expandafter\string\reserved@a'
569 <latexrelease> \expandafter\@gobble\string\@undefined
570 <latexrelease> \MessageBreak
571 <latexrelease> using '\curr@fontshape' instead\@wrong@font@char}%
572 <latexrelease> \global\let\last@fontshape\reserved@a
573 <latexrelease> \gdef\@defaultsubs{%
574 <latexrelease> \font@warning{Some font shapes were not available,
```

```

575 <latexrelease> defaults substituted.\@gobbletwo}}%
576 <latexrelease> \global\expandafter\expandafter\expandafter\let
577 <latexrelease> \expandafter\reserved@a
578 <latexrelease> \csname\curr@fontshape\endcsname
579 <latexrelease> \xdef\font@name{%
580 <latexrelease> \csname\curr@fontshape/\f@size\endcsname}%
581 <latexrelease> \pickup@font}
582 <latexrelease>\EndIncludeInRelease
583 (*2ekernel)

```

(End definition for \wrong@fontshape.)

\@wrong@font@char Normally empty but redefined in \UseTextSymbol so that the Font shape undefined message can refer to the symbol causing the problem.

```
584 \let\@wrong@font@char\@empty
```

(End definition for \@wrong@font@char.)

\@@defaultsubs See above.

```
\@defaultsubs 585 \let\@defaultsubs\relax
```

(End definition for \@@defaultsubs and \@defaultsubs.)

\strip@prefix In \extract@font we will need a way to recover the replacement text of a macro. This is done by the primitive \meaning together with the macro \strip@prefix (for the details see appendix D of the T_EXbook, p. 382).

```
586 \def\strip@prefix#1>{\}
```

(End definition for \strip@prefix.)

4 Assigning math fonts to *versions*

\install@mathalphabet This is just another name for \gdef but we can redefine it if necessary later on.

```
587 \let\install@mathalphabet\gdef
```

(End definition for \install@mathalphabet.)

\math@fonts

```
588 \let\math@fonts\@empty
```

(End definition for \math@fonts.)

\select@group \select@group has four arguments: the new *math alphabet identifier* (a control sequence), the *math group number*, the extra macro for math mode and the \curr@fontshape definition macro name. We first check if we are in math mode.

```
589 %\def\select@group#1#2#3{\relax\ifmmode
```

We do these things locally using \begingroup instead of \bgroup to avoid the appearance of an empty Ord atom on the math list.

```
590 % \begingroup
```

We set the math fonts for the *family* in question by calling `\getanddefine@fonts` in the correct environment.

```
591 % \escapechar\m@ne
592 % \getanddefine@fonts{\csname c@mv@\math@version\endcsname}#3%
```

We globally select the math fonts...

```
593 % \globaldefs\@ne \math@fonts
```

... and close the group to restore `\globaldefs` and `\escapechar`.

```
594 % \endgroup
```

As long as no *size* or *version* change occurs the $\langle\textit{math alphabet identifier}\rangle$ should simply switch to the installed *math group* instead of calling `\select@group` unnecessarily. So we globally redefine the first argument (the new $\langle\textit{math alphabet identifier}\rangle$) to expand into a `\mathgroup` switch and then select this *alphabet*. Note that this redefinition will be overwritten by the next call to a *version* macro. The original code for the end of `\select@group` was

```
\gdef#1{#3\mathgroup #2}#1\fi}
```

i.e. first redefining the $\langle\textit{math alphabet identifier}\rangle$ and then calling the new definition to switch to the wanted $\langle\textit{math group}\rangle$. Now we define the $\langle\textit{math alphabet identifier}\rangle$ as a call to the `\use@mathgroup` command.

```
595 % \xdef#1{\noexpand\use@mathgroup\noexpand#2%
596 % {\number\csname c@mv@\math@version\endcsname}}%
```

But this is not sufficient, as we learned the hard way. The problem here is that the loading of the fonts that comprise the alphabet identifier `#1`, as well as the necessary math font assignments is deferred until it is used. This is OK so far, but if the fonts are switched within the current formula (which may happen if a sub-formula is a box that contains a math version switch) the font assignments for `#1` are not restored unless `#1` is used again. This is disastrous since TeX sees the wrong fonts at the end of the math formula, when it converts the math list into a horizontal list.

This is taken into account as follows: When a math alphabet identifier is used for the first time in a certain version it modifies the corresponding macro `\mv@<version>` so that it calls `\getanddefine@fonts` directly in future as well. We use the macro `\extract@alph@from@version` to do this. It takes the math alphabet identifier `#1` and the math version macro as arguments.

```
597 % \expandafter\extract@alph@from@version
598 % \csname mv@\math@version\expandafter\endcsname
599 % \expandafter{\number\csname c@mv@\math@version\endcsname}%
600 % #1%
601 % \stepcounter{mv@\math@version}%
```

Finally, it is not possible to simply call the new definition since we have an argument (the third argument of `\use@mathgroup` or more exactly the argument of `\math@egroup` if the `margid` option is in force) which would swallow our closing `\fi`. So we use the `\expandafter` technique to remove the `\fi` before the `\use@mathgroup` is expanded.

```
602 %\expandafter #1\fi}
```

(End definition for `\select@group`.)

`\extract@alph@from@version` We proceed to the definition of the macro `\extract@alph@from@version`. As stated above, it takes a math alphabet identifier and a math version macro (e.g. `\mv@normal`) as its arguments.

```
603 \def\extract@alph@from@version#1#2#3{%
```

To extract and replace the definition of math alphabet identifier `#3` in macro `#1` we have to recall how this definition looks like: Somewhere in the replacement text of `#1` there is the sequence

```
\install@mathalphabet<math alphabet identifier> #3{%
  <Definitions for >#3}
```

Hence, the first thing we do is to extract the tokens preceding this definitions, the definition itself, and the tokens following it. To this end we define one auxiliary macro `\reserved@a`.

```
604 \def\reserved@a##1\install@mathalphabet#3##2##3\@nil{%
```

When `\reserved@a` is expanded, it will have the tokens preceding the definition in question in its first argument (`##1`), the following tokens in its third argument (`##3`), and the replacement text for the math alphabet identifier `#3` in its second argument. (`##2`). This is then recorded for later use in a temporary macro `\reserved@b`.

```
605 \def\reserved@b{##2}%
```

Additionally, we define a macro `\reserved@c` to reconstruct the definitions for the math version in question from the tokens that will remain unchanged (`##1` and `##3`) and the yet to build new definitions for the math alphabet identifier `#3`.

```
606 \def\reserved@c####1{\gdef#1{##1####1##3}}%
```

Then we execute our auxiliary macro.

```
607 \expandafter\reserved@a#1\@nil
```

OK, so now we have to build the new definition for `#3`. To do so, we first extract the interesting parts out of the old one. The old definition looks like:

```
\select@group<math alphabet identifier>
  <math group number><math extra part>
<curr@fontshape definition>
```

So we define a new temporary macro `\reserved@a` that extracts these parts.

```
608 \def\reserved@a\select@group#3##1##2\@nil{%
```

This macro can now directly rebuild the math version definition by calling `\reserved@c`:

```
609 \reserved@c{%
610 \getanddefine@fonts{#2}##2%
611 \install@mathalphabet#3{%
612 \relax\ifmmode \else \non@alpherr#3\fi
613 \use@mathgroup##1{#2}}%
```

In addition it defines the alphabet the way it should be used from now on.

```
614 \gdef#3{\relax\ifmmode \else \non@alpherr#3\fi
615 \use@mathgroup##1{#2}}%
```

Finally, we only have to call this macro `\reserved@a` on the old definitions recorded in `\reserved@b`:

```
616 \expandafter\reserved@a\reserved@b\@nil
617 }
```

(End definition for `\extract@alph@from@version`.)

`\math@bgroup` Here are the default definitions for `\math@bgroup` and `\math@egroup`. We use `\bgroup`
`\math@egroup` instead of `\begingroup` to avoid ‘leaking out’ of style changes. This has the side effect
of always producing mathord atoms.

```
618 \let\math@bgroup\bgroup
619 \def\math@egroup#1{#1\egroup}
```

(End definition for `\math@bgroup` and `\math@egroup`.)

`\calculate@math@sizes` Here is the default definition for `\calculate@math@sizes` a more elaborate interface is
under testing in `mtscale.sty`.

```
620 \gdef\calculate@math@sizes{%
621   \@font@info{Calculating\space math\space sizes\space for\space
622     size\space <\f@size>}%
623   \dimen@f@size \p@
624   \@tempdimb \defaultscriptratio \dimen@
625   \dimen@ \defaultscriptscriptratio \dimen@
626   \expandafter\xdef\csname S@\f@size\endcsname{%
627     \gdef\noexpand\tf@size{\f@size}%
628     \gdef\noexpand\sf@size{\strip@pt\@tempdimb}%
629     \gdef\noexpand\ssf@size{\strip@pt\dimen@}%
630     \noexpand\math@fontstrue}}
```

(End definition for `\calculate@math@sizes`.)

`\defaultscriptratio` The default ratio for math sizes is:
`\defaultscriptscriptratio` 1 to `\defaultscriptratio` to `\defaultscriptscriptratio`.
By default this is 1 to .7 to .5.

```
631 \def\defaultscriptratio{.7}
632 \def\defaultscriptscriptratio{.5}
```

(End definition for `\defaultscriptratio` and `\defaultscriptscriptratio`.)

`\noaccents@` If we don’t have a definition for `\noaccents@` we provide a dummy.

```
633 \ifx\noaccents@\@undefined
634   \let\noaccents@\@empty
635 \fi
```

(End definition for `\noaccents@`.)

`\showhyphens` The `\showhyphens` command must be redefined since the version in `plain.tex` uses
`\tenrm`. We have also made some further adjustments for its use in \LaTeX .

```
636 \if2ekernel
637   \latexrelease\IncludeInRelease{2017/01/01}{\showhyphens}%
638   \latexrelease{XeTeX support for \showhyphens}%
639   \if2ekernel\latexrelease
640   \ifx\XeTeXcharclass\@undefined
```

Version for engines other than \LaTeX .

```
641 \DeclareRobustCommand\showhyphens[1]{%
642   \setbox0\vbox{%
643     \color@begingroup
644     \everypar{}%
645     \parfillskip\z@skip\hsize\maxdimen
646     \normalfont
647     \pretolerance\m@ne\tolerance\m@ne\hbadness\z@\showboxdepth\z@\ #1%
648     \color@endgroup}}
```

649 `\else`

XeTeX version. When using system fonts XeTeX reports consecutive runs of characters as a single item in box logging, which means the standard `\showhyphens` does not work. This version typesets the text into a narrow box to force hyphenation and then reconstructs a horizontal list with explicit hyphens to generate the display. Note that the `lmr` OpenType font is forced, this works even if the characters are not in the font as hyphenation is attempted due to the width of the space and hyphen character. It may generate spurious Missing Character warnings in the log, these are however suppressed from the terminal output by ensuring that `\tracingonline` is locally zero.

```
650 \DeclareRobustCommand\showhyphens[1]{%
651   \setbox0\vbox{%
652     \usefont{TU}{lmr}{m}{n}%
653     \hsize 1sp %
654     \hbadness\@M
655     \hfuzz\maxdimen
656     \tracingonline\z@
657     \everypar={}%
658     \leftskip\z@skip
659     \rightskip\z@skip
660     \parfillskip\z@skip
661     \hyphenpenalty=-\@M
662     \pretolerance\m@ne
663     \interlinepenalty\z@
664     \clubpenalty\z@
665     \widowpenalty\z@
666     \brokenpenalty1127 %
667     \setbox\z@\hbox{}%
668     \noindent
669     \hskip\z@skip
670     #1%
671     \par
```

Note here we stop the loop if made no progress, non-removable items may mean that we can not process the whole list (which would be testable as `\lastnodetype=-1`).

```
672   \loop
673   \@tempswafalse
674   \ifnum\lastnodetype=11\unskip\@tempswatrue\fi
675   \ifnum\lastnodetype=12\unkern\@tempswatrue\fi
676   \ifnum\lastnodetype=13 %
677     \count@\lastpenalty
678     \unpenalty\@tempswatrue
679   \fi
680   \ifnum\lastnodetype=\@ne
681     \setbox\tw@\lastbox\@tempswatrue
682     \setbox0\hbox{\unhbox\tw@\unskip\unskip\unpenalty
683                   \ifnum\count@=1127 \else\ \fi
684                   \unhbox0}%
685     \count@\z@
686   \fi
687   \if@tempswa
688     \repeat
689   \hbadness\z@
690   \hsize\maxdimen
```

```

691 \showboxdepth\z@
692 \tolerance\m@ne
693 \hyphenpenalty\z@
694 \noindent\unhbox\z@
695 }}

696 \fi

697 </2ekernel | latexrelease>
698 <latexrelease>\EndIncludeInRelease
699 <latexrelease>\IncludeInRelease{0000/00/00}{\showhyphens}%
700 <latexrelease> {XeTeX support for \showhyphens}%
701 <latexrelease>\gdef\showhyphens#1{%
702 <latexrelease> \setbox0\vbox{%
703 <latexrelease> \color@begingroup
704 <latexrelease> \everypar{}%
705 <latexrelease> \parfillskip\z@skip\hsize\maxdimen
706 <latexrelease> \normalfont
707 <latexrelease> \pretolerance\m@ne\tolerance\m@ne
708 <latexrelease> \hbadness\z@\showboxdepth\z@\ #1%
709 <latexrelease> \color@endgroup}}
710 <latexrelease>\EndIncludeInRelease
711 <*2ekernel>

```

(End definition for \showhyphens.)

\addto@hook We need a macro to add tokens to a hook.

```

712 \long\def\addto@hook#1#2{#1\expandafter{\the#1#2}}

```

(End definition for \addto@hook.)

\@vpt

```

713 \def\@vpt{5}

```

(End definition for \@vpt.)

\@vipt

```

714 \def\@vipt{6}

```

(End definition for \@vipt.)

\@viipt

```

715 \def\@viipt{7}

```

(End definition for \@viipt.)

\@viipt

```

716 \def\@viipt{8}

```

(End definition for \@viipt.)

\@ixpt

```

717 \def\@ixpt{9}

```

(End definition for \@ixpt.)


```

\@xpt
718 \def\@xpt{10}
      (End definition for \@xpt.)

\@xipt
719 \def\@xipt{10.95}
      (End definition for \@xipt.)

\@xiipt
720 \def\@xiipt{12}
      (End definition for \@xiipt.)

\@xivpt
721 \def\@xivpt{14.4}
      (End definition for \@xivpt.)

\@xvipt
722 \def\@xvipt{17.28}
      (End definition for \@xvipt.)

\@xxpt
723 \def\@xxpt{20.74}
      (End definition for \@xxpt.)

\@xxvpt
724 \def\@xxvpt{24.88}
      (End definition for \@xxvpt.)
725 \</2ekernel>

```

File v

ltfssaxes.dtx

This file contains the implementation for handling extra axes splitting the series and the values into sub-categories. selection commands. See other parts of the L^AT_EX distribution, or *The L^AT_EX Companion* for higher level documentation of the L^AT_EX Font Selection Scheme.

Everything in the this file got introduced 2020/02/02, so we use large rollback chunks, only interrupted if necessary.

```
1 <*2ekernel | latexrelease>
2 <latexrelease> \IncludeInRelease{2020/02/02}%
3 <latexrelease>   {\DeclareFontSeriesChangeRule}{Series change rules}%
```

1 Changing the font series

In the original NFSS implementation the series was a single attribute stored in `\f@series` and so one always had to specify both weight and width together. This means it was impossible to typeset, a paragraph in a condensed font and inside have a few words in bold weight (but still condensed) without doing this manually by requesting `\fontseries{bc}\selectfont`.

The new implementation now works differently by looking both at the current value of `\f@series` and the requested new series and out of that combination selects a resulting series value. Thus, if the current series is `c` and we ask for `b` we now get `bc`.

This is done by consulting a simple lookup table. This table is configurable (though most likely that flexibility will seldom of ever be needed) Adding or changing entries in this table are done with `\DeclareFontSeriesChangeRule`.

1.1 The series lookup table

`\DeclareFontSeriesChangeRule` The `\DeclareFontSeriesChangeRule` defines entries in a simple database (implemented as a set of commands) that define mappings between from an existing series and requested new series and maps that to a result series (and additionally offers an alternative if the desired one is not existing):

```
#1 current \f@series
#2 requested new series
#3 result (if that exist for the given font family
#4 alternative result (if #3 does not exist)
```

If an `.fd` file has its own substitution rules then `#3` exist and thus `#4` is not applied.

If there is no matching database entry or if neither the result nor the alternate result exist in the font family the requested new series is used (which then may trigger substitutions later on.

```
4 \def\DeclareFontSeriesChangeRule#1#2#3#4{%
5   \@namedef{series@#1@#2}{#{#3}#{#4}}}
```

(End definition for `\DeclareFontSeriesChangeRule`.)

1.2 Mapping rules for series changes

The rules set up use explicit series values not `\..default` indirections; my current feeling is that this is in fact better.

With 9 weights and 9 width classes this table is getting a bit large in the end (324 entries) but on the other hand it doesn't change and accessing speed and it is fast this way.

We could alternatively split the axis and maintain weight and width separately, but that would take more processing time and would not allow for setting up explicit exceptions nicely (not sure that this would ever get used though).

Design considerations for mapping entries:

- We make `m` to reset both weight and width (as this is how it always worked). To reset just the width `?m` is provided and to reset just the weight `m?`.
- We do support “*mwidth*” and “*weightm*”, e.g., `mec` to mean “go to medium weight and extra-condensed width”. At the end of the process we automatically drop any leftover `m` in the series name (unless it is just a single `m`).
- If there is no table entry then the target series is used unconditionally. This means that any request to set both weight and width (e.g. `bx` or `ulc`) needs no table entries. For that reason there are no entries which have a weight+width as request (i.e., second argument).

In particular this is also true for cases involving `m`, e.g., `bm` (bold medium width) which automatically gets reduced result in `b` or `mc` (medium weight condensed) which becomes `c` as a result.

- Only a few entries have “alternative” values and perhaps most of them should get dropped. Or maybe not ... needs some thought perhaps.

The idea is that you don't want the normal substitution to kick in because that would reset the shape first and it may be better to stay with `b` when a change to `c` is requested and `bc` doesn't exist, than to go to first change the shape to `n` and then find that `bc/n` doesn't exist either and thus ending up with `m/n`.

- Also: while I did set up all nine standard weight values from `ul` to `ub` I only bothered to provide entries for `ec`, `sc`, `c` and `x`, because other levels of compression/expansion are not in any real fonts that I know.

Could and perhaps should be eventually extended to cover the whole set.

```
6 \DeclareFontSeriesChangeRule {bc}{b}{bc}{}
7 \DeclareFontSeriesChangeRule {bc}{c}{bc}{}
8 \DeclareFontSeriesChangeRule {bc}{eb}{ebc}{}
9 \DeclareFontSeriesChangeRule {bc}{ec}{bec} {bc}
10 \DeclareFontSeriesChangeRule {bc}{el}{elc}{}
11 \DeclareFontSeriesChangeRule {bc}{l}{lc}{}
12 \DeclareFontSeriesChangeRule {bc}{sb}{sbc}{}
13 \DeclareFontSeriesChangeRule {bc}{sc}{bsc} {bc}
14 \DeclareFontSeriesChangeRule {bc}{sl}{slc}{}
15 \DeclareFontSeriesChangeRule {bc}{ub}{ubc}{}
16 \DeclareFontSeriesChangeRule {bc}{ul}{ulc}{}
17 \DeclareFontSeriesChangeRule {bc}{x}{bx}{}

```

```

18 \DeclareFontSeriesChangeRule {bx}{b}{bx}{}
19 \DeclareFontSeriesChangeRule {bx}{c} {bc} {bx} %<-----
20 \DeclareFontSeriesChangeRule {bx}{eb}{ebx}{}
21 \DeclareFontSeriesChangeRule {bx}{ec} {bec} {bx} %<-----
22 \DeclareFontSeriesChangeRule {bx}{el}{elx}{}
23 \DeclareFontSeriesChangeRule {bx}{l}{lx}{}
24 \DeclareFontSeriesChangeRule {bx}{sb} {sbx} {}
25 \DeclareFontSeriesChangeRule {bx}{sc} {bsc} {bx} %<-----
26 \DeclareFontSeriesChangeRule {bx}{sl}{slx} {}
27 \DeclareFontSeriesChangeRule {bx}{ub}{ubx}{}
28 \DeclareFontSeriesChangeRule {bx}{ul}{ulx}{}
29 \DeclareFontSeriesChangeRule {bx}{x}{bx}{}

30 \DeclareFontSeriesChangeRule {b}{bx} {bx} {b} %<-----
31 \DeclareFontSeriesChangeRule {b}{c} {bc} {b} %<-----
32 \DeclareFontSeriesChangeRule {b}{ec} {bec} {b} %<-----
33 \DeclareFontSeriesChangeRule {b}{sb} {sb} {b} %<-----
34 \DeclareFontSeriesChangeRule {b}{sc} {bsc} {b} %<-----
35 \DeclareFontSeriesChangeRule {b}{x} {bx} {b} %<-----

36 \DeclareFontSeriesChangeRule {c}{bx} {bx} {b} %<-----
37 \DeclareFontSeriesChangeRule {c}{b}{bc}{}
38 \DeclareFontSeriesChangeRule {c}{eb}{ebc}{}
39 \DeclareFontSeriesChangeRule {c}{el}{elc}{}
40 \DeclareFontSeriesChangeRule {c}{l}{lc}{}
41 \DeclareFontSeriesChangeRule {c}{sb}{sbc}{}
42 \DeclareFontSeriesChangeRule {c}{sl}{slc}{}
43 \DeclareFontSeriesChangeRule {c}{ub}{ubc}{}
44 \DeclareFontSeriesChangeRule {c}{ul}{ulc}{}
45 \DeclareFontSeriesChangeRule {c}{x}{x}{m} %<-----

46 \DeclareFontSeriesChangeRule {ebc}{b}{bc}{}
47 \DeclareFontSeriesChangeRule {ebc}{c}{ebc}{}
48 \DeclareFontSeriesChangeRule {ebc}{eb}{ebc}{}
49 \DeclareFontSeriesChangeRule {ebc}{ec}{ebec}{ebc}
50 \DeclareFontSeriesChangeRule {ebc}{el}{elc}{}
51 \DeclareFontSeriesChangeRule {ebc}{l}{lc}{}
52 \DeclareFontSeriesChangeRule {ebc}{sb}{sbc}{}
53 \DeclareFontSeriesChangeRule {ebc}{sc}{ebsc}{ebc}
54 \DeclareFontSeriesChangeRule {ebc}{sl}{slc}{}
55 \DeclareFontSeriesChangeRule {ebc}{ub}{ubc}{}
56 \DeclareFontSeriesChangeRule {ebc}{ul}{ulc}{}
57 \DeclareFontSeriesChangeRule {ebc}{x}{ebx}{}

58 \DeclareFontSeriesChangeRule {ec}{bx} {bx} {b} %<-----
59 \DeclareFontSeriesChangeRule {ec}{b}{bec}{}
60 \DeclareFontSeriesChangeRule {ec}{eb}{ebec}{}
61 \DeclareFontSeriesChangeRule {ec}{el}{elec}{}
62 \DeclareFontSeriesChangeRule {ec}{l}{lec}{}
63 \DeclareFontSeriesChangeRule {ec}{sb}{sbec}{}
64 \DeclareFontSeriesChangeRule {ec}{sl}{slec}{}
65 \DeclareFontSeriesChangeRule {ec}{ub}{ubec}{}
66 \DeclareFontSeriesChangeRule {ec}{ul}{ulec}{}
67 \DeclareFontSeriesChangeRule {ec}{x}{x}{m} %<-----

68 \DeclareFontSeriesChangeRule {sc}{bx} {bx} {b} %<-----
69 \DeclareFontSeriesChangeRule {sc}{b}{bsc}{}

```

```

70 \DeclareFontSeriesChangeRule {sc}{eb}{ebsc}{}
71 \DeclareFontSeriesChangeRule {sc}{el}{elsc}{}
72 \DeclareFontSeriesChangeRule {sc}{l}{lsc}{}
73 \DeclareFontSeriesChangeRule {sc}{sb}{sbsc}{}
74 \DeclareFontSeriesChangeRule {sc}{sl}{slsc}{}
75 \DeclareFontSeriesChangeRule {sc}{ub}{ubsc}{}
76 \DeclareFontSeriesChangeRule {sc}{ul}{ulsc}{}
77 \DeclareFontSeriesChangeRule {sc}{x}{x}{m} %<-----

78 \DeclareFontSeriesChangeRule {ebx}{b}{bx}{}
79 \DeclareFontSeriesChangeRule {ebx}{c}{ebc}{}
80 \DeclareFontSeriesChangeRule {ebx}{eb}{ebx}{}
81 \DeclareFontSeriesChangeRule {ebx}{ec}{ebec}{}
82 \DeclareFontSeriesChangeRule {ebx}{el}{elx}{}
83 \DeclareFontSeriesChangeRule {ebx}{l}{lx}{}
84 \DeclareFontSeriesChangeRule {ebx}{sb}{sbx}{}
85 \DeclareFontSeriesChangeRule {ebx}{sc}{ebsc}{}
86 \DeclareFontSeriesChangeRule {ebx}{sl}{slx}{}
87 \DeclareFontSeriesChangeRule {ebx}{ub}{ubx}{}
88 \DeclareFontSeriesChangeRule {ebx}{ul}{ulx}{}
89 \DeclareFontSeriesChangeRule {ebx}{x}{ebx}{}

90 \DeclareFontSeriesChangeRule {eb}{c}{ebc}{}
91 \DeclareFontSeriesChangeRule {eb}{ec}{ebec}{}
92 \DeclareFontSeriesChangeRule {eb}{sc}{ebsc}{}
93 \DeclareFontSeriesChangeRule {eb}{x}{ebx}{}

94 \DeclareFontSeriesChangeRule {elc}{b}{bc}{}
95 \DeclareFontSeriesChangeRule {elc}{c}{elc}{}
96 \DeclareFontSeriesChangeRule {elc}{eb}{ebc}{}
97 \DeclareFontSeriesChangeRule {elc}{ec}{elec}{}
98 \DeclareFontSeriesChangeRule {elc}{el}{elc}{}
99 \DeclareFontSeriesChangeRule {elc}{l}{lc}{}
100 \DeclareFontSeriesChangeRule {elc}{sb}{sbc}{}
101 \DeclareFontSeriesChangeRule {elc}{sc}{elsc}{}
102 \DeclareFontSeriesChangeRule {elc}{sl}{slc}{}
103 \DeclareFontSeriesChangeRule {elc}{ub}{ubc}{}
104 \DeclareFontSeriesChangeRule {elc}{ul}{ulc}{}
105 \DeclareFontSeriesChangeRule {elc}{x}{elx}{}

106 \DeclareFontSeriesChangeRule {elx}{b}{bx}{}
107 \DeclareFontSeriesChangeRule {elx}{c}{elc}{}
108 \DeclareFontSeriesChangeRule {elx}{eb}{ebx}{}
109 \DeclareFontSeriesChangeRule {elx}{ec}{elec}{}
110 \DeclareFontSeriesChangeRule {elx}{el}{elx}{}
111 \DeclareFontSeriesChangeRule {elx}{l}{lx}{}
112 \DeclareFontSeriesChangeRule {elx}{sb}{sbx}{}
113 \DeclareFontSeriesChangeRule {elx}{sc}{elsc}{}
114 \DeclareFontSeriesChangeRule {elx}{sl}{slx}{}
115 \DeclareFontSeriesChangeRule {elx}{ub}{ubx}{}
116 \DeclareFontSeriesChangeRule {elx}{ul}{ulx}{}
117 \DeclareFontSeriesChangeRule {elx}{x}{elx}{}

118 \DeclareFontSeriesChangeRule {el}{c}{elc}{}
119 \DeclareFontSeriesChangeRule {el}{ec}{elec}{}
120 \DeclareFontSeriesChangeRule {el}{sc}{elsc}{}
121 \DeclareFontSeriesChangeRule {el}{x}{elx}{}

```

```

122 \DeclareFontSeriesChangeRule {lc}{b}{bc}{}
123 \DeclareFontSeriesChangeRule {lc}{c}{lc}{}
124 \DeclareFontSeriesChangeRule {lc}{eb}{ebc}{}
125 \DeclareFontSeriesChangeRule {lc}{ec}{lec}{}
126 \DeclareFontSeriesChangeRule {lc}{el}{elc}{}
127 \DeclareFontSeriesChangeRule {lc}{l}{lc}{}
128 \DeclareFontSeriesChangeRule {lc}{sb}{sbc}{}
129 \DeclareFontSeriesChangeRule {lc}{sc}{lsc}{}
130 \DeclareFontSeriesChangeRule {lc}{sl}{slc}{}
131 \DeclareFontSeriesChangeRule {lc}{ub}{ubc}{}
132 \DeclareFontSeriesChangeRule {lc}{ul}{ulc}{}
133 \DeclareFontSeriesChangeRule {lc}{x}{lx}{}

134 \DeclareFontSeriesChangeRule {lx}{b}{bx}{}
135 \DeclareFontSeriesChangeRule {lx}{c}{lc}{}
136 \DeclareFontSeriesChangeRule {lx}{eb}{ebx}{}
137 \DeclareFontSeriesChangeRule {lx}{ec}{lec}{}
138 \DeclareFontSeriesChangeRule {lx}{el}{elx}{}
139 \DeclareFontSeriesChangeRule {lx}{l}{lx}{}
140 \DeclareFontSeriesChangeRule {lx}{sb}{sbx}{}
141 \DeclareFontSeriesChangeRule {lx}{sc}{lsc}{}
142 \DeclareFontSeriesChangeRule {lx}{sl}{slx}{}
143 \DeclareFontSeriesChangeRule {lx}{ub}{ubx}{}
144 \DeclareFontSeriesChangeRule {lx}{ul}{ulx}{}
145 \DeclareFontSeriesChangeRule {lx}{x}{lx}{}

146 \DeclareFontSeriesChangeRule {l}{bx} {bx} {b} %<-----
147 \DeclareFontSeriesChangeRule {l}{b} {b} {bx} %<-----
148 \DeclareFontSeriesChangeRule {l}{c} {lc} {l} % ? %<-----
149 \DeclareFontSeriesChangeRule {l}{ec} {lec} {l} % ? %<-----
150 \DeclareFontSeriesChangeRule {l}{sb} {sb} {b} % ? %<-----
151 \DeclareFontSeriesChangeRule {l}{sc} {lsc} {l} % ? %<-----
152 \DeclareFontSeriesChangeRule {l}{x} {lx} {l} % ? %<-----

153 \DeclareFontSeriesChangeRule {m}{bx} {bx} {b} %<-----
154 \DeclareFontSeriesChangeRule {m}{b} {b} {bx} %<-----
155 \DeclareFontSeriesChangeRule {m}{c} {c} {m} %<-----
156 \DeclareFontSeriesChangeRule {m}{ec} {ec} {m} %<-----
157 \DeclareFontSeriesChangeRule {m}{l} {l} {m} %<-----
158 \DeclareFontSeriesChangeRule {m}{sb} {sb} {b} %<-----
159 \DeclareFontSeriesChangeRule {m}{sc} {sc} {m} %<-----
160 \DeclareFontSeriesChangeRule {m}{x} {x} {m} %<-----

161 \DeclareFontSeriesChangeRule {sbc}{b}{bc}{}
162 \DeclareFontSeriesChangeRule {sbc}{c}{sbc}{}
163 \DeclareFontSeriesChangeRule {sbc}{eb}{ebc}{}
164 \DeclareFontSeriesChangeRule {sbc}{ec}{sbec}{sbc}
165 \DeclareFontSeriesChangeRule {sbc}{el}{elc}{}
166 \DeclareFontSeriesChangeRule {sbc}{l}{lc}{}
167 \DeclareFontSeriesChangeRule {sbc}{sb}{sbc}{}
168 \DeclareFontSeriesChangeRule {sbc}{sc}{sbsc}{sbc}
169 \DeclareFontSeriesChangeRule {sbc}{sl}{slc}{}
170 \DeclareFontSeriesChangeRule {sbc}{ub}{ubc}{}
171 \DeclareFontSeriesChangeRule {sbc}{ul}{ulc}{}
172 \DeclareFontSeriesChangeRule {sbc}{x}{sbx}{}

173 \DeclareFontSeriesChangeRule {sbx}{b}{bx}{}

```

```

174 \DeclareFontSeriesChangeRule {sbx}{c}{sbc}{ }
175 \DeclareFontSeriesChangeRule {sbx}{eb}{ebx}{ }
176 \DeclareFontSeriesChangeRule {sbx}{ec}{sbec}{ }
177 \DeclareFontSeriesChangeRule {sbx}{el}{elx}{ }
178 \DeclareFontSeriesChangeRule {sbx}{l}{lx}{ }
179 \DeclareFontSeriesChangeRule {sbx}{sb}{sbsb}{ }
180 \DeclareFontSeriesChangeRule {sbx}{sc}{sbsc}{ }
181 \DeclareFontSeriesChangeRule {sbx}{sl}{slx}{ }
182 \DeclareFontSeriesChangeRule {sbx}{ub}{ubx}{ }
183 \DeclareFontSeriesChangeRule {sbx}{ul}{ulx}{ }
184 \DeclareFontSeriesChangeRule {sbx}{x}{sbx}{ }

185 \DeclareFontSeriesChangeRule {sb}{c} {sbc} {bc} %? %<-----
186 \DeclareFontSeriesChangeRule {sb}{ec} {sbec} {sbc} %? %<-----
187 \DeclareFontSeriesChangeRule {sb}{sc} {sbsc} {sbc} %? %<-----
188 \DeclareFontSeriesChangeRule {sb}{x} {sbx} {bx} %? %<-----

189 \DeclareFontSeriesChangeRule {slc}{b}{bc}{ }
190 \DeclareFontSeriesChangeRule {slc}{c}{slc}{ }
191 \DeclareFontSeriesChangeRule {slc}{eb}{ebc}{ }
192 \DeclareFontSeriesChangeRule {slc}{ec}{slec}{ }
193 \DeclareFontSeriesChangeRule {slc}{el}{elc}{ }
194 \DeclareFontSeriesChangeRule {slc}{l}{lc}{ }
195 \DeclareFontSeriesChangeRule {slc}{sb}{sbc}{ }
196 \DeclareFontSeriesChangeRule {slc}{sc}{slsc}{ }
197 \DeclareFontSeriesChangeRule {slc}{sl}{slc}{ }
198 \DeclareFontSeriesChangeRule {slc}{ub}{ubc}{ }
199 \DeclareFontSeriesChangeRule {slc}{ul}{ulc}{ }
200 \DeclareFontSeriesChangeRule {slc}{x}{slx}{ }

201 \DeclareFontSeriesChangeRule {slx}{b}{bx}{ }
202 \DeclareFontSeriesChangeRule {slx}{c}{slc}{ }
203 \DeclareFontSeriesChangeRule {slx}{eb}{ebx}{ }
204 \DeclareFontSeriesChangeRule {slx}{ec}{slec}{ }
205 \DeclareFontSeriesChangeRule {slx}{el}{elx}{ }
206 \DeclareFontSeriesChangeRule {slx}{l}{lx}{ }
207 \DeclareFontSeriesChangeRule {slx}{sb}{sbsb}{ }
208 \DeclareFontSeriesChangeRule {slx}{sc}{slsc}{ }
209 \DeclareFontSeriesChangeRule {slx}{sl}{slx}{ }
210 \DeclareFontSeriesChangeRule {slx}{ub}{ubx}{ }
211 \DeclareFontSeriesChangeRule {slx}{ul}{ulx}{ }
212 \DeclareFontSeriesChangeRule {slx}{x}{slx}{ }

213 \DeclareFontSeriesChangeRule {sl}{c}{slc}{ }
214 \DeclareFontSeriesChangeRule {sl}{ec}{slec}{ }
215 \DeclareFontSeriesChangeRule {sl}{sc}{slsc}{ }
216 \DeclareFontSeriesChangeRule {sl}{x}{slx}{ }

217 \DeclareFontSeriesChangeRule {ubc}{b}{bc}{ }
218 \DeclareFontSeriesChangeRule {ubc}{c}{ubc}{ }
219 \DeclareFontSeriesChangeRule {ubc}{eb}{ebc}{ }
220 \DeclareFontSeriesChangeRule {ubc}{ec}{ubec}{ }
221 \DeclareFontSeriesChangeRule {ubc}{el}{elc}{ }
222 \DeclareFontSeriesChangeRule {ubc}{l}{lc}{ }
223 \DeclareFontSeriesChangeRule {ubc}{sb}{sbc}{ }
224 \DeclareFontSeriesChangeRule {ubc}{sc}{ubsc}{ }
225 \DeclareFontSeriesChangeRule {ubc}{sl}{slc}{ }

```

```

226 \DeclareFontSeriesChangeRule {ubc}{ub}{ubc}{}
227 \DeclareFontSeriesChangeRule {ubc}{ul}{ulc}{}
228 \DeclareFontSeriesChangeRule {ubc}{x}{ubx}{}

229 \DeclareFontSeriesChangeRule {ubx}{b}{bx}{}
230 \DeclareFontSeriesChangeRule {ubx}{c}{ubc}{}
231 \DeclareFontSeriesChangeRule {ubx}{eb}{ebx}{}
232 \DeclareFontSeriesChangeRule {ubx}{ec}{ubec}{}
233 \DeclareFontSeriesChangeRule {ubx}{el}{elx}{}
234 \DeclareFontSeriesChangeRule {ubx}{l}{lx}{}
235 \DeclareFontSeriesChangeRule {ubx}{sb}{sbx}{}
236 \DeclareFontSeriesChangeRule {ubx}{sc}{ubsc}{}
237 \DeclareFontSeriesChangeRule {ubx}{sl}{slx}{}
238 \DeclareFontSeriesChangeRule {ubx}{ub}{ubx}{}
239 \DeclareFontSeriesChangeRule {ubx}{ul}{ulx}{}
240 \DeclareFontSeriesChangeRule {ubx}{x}{ubx}{}

241 \DeclareFontSeriesChangeRule {ub}{c}{ubc}{}
242 \DeclareFontSeriesChangeRule {ub}{ec}{ubec}{}
243 \DeclareFontSeriesChangeRule {ub}{sc}{ubsc}{}
244 \DeclareFontSeriesChangeRule {ub}{x}{ubx}{}

245 \DeclareFontSeriesChangeRule {ulc}{b}{bc}{}
246 \DeclareFontSeriesChangeRule {ulc}{c}{ulc}{}
247 \DeclareFontSeriesChangeRule {ulc}{eb}{ebc}{}
248 \DeclareFontSeriesChangeRule {ulc}{ec}{ulec}{ulc}
249 \DeclareFontSeriesChangeRule {ulc}{el}{elc}{}
250 \DeclareFontSeriesChangeRule {ulc}{l}{lc}{}
251 \DeclareFontSeriesChangeRule {ulc}{sb}{sbc}{}
252 \DeclareFontSeriesChangeRule {ulc}{sc}{ulsc}{ulc}
253 \DeclareFontSeriesChangeRule {ulc}{sl}{slc}{}
254 \DeclareFontSeriesChangeRule {ulc}{ub}{ubc}{}
255 \DeclareFontSeriesChangeRule {ulc}{ul}{ulc}{}
256 \DeclareFontSeriesChangeRule {ulc}{x}{ulx}{}

257 \DeclareFontSeriesChangeRule {ulx}{b}{bx}{}
258 \DeclareFontSeriesChangeRule {ulx}{c}{ulc}{}
259 \DeclareFontSeriesChangeRule {ulx}{eb}{ebx}{}
260 \DeclareFontSeriesChangeRule {ulx}{ec}{ulec}{}
261 \DeclareFontSeriesChangeRule {ulx}{el}{elx}{}
262 \DeclareFontSeriesChangeRule {ulx}{l}{lx}{}
263 \DeclareFontSeriesChangeRule {ulx}{sb}{sbx}{}
264 \DeclareFontSeriesChangeRule {ulx}{sc}{ulsc}{}
265 \DeclareFontSeriesChangeRule {ulx}{sl}{slx}{}
266 \DeclareFontSeriesChangeRule {ulx}{ub}{ubx}{}
267 \DeclareFontSeriesChangeRule {ulx}{ul}{ulx}{}
268 \DeclareFontSeriesChangeRule {ulx}{x}{ulx}{}

269 \DeclareFontSeriesChangeRule {ul}{c}{ulc}{}
270 \DeclareFontSeriesChangeRule {ul}{ec}{ulec}{}
271 \DeclareFontSeriesChangeRule {ul}{sc}{ulsc}{}
272 \DeclareFontSeriesChangeRule {ul}{x}{ulx}{}

273 \DeclareFontSeriesChangeRule {x}{b}{bx}{}
274 \DeclareFontSeriesChangeRule {x}{c}{c}{}
275 \DeclareFontSeriesChangeRule {x}{eb}{ebx}{}
276 \DeclareFontSeriesChangeRule {x}{ec}{ec}{}
277 \DeclareFontSeriesChangeRule {x}{el}{elx}{}

```



```

278 \DeclareFontSeriesChangeRule {x}{l}{lx}{}
279 \DeclareFontSeriesChangeRule {x}{sb}{sbx}{}
280 \DeclareFontSeriesChangeRule {x}{sc}{sc}{}
281 \DeclareFontSeriesChangeRule {x}{sl}{slx}{}
282 \DeclareFontSeriesChangeRule {x}{ub}{ubx}{}
283 \DeclareFontSeriesChangeRule {x}{ul}{ulx}{}

```

Special rules for `lm` etc. aren't needed because if the target `lm` is request it will used if there is no rule and that id then reduced to `l` automatically. Same for `mc` and friends. Only `?m` and `m?` need rules.

So here are the special rules for `m?`:

```

284 \DeclareFontSeriesChangeRule {bc}{m?}{c}{}
285 \DeclareFontSeriesChangeRule {bec}{m?}{ec}{}
286 \DeclareFontSeriesChangeRule {bsc}{m?}{sc}{}
287 \DeclareFontSeriesChangeRule {bx}{m?}{x}{}
288 \DeclareFontSeriesChangeRule {b}{m?}{m}{}
289 \DeclareFontSeriesChangeRule {c}{m?}{c}{}
290 \DeclareFontSeriesChangeRule {ebc}{m?}{c}{}
291 \DeclareFontSeriesChangeRule {ebec}{m?}{ec}{}
292 \DeclareFontSeriesChangeRule {ebsc}{m?}{sc}{}
293 \DeclareFontSeriesChangeRule {ebx}{m?}{x}{}
294 \DeclareFontSeriesChangeRule {eb}{m?}{m}{}
295 \DeclareFontSeriesChangeRule {ec}{m?}{ec}{}
296 \DeclareFontSeriesChangeRule {elc}{m?}{c}{}
297 \DeclareFontSeriesChangeRule {elec}{m?}{ec}{}
298 \DeclareFontSeriesChangeRule {elsc}{m?}{sc}{}
299 \DeclareFontSeriesChangeRule {elx}{m?}{x}{}
300 \DeclareFontSeriesChangeRule {el}{m?}{m}{}
301 \DeclareFontSeriesChangeRule {lc}{m?}{c}{}
302 \DeclareFontSeriesChangeRule {lec}{m?}{ec}{}
303 \DeclareFontSeriesChangeRule {lsc}{m?}{sc}{}
304 \DeclareFontSeriesChangeRule {lx}{m?}{x}{}
305 \DeclareFontSeriesChangeRule {l}{m?}{m}{}
306 \DeclareFontSeriesChangeRule {m}{m?}{m}{}
307 \DeclareFontSeriesChangeRule {sbc}{m?}{c}{}
308 \DeclareFontSeriesChangeRule {sbec}{m?}{ec}{}
309 \DeclareFontSeriesChangeRule {bsc}{m?}{sc}{}
310 \DeclareFontSeriesChangeRule {sbx}{m?}{x}{}
311 \DeclareFontSeriesChangeRule {sb}{m?}{m}{}
312 \DeclareFontSeriesChangeRule {sc}{m?}{sc}{}
313 \DeclareFontSeriesChangeRule {slc}{m?}{c}{}
314 \DeclareFontSeriesChangeRule {slec}{m?}{ec}{}
315 \DeclareFontSeriesChangeRule {slsc}{m?}{sc}{}
316 \DeclareFontSeriesChangeRule {slx}{m?}{x}{}
317 \DeclareFontSeriesChangeRule {sl}{m?}{m}{}
318 \DeclareFontSeriesChangeRule {ubc}{m?}{c}{}
319 \DeclareFontSeriesChangeRule {ubec}{m?}{ec}{}
320 \DeclareFontSeriesChangeRule {ubsc}{m?}{sc}{}
321 \DeclareFontSeriesChangeRule {ubx}{m?}{x}{}
322 \DeclareFontSeriesChangeRule {ub}{m?}{ub}{}
323 \DeclareFontSeriesChangeRule {ulc}{m?}{c}{}
324 \DeclareFontSeriesChangeRule {ulec}{m?}{ec}{}
325 \DeclareFontSeriesChangeRule {ulsc}{m?}{sc}{}
326 \DeclareFontSeriesChangeRule {ulx}{m?}{x}{}

```

```

327 \DeclareFontSeriesChangeRule {ul}{m?}{m}{-}
328 \DeclareFontSeriesChangeRule {x}{m?}{x}{-}
    And there the special rules for ?m:
329 \DeclareFontSeriesChangeRule {bc}{?m}{b}{-}
330 \DeclareFontSeriesChangeRule {bec}{?m}{b}{-}
331 \DeclareFontSeriesChangeRule {bsc}{?m}{b}{-}
332 \DeclareFontSeriesChangeRule {bsc}{?m}{b}{-}
333 \DeclareFontSeriesChangeRule {bx}{?m}{b}{-}
334 \DeclareFontSeriesChangeRule {b}{?m}{b}{-}
335 \DeclareFontSeriesChangeRule {c}{?m}{m}{-}
336 \DeclareFontSeriesChangeRule {ebc}{?m}{eb}{-}
337 \DeclareFontSeriesChangeRule {ebec}{?m}{eb}{-}
338 \DeclareFontSeriesChangeRule {ebsc}{?m}{eb}{-}
339 \DeclareFontSeriesChangeRule {ebsc}{?m}{eb}{-}
340 \DeclareFontSeriesChangeRule {ebx}{?m}{eb}{-}
341 \DeclareFontSeriesChangeRule {eb}{?m}{eb}{-}
342 \DeclareFontSeriesChangeRule {ec}{?m}{m}{-}
343 \DeclareFontSeriesChangeRule {elc}{?m}{el}{-}
344 \DeclareFontSeriesChangeRule {elec}{?m}{el}{-}
345 \DeclareFontSeriesChangeRule {elsc}{?m}{el}{-}
346 \DeclareFontSeriesChangeRule {elsc}{?m}{el}{-}
347 \DeclareFontSeriesChangeRule {elx}{?m}{el}{-}
348 \DeclareFontSeriesChangeRule {el}{?m}{el}{-}
349 \DeclareFontSeriesChangeRule {lc}{?m}{l}{-}
350 \DeclareFontSeriesChangeRule {lec}{?m}{l}{-}
351 \DeclareFontSeriesChangeRule {lsc}{?m}{l}{-}
352 \DeclareFontSeriesChangeRule {lsc}{?m}{l}{-}
353 \DeclareFontSeriesChangeRule {lx}{?m}{l}{-}
354 \DeclareFontSeriesChangeRule {l}{?m}{l}{-}
355 \DeclareFontSeriesChangeRule {m}{?m}{m}{-}
356 \DeclareFontSeriesChangeRule {sbc}{?m}{sb}{-}
357 \DeclareFontSeriesChangeRule {sbec}{?m}{sb}{-}
358 \DeclareFontSeriesChangeRule {bsbc}{?m}{sb}{-}
359 \DeclareFontSeriesChangeRule {bsbc}{?m}{sb}{-}
360 \DeclareFontSeriesChangeRule {sbx}{?m}{sb}{-}
361 \DeclareFontSeriesChangeRule {sb}{?m}{sb}{-}
362 \DeclareFontSeriesChangeRule {sc}{?m}{m}{-}
363 \DeclareFontSeriesChangeRule {sc}{?m}{m}{-}
364 \DeclareFontSeriesChangeRule {slc}{?m}{sl}{-}
365 \DeclareFontSeriesChangeRule {slec}{?m}{sl}{-}
366 \DeclareFontSeriesChangeRule {slsc}{?m}{sl}{-}
367 \DeclareFontSeriesChangeRule {slsc}{?m}{sl}{-}
368 \DeclareFontSeriesChangeRule {slx}{?m}{sl}{-}
369 \DeclareFontSeriesChangeRule {sl}{?m}{sl}{-}
370 \DeclareFontSeriesChangeRule {ubc}{?m}{ub}{-}
371 \DeclareFontSeriesChangeRule {ubec}{?m}{ub}{-}
372 \DeclareFontSeriesChangeRule {ubsc}{?m}{ub}{-}
373 \DeclareFontSeriesChangeRule {ubsc}{?m}{ub}{-}
374 \DeclareFontSeriesChangeRule {ubx}{?m}{ub}{-}
375 \DeclareFontSeriesChangeRule {ub}{?m}{m}{-}
376 \DeclareFontSeriesChangeRule {ulc}{?m}{ul}{-}
377 \DeclareFontSeriesChangeRule {ulec}{?m}{ul}{-}
378 \DeclareFontSeriesChangeRule {ulsc}{?m}{ul}{-}
379 \DeclareFontSeriesChangeRule {ulsc}{?m}{ul}{-}

```

```

380 \DeclareFontSeriesChangeRule {ulx}{?m}{ul}{\}
381 \DeclareFontSeriesChangeRule {ul}{?m}{ul}{\}
382 \DeclareFontSeriesChangeRule {x}{?m}{m}{\}
    Supporting rollback ...
383 </2ekernel | latexrelease>
384 <latexrelease>\EndIncludeInRelease
385 <latexrelease>\IncludeInRelease{0000/00/00}%
386 <latexrelease> {\DeclareFontSeriesChangeRule}{Series change rules}%
387 <latexrelease>
388 <latexrelease>\let\DeclareFontSeriesChangeRule\@undefined
389 <latexrelease>
390 <latexrelease>\EndIncludeInRelease

```

1.3 Changing to a new series

```

391 <*2ekernel | latexrelease>
392 <latexrelease>\IncludeInRelease{2021/06/01}%
393 <latexrelease> {\fontseries}{delay fontseries update}%

```

\fontseries The **\fontseries** command takes one argument which is the requested new font series. In the original implementation it simply saved the expanded value in **\f@series**. Now we do a bit more processing and look up the final value in the font series data base. This is done by **\merge@font@series**. But the lookup should be done within the target family and call to **\fontseries** might be followed by a **\fontfamily** call. So we delay the processing to **\selectfont** and only record the necessary action in **\delayed@f@adjustment**.

```

394 \DeclareRobustCommand\fontseries[1]{\@forced@seriesfalse
395   \expandafter\def\expandafter\delayed@f@adjustment\expandafter
396     {\delayed@f@adjustment\delayed@merge@font@series{#1}}}

```

(End definition for \fontseries.)

\delayed@f@adjustment The macro holding the delayed action(s) for use in **\selectfont**.

```

397 \let\delayed@f@adjustment\@empty

```

(End definition for \delayed@f@adjustment.)

\fontseriesforce To change unconditionally to a new series you can use **\fontseriesforce**. Of course, if the series doesn't exist for the current family substitution still happens, but there is not dependency on the current series.

```

398 \DeclareRobustCommand\fontseriesforce[1]{\@forced@seriestrue
399   \expandafter\def\expandafter\delayed@f@adjustment\expandafter
400     {\delayed@f@adjustment\edef\f@series{#1}}}

```

(End definition for \fontseriesforce.)

\if@forced@series If the series gets forced we need to know that fact later on.

```

401 \newif\if@forced@series

```

(End definition for \if@forced@series.)

```

402 </2ekernel | latexrelease>
403 <latexrelease>\EndIncludeInRelease

```

```

404 <latexrelease>\IncludeInRelease{2020/02/02}%
405 <latexrelease>          {\fontseries}{delay fontseries update}%
406 <latexrelease>
407 <latexrelease>\DeclareRobustCommand\fontseries[1]{\@forced@seriesfalse
408 <latexrelease>          \merge@font@series{#1}}
409 <latexrelease>\DeclareRobustCommand\fontseriesforce[1]{\@forced@seriestrue
410 <latexrelease>          \edef\f@series{#1}}
411 <latexrelease>\let\delayed@f@adjustment\@undefined
412 <latexrelease>

```

For a roll forward we may have to define `\if@forced@series` but this needs doing in a way that $\mathrm{T}_{\mathrm{E}}\mathrm{X}$ doesn't see it when skipping over conditionals.

```

413 <latexrelease>\expandafter\newif\csname if@forced@series\endcsname
414 <latexrelease>
415 <latexrelease>\EndIncludeInRelease

416 <latexrelease>\IncludeInRelease{0000/00/00}%
417 <latexrelease>          {\fontseries}{delay fontseries update}%
418 <latexrelease>
419 <latexrelease>\DeclareRobustCommand\fontseries[1]{\edef\f@series{#1}}
420 <latexrelease>\let\fontseriesforce\@undefined
421 <latexrelease>
422 <latexrelease>\EndIncludeInRelease

423 <*2ekernel | latexrelease>
424 <latexrelease>\IncludeInRelease{2020/02/02}%
425 <latexrelease>    {\merge@font@series}{Merge series values}%

```

`\merge@font@series` We look up the data base value by expanding the right command twice. If no such value exist then the result will be `\relax` otherwise it will be the two brace groups: the desired result and the alternate result. The first case means that the third argument to `\merge@font@series` will be empty.

```

426 \def\merge@font@series#1{%
427   \expandafter\expandafter\expandafter
428   \merge@font@series@
429   \csname series@\f@series @#1\endcsname
430   {#1}%
431   \@nil
432 }

```

(End definition for `\merge@font@series`.)

`\merge@font@series@` This now defines the new `\f@series`:

```

433 \def\merge@font@series@#1#2#3\@nil{%

```

If the third argument is empty there is no database entry for the combination and the second argument holds the new series so we return that.

Originally the test was simply `\ifx!#3!` but that actually dies if `#3` starts with a conditional and in the definition of `\AmSfont` that is actually the case.

```

434 %\ifcat\expandafter X\detokenize{#1}X%
435 \def\reserved@a{#3}%
436 \ifx\reserved@a\@empty
437   \set@target@series{#2}%
438 \else

```

Otherwise we check if the desired result for the series (#1) exists for the font family and the current shape. All this happens inside `\selectfont` which has already taken care to load the `.fd` file if necessary.

```
439 \edef\reserved@a{\f@encoding /\f@family /#1/\f@shape}%
440 \ifcsname \reserved@a \endcsname
```

If the desired result is available then we use that. However, we do need some post-processing because we need to drop surplus ms due to the way naming convention was designed in the '90s (sigh).

```
441 \set@target@series{#1}%
```

If not, then we try the alternate result (#2).

```
442 \else
443 \ifcsname \f@encoding /\f@family /#2/\f@shape \endcsname
```

If the alternate result exist we use that and also issue a warning (or rather a log entry) that we didn't managed to change to the desired font.

```
444 \set@target@series{#2}%
445 \@font@shape@subst@warning
```

If that doesn't exist either, then we use the requested series unmodified (again with a warning).

```
446 \else
447 \set@target@series{#3}%
448 \@font@shape@subst@warning
449 \fi
450 \fi
451 \fi
452 }
```

It is possible that the previous font and the new one are actually identical (and the font was not found because it still needs loading) in which case a warning would look rather odd. So we make a quick check for that (which is the reason why we defined `\@reserved@a` above instead of doing inline testing inside `\ifcsname`).

```
453 \def\@font@shape@subst@warning{%
454 \edef\reserved@b{\curr@fontshape}%
455 \ifx\reserved@a\reserved@b \else
456 \@font@warning{Font shape '\reserved@a' undefined\MessageBreak
457 using '\reserved@b' instead}%
458 \fi
459 }
```

(End definition for `\merge@font@series@`.)

`\merge@font@series@without@substitution`
`\merge@font@series@without@substitution@`
`\delayed@merge@font@series`

`\merge@font@series@without@substitution` works like `\merge@font@series`, i.e., it looks up the combination in the rule base and if there exists an entry it uses it and if not it uses the new series value. However, it doesn't check if there is actually a font face with the new series value as `\merge@font@series` does. This simplified command is used in `\selectfont` at a point where other font attributes are not yet updated so that checking the font face might result incorrect in substitutions.

```
460 \def\merge@font@series@without@substitution#1{%
461 \expandafter\expandafter\expandafter
462 \merge@font@series@without@substitution@
463 \csname series@\f@series @#1\endcsname
```

```

464     {#1}%
465     \@nil
466 }
467 \def\merge@font@series@without@substitution@#1#2#3\@nil{%
468   \def\reserved@a{#3}%
469   \ifx\reserved@a\@empty
470     \set@target@series{#2}%
471   \else
472     \set@target@series{#1}%
473   \fi
474 }

```

(End definition for \merge@font@series@without@substitution,
\merge@font@series@without@substitution@, and \delayed@merge@font@series.)

\delayed@merge@font@series When we delay the merge action in `\fontseries` we first attempt to use merging without substitution. If that results in a non-existing font face the merge is redone in `\selectfont` using a version with substitution. See `\selectfont` for details.

```

475 \let\delayed@merge@font@series\merge@font@series@without@substitution

```

(End definition for \delayed@merge@font@series.)

\maybe@load@fontshape A small helper that we use a couple of times: try loading a fontshape (in a group because `\try@load@fontshape` normalizes catcodes and we also want to change `\typeout` so that it doesn't report missing .fd files on the terminal).

```

476 \def\maybe@load@fontshape{%
477   \begingroup
478   \let \typeout \@font@info
479   \try@load@fontshape
480   \endgroup}

```

(End definition for \maybe@load@fontshape.)

\set@target@series Finally the code for normalizing the `\f@series` value.
The combined series value determined by the mapping may still contain an `m` that we have to remove (as the .fd files use `c` not `mc` to denote a medium weight condensed series, etc.). We do this in all branches above because a user might have written

```
\DeclareFontSeriesChangeRule {m}{sc}{msc}{mc}
```

instead of using `sc` and `c` as needed in the .fd file.

```

481 \def\set@target@series#1{%

```

We need to `\edef` the argument first in case it starts with a conditional. Then we check (and perhaps drop) an “m” from the value and assign the result to `\f@series`.

```

482   \edef\f@series{#1}%
483   \series@maybe@drop@one@m\f@series\f@series
484 }

```

(End definition for \set@target@series.)

`\series@maybe@drop@one@m` If the series value is in NFSS notation then it should not contain any “m” unless it is just an “m” by its own. So we need to drop surplus “m”s. But we better don’t do this for full names, such as “**semibold**” as used by `autoinst`, for example. So we test against the possible explicit values that should drop an “m”. After that we assign the result to #2 for further use.

```

485 \def\series@maybe@drop@one@m#1{%
486   \expandafter\series@maybe@drop@one@m@x\expandafter{#1}}
487
488 \def\series@maybe@drop@one@m@x#1#2{%

```

The code below is an inline version of the `\in@` macro without the group, so that it works in `\accent`.

```

489   \def\in@@#1,#1,{}%
490   \series@check@toks\expandafter{\in@@
491     ,ulm,elm,lm,slm,mm,sbm,bm,ebm,ubm,muc,mec,mc,msc,msx,mx,mex,mux,{},{},#1,}%
492   \edef\in@@{\the\series@check@toks}%
493   \ifx\in@@\@empty

```

The default definition for `\bfdefault` etc is actually `b\@empty` so that we can detect if the user has changed the default. However that means a) the above test will definitely fail (maybe something to change) and b) we better use `\edef` on the next line to get rid of it as otherwise the test against #2 (e.g. `\bfdef@ult`) will fail in other places.

```

494     \edef#2{#1}%
495   \else
496     \edef#2{\expandafter\series@drop@one@m #1m\series@drop@one@m}%
497   \fi
498 }

```

As a precaution we use a private toks register not `\toks@` as that is no longer hidden inside the group.

```

499 \newtoks\series@check@toks

```

(End definition for \series@maybe@drop@one@m.)

`\series@drop@one@m` Drop up to two ms but keep one if that makes the series value empty. Actually, with the current implementation we know that there is at least one in the series value itself and we added one after it, so all we have to do is now returning #1#2 and dropping the rest.

```

500 \def\series@drop@one@m#1m#2m#3\series@drop@one@m{%
501 %   \ifx\relax#1#2\relax m\else#1#2\fi
502   #1#2%
503 }

```

(End definition for \series@drop@one@m.)

Supporting rollback ...

```

504 </2ekernel | latexrelease>
505 <latexrelease>\EndIncludeInRelease
506 <latexrelease>\IncludeInRelease{0000/00/00}%
507 <latexrelease>   {\merge@font@series}{Merge series values}%
508 <latexrelease>
509 <latexrelease>\let\merge@font@series\@undefined
510 <latexrelease>\let\merge@font@series@\@undefined
511 <latexrelease>\let\@font@shape@subst@warning\@undefined
512 <latexrelease>\let\merge@font@series@without@substitution\@undefined
513 <latexrelease>\let\merge@font@series@without@substitution@\@undefined

```

```

514 <latexrelease>\let\delayed@merge@font@series\@undefined
515 <latexrelease>\let\maybe@load@fontshape\@undefined
516 <latexrelease>\let\set@target@series\@undefined
517 <latexrelease>\let\series@maybe@drop@one@m\@undefined
518 <latexrelease>\let\series@drop@one@m\@undefined
519 <latexrelease>
520 <latexrelease>\EndIncludeInRelease

```

2 Changing the shape

Shapes are also split in two axes (though it could be more if that is desirable), essentially building in an “sc” axis).

```

521 <*2ekernel | latexrelease>
522 <latexrelease>\IncludeInRelease{2020/02/02}%
523 <latexrelease> \{\DeclareFontShapeChangeRule\}{Font shape change rules}%

```

`\DeclareFontShapeChangeRule` The database for shapes is done in exactly the same way, only that it is much smaller and we usually have no alternative shape (or rather it is empty thus not used).

```

524 \def\DeclareFontShapeChangeRule #1#2#3#4{%
525   \namedef{shape@#1@#2}{#{#3}{#4}}

```

(End definition for `\DeclareFontShapeChangeRule`.)

There is kind of the same problem with returning back from `sc` to normal. It sort of needs its own letter. In `fontspec` this was solved by the first time `\upshape` changes `it` or `sl` back (so only `sc` remains) and second time it changes then `sc` back to normal. Maybe that’s not a bad way to handle it, but decided for a slightly different approach: `n` always returns to “normal”, ie resets everything and `up` changes italic or slanted to upright and `ulc` undoes small caps.

So we now offer `\normalshape` (using `\shapedefault` which is normally the same as calling both `\ulcshape` and `\upshape`, only more efficient.

`\ulcshape` To request going back to upper/lowercase we need a new command. It uses `ulc` as shape name but this shape is virtual, i.e., it doesn’t exist as a real shape, it is only used as part of the database table entries and thus only appears in the second argument there (but not in the first).

```

526 \DeclareRobustCommand\ulcshape
527   {\not@math@alphabet\ulcshape\relax
528   \fontshape\ulcdefault\selectfont}
529 \let\ulcdefault\@undefined % for rollback
530 \newcommand\ulcdefault{ulc}

```

(End definition for `\ulcshape`, `\textulc`, and `\ulcdefault`.)

`\swshape` New command to select a swash shape. The standard rules put this in the same category as italics or slanted, i.e., if you ask for it then italics are undone. One could provide more complicated rules so that `it + sw` becomes `swit` but given that there are only very few fonts that have swash letters that level of flexibility (these days) would be just resulting in a lot of combinations that do not exist.

```

531 \DeclareRobustCommand\swshape
532   {\not@math@alphabet\swshape\relax
533   \fontshape\swdefault\selectfont}
534 \let\swdefault\@undefined % for rollback
535 \newcommand\swdefault{sw}

```


(End definition for `\swshape`, `\textsw`, and `\swdefault`.)

`\sscshape` New command to select spaced small capitals. This is only here because `fontaxes` offered it. There isn't a single free font that supports it. However, some commercial ones do, so we offer it so that at some point `fontaxes` could be retired.

So far there aren't any rules for it—probably there should be some putting it in the same category as `sc`.

```
536 \DeclareRobustCommand\sscshape
537     {\not@math@alphabet\sscshape\relax
538     \fontshape\sscdefault\selectfont}
539 \let\sscdefault\undefined      % for rollback
540 \newcommand\sscdefault{ssc}
```

(End definition for `\sscshape`, `\textssc`, and `\sscdefault`.)

2.1 Mapping rules for shape combinations

Many of the entries are commented out as we will get that result without any entry.

```
541 %\DeclareFontShapeChangeRule {n}{n} {n} {}
542 \DeclareFontShapeChangeRule {n}{it} {it} {sl}
543 \DeclareFontShapeChangeRule {n}{sl} {sl} {it}
544 %\DeclareFontShapeChangeRule {n}{sw} {sw} {}
545 %\DeclareFontShapeChangeRule {n}{sc} {sc} {}
546 \DeclareFontShapeChangeRule {n}{ulc} {n} {}
547 \DeclareFontShapeChangeRule {n}{up} {n} {}

548 %\DeclareFontShapeChangeRule {it}{n} {n} {}
549 %\DeclareFontShapeChangeRule {it}{it} {it} {}
550 \DeclareFontShapeChangeRule {it}{sl} {sl} {it}
551 %\DeclareFontShapeChangeRule {it}{sw} {sw} {}
```

If neither `scit` nor `scsl` exist then `sc` will be used as a fallback albeit with a log entry, so except for the latter there will be no change for CM or Latin Modern fonts.

```
552 \DeclareFontShapeChangeRule {it}{sc} {scit} {scsl}
553 \DeclareFontShapeChangeRule {it}{ulc} {it} {}
554 \DeclareFontShapeChangeRule {it}{up} {n} {}

555 %\DeclareFontShapeChangeRule {sl}{n} {n} {}
556 \DeclareFontShapeChangeRule {sl}{it} {it} {sl}
557 %\DeclareFontShapeChangeRule {sl}{sl} {sl} {}
558 %\DeclareFontShapeChangeRule {sl}{sw} {sw} {}
559 \DeclareFontShapeChangeRule {sl}{sc} {scsl} {scit}
560 \DeclareFontShapeChangeRule {sl}{ulc} {sl} {}
561 \DeclareFontShapeChangeRule {sl}{up} {n} {}

562 %\DeclareFontShapeChangeRule {sc}{n} {n} {}
563 \DeclareFontShapeChangeRule {sc}{it} {scit} {scsl}
564 \DeclareFontShapeChangeRule {sc}{sl} {scsl} {scit}
565 \DeclareFontShapeChangeRule {sc}{sw} {scsw} {sw}
566 %\DeclareFontShapeChangeRule {sc}{sc} {sc} {}
567 \DeclareFontShapeChangeRule {sc}{ulc} {n} {}
```

The next rule might be a bit surprising and rightly so. Correct would be that `sc` is not affected by `up`, i.e., remains `sc` as showed in the commented out rule. However, for nearly three decades commands such as `sc` or `\textup` changed small caps back to the “normal” shape. So for backward compatibility we keep that behavior.

As a result you are currently typesetting in `scit` or `scsl` using `\upshape` twice will return you to the normal shape too, the first will change to `sc` and the second (because of the rule below) change that to `n`. This is the way `fontspec` implemented its version on this interface, so this rule means we are also compatible with the way `fontspec` behaved. Still it remains an oddity which I would rather liked to have avoided.

```

568 %\DeclareFontShapeChangeRule {sc}{up} {sc} {}
569 \DeclareFontShapeChangeRule {sc}{up} {n} {}

570 %\DeclareFontShapeChangeRule {scit}{n} {n} {}
571 \DeclareFontShapeChangeRule {scit}{it} {scit} {}
572 \DeclareFontShapeChangeRule {scit}{sl} {scsl} {scit}
573 \DeclareFontShapeChangeRule {scit}{sw} {scsw} {sc} % or scit?
574 \DeclareFontShapeChangeRule {scit}{sc} {scit} {}
575 \DeclareFontShapeChangeRule {scit}{ulc} {it} {}
576 \DeclareFontShapeChangeRule {scit}{up} {sc} {}

```

The previous rule assumes that if `scit` exists then it exists as well. If not, the mechanism will save `ulc` in `\f@series` which most certainly doesn't exist. So when a font is later selected that would result in a substitution (so no harm done really). Alternatively, we could in this case use `n` as alternative, which may be a bit faster, but such a setup would be so weird in the first place that this isn't worth the effort.

```

577 %\DeclareFontShapeChangeRule {scsl}{n} {n} {}
578 \DeclareFontShapeChangeRule {scsl}{it} {scit} {scsl}
579 \DeclareFontShapeChangeRule {scsl}{sl} {scsl} {}
580 \DeclareFontShapeChangeRule {scsl}{sw} {scsw} {sc} % or scsl?
581 \DeclareFontShapeChangeRule {scsl}{sc} {scsl} {}
582 \DeclareFontShapeChangeRule {scsl}{ulc} {sl} {}
583 \DeclareFontShapeChangeRule {scsl}{up} {sc} {}

584 %\DeclareFontShapeChangeRule {scsw}{n} {n} {}
585 \DeclareFontShapeChangeRule {scsw}{it} {scit} {scsw}
586 \DeclareFontShapeChangeRule {scsw}{sl} {scsl} {}
587 \DeclareFontShapeChangeRule {scsw}{sw} {scsw} {}
588 \DeclareFontShapeChangeRule {scsw}{sc} {scsw} {}
589 \DeclareFontShapeChangeRule {scsw}{ulc} {sw} {}
590 \DeclareFontShapeChangeRule {scsw}{up} {sc} {}

591 %\DeclareFontShapeChangeRule {sw}{n} {n} {}
592 %\DeclareFontShapeChangeRule {sw}{it} {it} {}
593 %\DeclareFontShapeChangeRule {sw}{sl} {sl} {}
594 %\DeclareFontShapeChangeRule {sw}{sw} {sw} {}
595 \DeclareFontShapeChangeRule {sw}{sc} {scsw} {}
596 \DeclareFontShapeChangeRule {sw}{ulc} {sw} {}
597 \DeclareFontShapeChangeRule {sw}{up} {n} {}

```

Supporting rollback ...

```

598 </2ekernel | latexrelease>
599 <latexrelease>\EndIncludeInRelease
600 <latexrelease>\IncludeInRelease{0000/00/00}%
601 <latexrelease> {\DeclareFontShapeChangeRule}{Font shape change rules}%
602 <latexrelease>
603 <latexrelease>\let\DeclareFontShapeChangeRule\@undefined
604 <latexrelease>\let\ulcshape\@undefined
605 <latexrelease>\let\ulcdefault\@undefined
606 <latexrelease>\let\swshape\@undefined

```

```

607 <latexrelease>\let\swdefault\@undefined
608 <latexrelease>\let\sscshape\@undefined
609 <latexrelease>\let\sscdefault\@undefined
610 <latexrelease>
611 <latexrelease>\EndIncludeInRelease

```

2.2 Changing to a new shape

```

612 <*2ekernel | latexrelease>
613 <latexrelease>\IncludeInRelease{2021/06/01}%
614 <latexrelease> {\fontshape}{Font shape change}%

```

`\fontshape` Again the `\fontshape` now has to do a lookup to get to its new value in `\f@shape`. The method is exactly the same as in `\fontseries`.

```

615 \DeclareRobustCommand\fontshape[1]
616   {\expandafter\def\expandafter\delayed@f@adjustment\expandafter
617     {\delayed@f@adjustment\delayed@merge@font@shape{#1}}}

```

(End definition for \fontshape.)

`\fontshapeforce` The unconditional version:

```

618 \DeclareRobustCommand\fontshapeforce[1]
619   {\expandafter\def\expandafter\delayed@f@adjustment\expandafter
620     {\delayed@f@adjustment\edef\f@shape{#1}}}

```

(End definition for \fontshapeforce.)

Supporting rollback ...

```

621 </2ekernel | latexrelease>
622 <latexrelease>\EndIncludeInRelease
623 <latexrelease>\IncludeInRelease{2020/02/02}%
624 <latexrelease> {\fontshape}{Font shape change}%
625 <latexrelease>
626 <latexrelease>\DeclareRobustCommand\fontshape[1]{\merge@font@shape{#1}}
627 <latexrelease>\DeclareRobustCommand\fontshapeforce[1]{\edef\f@shape{#1}}
628 <latexrelease>
629 <latexrelease>\EndIncludeInRelease

630 <latexrelease>\IncludeInRelease{0000/00/00}%
631 <latexrelease> {\fontshape}{Font shape change}%
632 <latexrelease>
633 <latexrelease>\DeclareRobustCommand\fontshape [1]{\edef\f@shape{#1}}
634 <latexrelease>\let\fontshapeforce\@undefined
635 <latexrelease>
636 <latexrelease>\EndIncludeInRelease

637 <*2ekernel | latexrelease>
638 <latexrelease>\IncludeInRelease{2020/02/02}%
639 <latexrelease> {\merge@font@shape}{Font shape change rules}%

```

`\merge@font@shape` Look up the database entry (if existing) and act accordingly.

```

640 \def\merge@font@shape#1{%
641   \expandafter\expandafter\expandafter
642   \merge@font@shape@
643   \csname shape@\f@shape @#1\endcsname
644   {#1}%

```

```

645 \@nil
646 }

```

(End definition for \merge@font@shape.)

\merge@font@shape@ Same game now, except that we look at shapes not series values and we can set the shape without the complication of dropping “m”s from the name as we had to for the series.

```

647 \def\merge@font@shape@#1#2#3\@nil{%
648 \def\reserved@a{#3}%
649 \ifx\reserved@a\@empty
650 \edef\f@shape{#2}%
651 \else

```

\reserved@a is used in \@font@shape@subst@warning so we have to define it in addition to do the \ifcsname test

```

652 \edef\reserved@a{\f@encoding /\f@family /\f@series/#1}%
653 \ifcsname \reserved@a\endcsname
654 \edef\f@shape{#1}%
655 \else
656 \ifcsname \f@encoding /\f@family /\f@series/#2\endcsname
657 \edef\f@shape{#2}%
658 \@font@shape@subst@warning
659 \else
660 \edef\f@shape{#3}%
661 \@font@shape@subst@warning
662 \fi
663 \fi
664 \fi
665 }

```

(End definition for \merge@font@shape@.)

\merge@font@shape@without@substitution See definition of \selectfont for how these macros are used.

\merge@font@shape@without@substitution@
\delayed@merge@font@shape

```

666 \def\merge@font@shape@without@substitution#1{%
667 \expandafter\expandafter\expandafter
668 \merge@font@shape@without@substitution@
669 \csname shape@\f@shape @#1\endcsname
670 {#1}%
671 \@nil
672 }

673 \def\merge@font@shape@without@substitution@#1#2#3\@nil{%
674 \def\reserved@a{#3}%
675 \ifx\reserved@a\@empty
676 \edef\f@shape{#2}%
677 \else
678 \edef\f@shape{#1}%
679 \fi
680 }

681 \let\delayed@merge@font@shape\merge@font@shape@without@substitution

```

(End definition for \merge@font@shape@without@substitution,
\merge@font@shape@without@substitution@, and \delayed@merge@font@shape.)

`\normalshape` `\normalshape` resets both sub-axes if the default rules are used.

```
682 \protected\def\normalshape
683   {\not@math@alphabet\normalshape\relax
684    \fontshape\shapedefault\selectfont}%
```

(End definition for \normalshape.)

3 Make sure we win ...

This code implements one aspect of what the package `fontaxes` provide. So its redefinitions for the various shape commands, such as `\itshape` should no longer happen. We therefore force the standard definitions at `\AtBeginDocument` (later when this is defined. Once `fontaxes` is no longer doing such redefinitions that could be taken out again.

We use a separate macro so that we can easily disable this (in case of rollback).

`\reinstall@nfss@defs` I use `\protected` here not `\DeclareRobustCommand` to avoid extra status lines.

```
685 \def\reinstall@nfss@defs{%
686   \protected\def\upshape
687     {\not@math@alphabet\upshape\relax
688      \fontshape\updefault\selectfont}%
689   \protected\def\slshape
690     {\not@math@alphabet\slshape\relax
691      \fontshape\sldefault\selectfont}%
692   \protected\def\scshape
693     {\not@math@alphabet\scshape\relax
694      \fontshape\scdefault\selectfont}%
695   \protected\def\itshape
696     {\not@math@alphabet\itshape\mathit
697      \fontshape\itdefault\selectfont}%
698   \protected\def\ulcshape
699     {\not@math@alphabet\ulcshape\relax
700      \fontshape\ulc\selectfont}%
701   \protected\def\swshape
702     {\not@math@alphabet\swshape\relax
703      \fontshape\swdefault\selectfont}%
704   \protected\def\sscshape
705     {\not@math@alphabet\sscshape\relax
706      \fontshape\sscdefault\selectfont}%
707 }
```

(End definition for \reinstall@nfss@defs.)

Supporting rollback ...

```
708 </2ekernel | latexrelease>
709 <latexrelease>\EndIncludeInRelease
710 <latexrelease>\IncludeInRelease{0000/00/00}%
711 <latexrelease>  {\merge@font@shape}{Font shape change rules}%
712 <latexrelease>
713 <latexrelease>\DeclareRobustCommand\fontshape [1]{\edef\f@shape{#1}}
714 <latexrelease>\let\fontshapeforce\@undefined
715 <latexrelease>
716 <latexrelease>\let\merge@font@shape\@undefined
717 <latexrelease>\let\merge@font@shape@\@undefined
718 <latexrelease>
```

```

719 <latexrelease>\let\merge@font@shape@without@substitution\@undefined
720 <latexrelease>\let\merge@font@shape@without@substitution@\@undefined
721 <latexrelease>\let\delayed@merge@font@shape\@undefined
722 <latexrelease>
723 <latexrelease>\let\normalshape\@undefined
724 <latexrelease>

```

This is always called in `\document` so don't make it undefined.

```

725 <latexrelease>
726 <latexrelease>\let\reinstall@nfss@defs\relax
727 <latexrelease>\EndIncludeInRelease

```

This initializes the 2020/02/02 extensions to NFSS after any changes in the preamble.

```

728 <*2ekernel | latexrelease>
729 <latexrelease>\IncludeInRelease{2020/10/01}%
730 <latexrelease>          {\reinstall@nfss@defs}{NFSS series init}%
731 \g@addto@macro\@kernel@after@begindocument@before
732          {\reinstall@nfss@defs\init@series@setup}
733 </2ekernel | latexrelease>
734 <latexrelease>\EndIncludeInRelease

```

The initialization was introduced in 2020/02/02 but

```

735 <latexrelease>\IncludeInRelease{2020/02/02}%
736 <latexrelease>          {\reinstall@nfss@defs}{NFSS series init}%
737 <latexrelease>\AtBeginDocument{\reinstall@nfss@defs\init@series@setup}
738 <latexrelease>\EndIncludeInRelease

739 <latexrelease>\IncludeInRelease{0000/00/00}%
740 <latexrelease>          {\reinstall@nfss@defs}{NFSS series init}%
741 <latexrelease>\EndIncludeInRelease
742 <*2ekernel>
743 </2ekernel>

```

File w

ltfsstrc.dtx

1 Introduction

This package contains the code for tracing font loading and font changes. It basically overlays some of the low-level functions of NFSS with additional code used for tracing.

The package accepts the following options:

errorshow Write all information about font changes etc. only to the transcript file unless an error happens. This means that information about font substitution will not be shown on the terminal.

warningshow Show all NFSS warnings on the terminal. This setting corresponds to the default behaviour of NFSS if the `tracefnt` package is *not* loaded!

infoshow Show all NFSS warning and all NFSS info messages (that are normally only written to the transcript file) also on the terminal. This is the default if the `tracefnt` package is loaded.

debugshow In addition to **infoshow** show also changing of math fonts as far as possible (this option can produce a large amount of output).

loading Show the name of external fonts when they are loaded. This option shows only “newly” loaded fonts not those already preloaded in the format or the class file before the `tracefnt` package became active.

pausing Turn all font warnings into errors so that L^AT_EX will stop.

2 A driver for this document

The next bit of code contains the documentation driver file for T_EX, i.e., the file that will produce the documentation you are currently reading. It will be extracted from this file by the DOCSTRIP program.

When this file is processed directly by L^AT_EX this will produce the documentation as well.

```
1 <*driver>
2 \documentclass{ltxdoc}
3
4
5 %\OnlyDescription % comment out for implementation details
6
7 \begin{document}
8   \DocInput{ltfsstrc.dtx}
9 \end{document}
10 </driver>
```

3 The Implementation

Warning: Read the macro documentation with a grain of salt. It is still basically the documentation from the first NFSS release and therefore in some cases probably not completely accurate.

If we are making a package file it is a good idea to test whether we are running under 2e. This code is actually placed at the very beginning of this file for easier maintenance, thus commented out here.

```
11 <*package>
12 %\NeedsTeXFormat{LaTeX2e}
13 %\ProvidesPackage{tracefnt}[??/??/?? v?.??]
14 %
15 </package>
16 % Standard LaTeX package (font tracing)
```

The `debug` module makes use of commands contained in a special package file named `trace.sty`.²⁴

```
16 <+debug> \input trace.sty
```

4 Handling Options

`\tracingfonts` Here is the definition of the integer register for the font trace. As a default in a package file we use 1 to give error messages if fonts are substituted. If this code is used for debugging or tracing reasons in the format file (i.e. in `fam.dtx`) we use 0 as the default. But if no font trace is used we build a definition that will produce a warning message.

```
17 <*2ekernel>
18 \def\tracingfonts{%
19   \@font@warning{Command \noexpand\tracingfonts
20     not provided.\MessageBreak
21     Use the ‘tracefnt’ package.\MessageBreak Command found:}%
22   \count@}
23 </2ekernel>
```

The `\count@` in the line above will remove the number after `\tracingfonts`. Note that this definition will be overwritten by the next line if one of these modules are included.

```
24 <*package, trace, debug>
25 \newcount\tracingfonts
26 \tracingfonts=0
27 </package, trace, debug>
```

(End definition for \tracingfonts.)

The option `errorshow` turns off all warnings so that only real errors are shown. `warningshow` corresponds to the NFSS default (when `tracefnt` is not loaded). `infoshow` is the default for this package here; and `debugshow`, `loading`, and `pausing` extend the amount of information even further.

```
28 <*package>
29 \DeclareOption{errorshow}{%
30   \def\@font@info#1{%
31     \GenericInfo{(Font)}\@spaces\@spaces\@spaces\space\space}%
32     {LaTeX Font Info: \space\space\space#1}}%
```

²⁴This package is not in distribution at the moment (and probably doesn't any longer work). Think of this part of the code as being historical artifacts.


```

33 \def\@font@warning#1{%
34     \GenericInfo{(Font)\@spaces\@spaces\@spaces\space\space}%
35     {LaTeX Font Warning: #1}}%
36 }
37 \DeclareOption{warningshow}{%
38     \def\@font@info#1{%
39         \GenericInfo{(Font)\@spaces\@spaces\@spaces\space\space}%
40         {LaTeX Font Info: \space\space\space#1}}%
41     \def\@font@warning#1{%
42         \GenericWarning{(Font)\@spaces\@spaces\@spaces\space\space}%
43         {LaTeX Font Warning: #1}}%
44     }
45 \DeclareOption{infoshow}{%
46     \def\@font@info#1{%
47         \GenericWarning{(Font)\@spaces\@spaces\@spaces\space\space}%
48         {LaTeX Font Info: \space\space\space#1}}%
49     \def\@font@warning#1{%
50         \GenericWarning{(Font)\@spaces\@spaces\@spaces\space\space}%
51         {LaTeX Font Warning: #1}}%
52     }
53 \DeclareOption{loading}{%
54     \tracingfonts\tw@
55 }
56 \DeclareOption{debugshow}{%
57     \ExecuteOptions{infoshow}%
58     \tracingfonts\thr@@
59 }
60 \DeclareOption{pausing}{%
61     \def\@font@warning#1{%
62         \GenericError
63         {(Font)\@spaces\@spaces\@spaces\space\space}%
64         {LaTeX Font Warning: #1}%
65         {See the LaTeX Companion for details.}%
66         {I'll stop for every LaTeX Font Warning because
67          you requested\MessageBreak the 'pausing' option
68          to the tracefont package.}}%
69     }

```

We make `infoshow` the default, which in turn defines `\font@warning` and `\font@info`.

```

70 \ExecuteOptions{infoshow}
71 \ProcessOptions
72 \</package>

```

We also need a default definition inside the kernel:

```

73 \<*2kernel>
74 \def\@font@info#1{%
75     \GenericInfo{(Font)\@spaces\@spaces\@spaces\space\space}%
76     {LaTeX Font Info: \space\space\space#1}}%
77 \def\@font@warning#1{%
78     \GenericWarning{(Font)\@spaces\@spaces\@spaces\space\space}%
79     {LaTeX Font Warning: #1}}%
80 \</2kernel>

```

5 Macros common to fam.tex and tracefmt.sty

In the first versions of `tracefmt.dtx` some macros of `fam.dtx`²⁵ were redefined to include the extra tracing information. Now these macros are all defined in this file (i.e. removed from `fam.dtx`) and different production versions can be obtained simply by specifying a different set of modules to include when generating `lftss.dtx`.

5.1 General font loading

`\extract@font` This macro organizes the font loading. It first calls `\get@external@font` which will return in `\external@font` the name of the external font file (the `.tfm`) as it was determined by the NFSS tables.

```

81 <*2ekernel|package>
82 \def\extract@font{%
83   \get@external@font

```

Then the external font is loaded and assigned to the font identifier stored inside `\font@name` (for this reason we need `\expandafter`).

```

84   \global\expandafter\font\font@name\external@font\relax

```

When tracing we typeout the internal and external font name.

```

85 <*trace>
86   \ifnum \tracingfonts >\@ne
87     \@font@info{External font '\external@font'
88               loaded as\MessageBreak \font@name}\fi
89 </trace>

```

Finally we call the corresponding “loading action” macros to finish things. First the font is locally selected to allow the use of `\font` inside the loading action macros.

```

90   \font@name \relax

```

The next two lines execute the “loading actions” for the family and then for the individual font shape.

```

91   \csname \f@encoding+\f@family\endcsname
92   \csname\curr@fontshape\endcsname
93   \relax
94   }
95 </2ekernel|package>

```

The `\relax` at the end needs to be explained. This is inserted to prevent `TEX` from scanning too far when it is executing the replacement text of the loading code macros.

(End definition for \extract@font.)

`\get@external@font` This function tries to find an external font name. It will place the name into the macro `\external@font`. If no font is found it will return the one that was defined via `\DeclareErrorFont`.

```

96 <*2ekernel>
97 \def\get@external@font{%

```

We don’t know the external font name at the beginning.

```

98   \let\external@font\@empty
99   \edef\font@info{\expandafter\expandafter\expandafter\string
100     \csname \curr@fontshape \endcsname}%
101   \try@size@range

```

²⁵This file is currently not distributed in documented form. Its code is part of `lftss.dtx`.

If this failed, we'll try to substitute another size of the same font. This is done by the `\try@size@substitution` macro. It “knows about” `\do@extract@font`, `\font@name`, `\f@size`, and so on.

```

102   \ifx\external@font\@empty
103     \try@size@substitution
104     \ifx\external@font\@empty
105       \@latex@error{Font \expandafter \string\font@name\space
106         not found}\@eha
107       \error@fontshape
108       \get@external@font
109   \fi\fi
110 }
111 \end{kernel}

```

(End definition for `\get@external@font`.)

```

112 <*2kernel | latexrelease | package>
113 <latexrelease> \IncludeInRelease{2021/06/01}%
114 <latexrelease>           {\selectfont}{Add hook to \selectfont}%

```

`\selectfont` The macro `\selectfont` is called whenever a font change must take place.

```

115 \DeclareRobustCommand\selectfont
116   {%

```

When `debug` is specified we actually want something like ‘`undebug`’. The font selection is now stable so that using `\tracingall` on some other macros will show us a lot of unwanted information about font loading. Therefore we disable tracing during font loading as long as `\tracingfonts` is less than 4.

```

117 <+debug> \pushtracing
118 <+debug> \ifnum\tracingfonts<4 \tracingoff
119 <+debug> \else \tracingon\p@selectfont \fi

```

If `\baselinestretch` was redefined by the user it will not longer match its internal counterpart `\f@linespread`. If so we call `\set@fontsize` to prepare `\size@update`.

```

120   \ifx\f@linespread\baselinestretch \else
121     \set@fontsize\baselinestretch\f@size\f@baselineskip \fi

```

The series and shape updates are only prepared by `\fontseries` and `\fontshape` but not executed until after we are ready to change the font face. This way they happen after a possibly new family is set which is important because they look at the available font faces in that family and alter the selection based on availability. Several calls to `\fontseries` or `\fontshape` are delayed in the order in which they appear, so that by switching them one can work around missing intermediate font faces and avoid substitutions.

We first attempt to do the merge without any substitution. As we might end up with a non-existing font face we may have to restart and therefore save the current values of `\f@series` and `\f@shape` before the merge.

But first we make a quick test to see if there are any delayed actions, because if not it is pointless to make all the assignments and try loading a missing fontshape.

```

122   \ifx\delayed@f@adjustment\@empty
123     \else
124       \let\f@shape@saved\f@shape
125       \let\f@series@saved\f@series

```

The we run the delayed adjustments (which is using the `\..@without@substitution` commands

```
126      \delayed@f@adjustment
```

We then check if the resulting combination is valid but for this we have to make sure the the appropriate `.fd` is loaded if that hasn't happened so far.

```
127      \maybe@load@fontshape
```

```
128      \ifcsname \f@encoding/\f@family/\f@series/\f@shape \endcsname
```

If this macro is defined then we are good and no further action is necessary.

Otherwise the combination is not valid, so we redo the merge but this time with substitutions.

```
129      \else
```

```
130          \let\f@shape\f@shape@saved
```

```
131          \let\f@series\f@series@saved
```

```
132          \let\delayed@merge@font@shape\merge@font@shape
```

```
133          \let\delayed@merge@font@series\merge@font@series
```

```
134          \delayed@f@adjustment
```

```
135          \let\delayed@merge@font@shape\merge@font@shape@without@substitution
```

```
136          \let\delayed@merge@font@series\merge@font@series@without@substitution
```

```
137      \fi
```

Now the series and shape values are updated and we clear `\delayed@f@adjustment`. This is important because on the next execution of `\selectfont` we should not mistakenly redo the delayed actions if there wasn't any series or shape change.

```
138      \let\delayed@f@adjustment\@empty
```

```
139      \fi
```

If the series was forced we should now cancel that in case the next series change is done with some low-level setting to `\f@series`.

```
140      \@forced@seriesfalse
```

Then we generate the internal name of the font by concatenating *family*, *series*, *shape*, and current *size*, with slashes as delimiters between them. This is much more readable than standard L^AT_EX's `\twfbf`, etc. We define `\font@name` globally, as always. The reason for this is explained later on.

```
141      \xdef\font@name{%
```

```
142          \csname\curr@fontshape/\f@size\endcsname}%
```

We call the macro `\pickup@font` which will load the font if necessary.

```
143      \pickup@font
```

Then we select the font.

```
144      \font@name
```

After switching fonts we run a hook, so that packages can make last minute alterations based on the new font (originally provided in `everyselectfont` but using a different interface).

```
145      \UseHook{selectfont}%
```

Finally we call `\size@update`. This macro is normally empty but will contain actions (like setting the `\baselineskip`) that have to be carried out when the font size, the base `\baselineskip` or the `\baselinestretch` have changed.

```
146      \size@update
```

A similar function is called to handle anything related to encoding updates. This one is changed from `\relax` by `\fontencoding`.

```
147 \enc@update
```

Just before ending this macro we have to pop the tracing stack if it was pushed before.

```
148 <+debug> \poptracing
149 }
```

(End definition for `\selectfont`.)

selectfont Declare the hook used in `selectfont` in the kernel, but not inside the `tracefnt` package.

```
150 <-trace>\NewHook{selectfont}
```

(End definition for `selectfont`.)

If `\tracingfonts` is greater than 2 we also show the font switch inside `\selectfont`. We do this by adding this code to the hook in the `tracefnt` package: macro might redefine `\font@name`.

```
151 <*trace>
152 \AddToHook{selectfont}
153   {\ifnum \tracingfonts>\tw@
154     \font@info{Switching to \font@name}\fi}
155 </trace>
156 </2ekernel | latexrelease | package>
157 <latexrelease>\EndIncludeInRelease
```

With `\selectfont` having different definitions in different kernels we also have to provide them in the `tracefnt` package to support rollback. In packages that works a bit differently and therefore we have to provide an empty block there.

```
158 <package>\IncludeInRelease{2021/06/01}%
159 <package>           {\selectfont}{Add hook to \selectfont}%
160 <package>\EndIncludeInRelease

161 <latexrelease | package>\IncludeInRelease{0000/00/00}%
162 <latexrelease | package>           {\selectfont}{Add hook to \selectfont}%
163 <latexrelease | package>
164 <latexrelease | package>\DeclareRobustCommand\selectfont
165 <latexrelease | package>  {%
166 <latexrelease | package>    \ifx\f@linespread\baselinestretch \else
167 <latexrelease | package>    \set@fontsize\baselinestretch\f@size\f@baselineskip \fi
168 <latexrelease | package>    \xdef\font@name{%
169 <latexrelease | package>      \curname\curr@fontshape/\f@size\endcurname}%
170 <latexrelease | package>    \pickup@font
171 <latexrelease | package>    \font@name
172 <latexrelease | package>    \size@update
173 <latexrelease | package>    \enc@update
174 <latexrelease | package>  }
175 <latexrelease | package>
176 <latexrelease | package>\EndIncludeInRelease
```

\set@fontsize The macro `\set@fontsize` does the actual work. First it assigns new values to `\f@size`, `\f@baselineskip` and `\f@linespread`.

```
177 <*2ekernel | package>
178 \def\set@fontsize#1#2#3{%
179   \@defaultunits\@tempdimb#2pt\relax\@nnil
```

```

180 \edef\f@size{\strip@pt\@tempdimb}%
181 \@defaultunits\@tempskipa#3pt\relax\@nnil
182 \edef\f@baselineskip{\the\@tempskipa}%
183 \edef\f@linespread{#1}%

```

For backward compatibility and for later testing within `\selectfont` the internal value of `\f@linespread` is passed back to `\baselinestretch`.

```

184 \let\baselinestretch\f@linespread

```

Additional processing will happen within `\selectfont`. For this reason the macro `\size@update` (which will be called in `\selectfont`) will be defined to be:

```

185 \def\size@update{%

```

First calculate the new `\baselineskip` and also store it in `normalbaselineskip`

```

186 \baselineskip\f@baselineskip\relax
187 \baselineskip\f@linespread\baselineskip
188 \normalbaselineskip\baselineskip

```

then to set up a new `\strutbox`

```

189 \setbox\strutbox\hbox{%
190 \vrule\@height.7\baselineskip
191 \@depth.3\baselineskip
192 \@width\z@}%

```

We end with a bit of tracing information.

```

193 \ifnum \tracingfonts>\tw@
194 \ifx\f@linespread\@empty
195 \let\reserved@a\@empty
196 \else
197 \def\reserved@a{\f@linespread x}%
198 \fi
199 \font@info{Changing size to \f@size/\reserved@a
200 \f@baselineskip}%
201 \aftergroup\type@restoreinfo \fi
202 \iftracingfonts\type@restoreinfo \fi
203 \iftracingfonts\type@restoreinfo \fi

```

When all this is processed `\size@update` redefines itself to `\relax` so that in later calls of `\selectfont` no extra code will be executed.

```

204 \let\size@update\relax}%
205 }

```

Instead of defining this macro internally we might speed things up by placing the code into a separate macro and use `\let`!

(End definition for \set@fontsize.)

\size@update Normally this macro does nothing; it will be redefined by `\set@fontsize` to initiate an update.

```

206 \let\size@update\relax

```

(End definition for \size@update.)

`\type@restoreinfo` This macro produces some info when a font size and/or baseline change will get restored.

```

207 <*trace>
208   \def\type@restoreinfo{%
209     \ifx\f@linespread\@empty
210       \let\reserved@a\@empty
211     \else
212       \def\reserved@a{\f@linespread x}%
213     \fi
214     \@font@info{Restoring size to
215                \f@size/\reserved@a\f@baselineskip}}
216 </trace>

```

(End definition for `\type@restoreinfo`.)

`\glb@settings` The macro `\glb@settings` globally selects all math fonts for the current size if necessary.
`\glb@currsz`

```

217 \def\glb@settings{%

```

When `\glb@settings` gains control a size change was requested and all previous font assignments need to be replaced. Therefore the old values of the fonts are no longer needed. For every *math group* the new assignments are appended to `\math@fonts`. But this happens only if the `\math@fonts` switch is set to true. However, we always set up the correct math sizes for script and scriptscript fonts since they may be needed even if we don't set up the whole math machinery.

Here we set the math size, script size and scriptscript size. If the `S@...` macro is not defined we have to first calculate the three sizes.

```

218   \expandafter\ifx\csname S@\f@size\endcsname\relax
219     \calculate@math@sizes
220   \fi

```

The effect of this is that `\calculate@math@sizes` may or may not define the `S@...` macro. In the first case the next time the same size is requested this macro is used, otherwise `\calculate@math@sizes` is called again. This also sets the `\math@fonts` switch. If it is true we must switch the math fonts.

```

221   \csname S@\f@size\endcsname
222   \ifmath@fonts
223 <*trace>
224     \ifnum \tracingfonts>\tw@
225       \@font@info{Setting up math fonts for
226                  \f@size/\f@baselineskip}\fi
227 </trace>

```

Inside a group we execute the macro for the current math *version*. This sets `\math@fonts` to a list of `\textfont...` assignments. `\getanddefine@fonts` (which may be called at this point) needs the `\escapechar` parameter to be set to `-1`.

```

228   \begingroup
229     \escapechar\m@ne
230     \csname mv@\math@version \endcsname

```

Then we set `\globaldefs` to 1 so that all following changes are done globally. The math font assignments recorded in `\math@fonts` are executed and `\glb@currsz` is set equal to `\f@size`. This signals that the fonts for math in this size are set up.

```

231     \globaldefs\@ne
232     \math@fonts

```

```

233         \let \glb@currsiz e \f@size
234     \endgroup

```

Finally we execute any code that is supposed to happen whenever the math font setup changes. This register will be executed in local mode which means that everything that is supposed to have any effect should be done globally inside. We can't execute it within `\globaldefs\@ne` as we don't know what ends up inside this register, e.g., it might contain calculations which use some local registers to calculate the final (global) value.

```

235     \the\every@math@size

```

Otherwise we announce that the math fonts are not set up for this size.

```

236 \< *trace>
237     \else
238         \ifnum \tracingfonts>\tw@
239             \@font@info{No math setup for
240                         \f@size/\f@baselineskip}\fi
241 \< /trace>
242     \fi
243 }
244 \< /2ekernel | package>

```

(End definition for \glb@settings and \glb@currsiz e.)

\baselinestretch In `\selectfont` we used `\baselinestretch` as a factor when assigning a value to `\baselineskip`. We use 1 as a default (i.e. no stretch).

```

245 \< *2ekernel>
246 \def\baselinestretch{1}

```

(End definition for \baselinestretch.)

\every@math@size We must still define the hook `\every@math@size` we used in `\glb@settings`. We initialize it to nothing. It is important to remember that everything that goes into this hook should to global updates, local changes will have weird effects.

```

247 \newtoks\every@math@size
248 \every@math@size={}
249 \< /2ekernel>

```

(End definition for \every@math@size.)

5.2 Math fonts setup

5.2.1 Outline of algorithm for math font sizes

\TeX uses the math fonts that are current when the end of a formula is reached. If we don't want to keep font setups local to every formula (which would result in an enormous overhead, we have to be careful not to end up with the wrong setup in case formulas are nested, e.g., we need to be able to handle

$\$ a=b+c \mathrel{\mathop{\boxtimes}} \mathrel{\mathop{\small}} \text{for all } b \text{ and } c \text{ in } \mathbb{Z} \$$

Here the inner formulae b and $c \text{ in } \mathbb{Z}$ are typeset in `\small` but we have to return to `\normalsize` before we reach the closing `\$` of the outer formula.

This is handled in the following way:

1. At any point in the document the global variable `\glb@currsiz e` contains the point size for which the math fonts currently are set up.

2. Whenever we start a formula we compare its value with the local variable `\f@size` that describes the current text font size.
3. If both are the same we assume that we can use the current math font setup without adjustment.
4. If they differ we call `\glb@settings` which changes the math font setup and updates `\glb@currsz`.
 - (a) If we are recursively inside another formula (`\if@inmath`) we ensure that `\glb@settings` is executed again in the outer formula, so that the old setup is automatically restored.
 - (b) Otherwise, we set the switch `@inmath` locally to `true` so that all nested formulae will be able to detect that they are nested in some outer formula.

The above algorithm has the following features:

- For sizes which are not containing any formula no math setup is done. Compared to the original algorithm of NFSS this results in the following savings:
 - No unnecessary loading of math fonts for sizes that are not used to typeset any math formulae (explicit or implicit ones).
 - No time overhead due to unnecessary changes of the math font setup on entrance and exit of the text font size.
- Math font setup changes for top-level formulae will survive (there is no restoration after the formula) thus any following formula in the same size will be directly typesettable. Compared to original implementation in NFSS2 the new algorithm has the overhead of one test per formula to see if the current math setup is valid (in the original algorithm the setup was always valid, thus no test was necessary).
- In nested formulae the math font setup is restored in the outer formula by a series of `\aftergroup` commands and checks. Compared to the original algorithm this involves additional checks ($2 \times \langle \text{non-math levels} \rangle$ per inner formula).

5.2.2 Code for math font size setting

`\check@mathfonts` In the `\check@mathfonts` macros we implement the steps 2 to 4 except that instead of a switch the macro `\init@restore@glb@settings` is used.

```

250 <*2kernel | package>
251 \def\check@mathfonts{%
252   \ifx \glb@currsz \f@size
253   <*trace>
254     \ifnum \tracingfonts>\thr@@
255       \@font@info{*** MATH: no change \f@size\space
256         curr/global (\curr@math@size/\glb@currsz)}\fi
257   </trace>
258   \else
259   <*trace>
260     \ifnum \tracingfonts>\thr@@
261       \@font@info{*** MATH: setting up \f@size\space
262         curr/global (\curr@math@size/\glb@currsz)}\fi
263   </trace>

```

```

264     \glb@settings
265     \init@restore@glb@settings
266   \fi
267   \let\curr@math@size\font@size
268   \def\init@restore@glb@settings{\aftergroup\restglb@settings}%
269 }

```

(End definition for \check@mathfonts.)

\init@restore@glb@settings This macro does by default nothing but get redefined inside \check@mathfonts to initiate fontsize restoring in nested formulas.

```

270 <-trace>\let\init@restore@glb@settings\relax
271 <*trace>
272 \def\init@restore@glb@settings{%
273     \ifnum \tracingfonts>\thr@@
274     \font@info{*** MATH: no resetting (not in
275               nested math)}\fi
276 }
277 </trace>

```

(End definition for \init@restore@glb@settings.)

\restglb@settings This macro will be executed the first time after the current formula.

```

278 \def\restglb@settings{%
279 <*trace>
280     \ifnum \tracingfonts>\thr@@
281     \font@info{*** MATH: restoring}\fi
282 </trace>
283     \begingroup
284     \let\font@size\curr@math@size
285     \ifx\glb@currsz \font@size
286 <*trace>
287     \ifnum \tracingfonts>\thr@@
288     \font@info{*** MATH: ... already okay (\font@size)}\fi
289 </trace>
290     \else
291 <*trace>
292     \ifnum \tracingfonts>\thr@@
293     \font@info{*** MATH: ... to \font@size}\fi
294 </trace>
295     \glb@settings
296   \fi
297 \endgroup
298 }

```

(End definition for \restglb@settings.)

5.2.3 Other code for math

\use@mathgroup The \use@mathgroup macro should be used in user macros to select a math group. Depending on whether or not the margid option is in force it has two or three arguments. For this reason it should be called as the last macro.

First we test if we are inside math mode since we don't want to apply a useless definition.

```

299 \def\use@mathgroup#1#2{\relax\ifmmode

```

```

300 <*trace>
301   \ifnum \tracingfonts>\tw@
302     \count@#2\relax
303     \@font@info{Using \noexpand\mathgroup
304               (\the\count@) #2}\fi
305 </trace>

```

If so we first call the ‘=’ macro (i.e. argument three) to set up special things for the selected math group. Then we call `\mathgroup` to select the group given by argument two and finally we place `#1` (i.e. the argument of the *math alphabet identifier*) at the end. This part of the code is surrounded by two commands which behave like `\begingroup` and `\endgroup` if we want *math alphabet identifier*s but will expand into `\@empty` if we want simply switches to a new math group. Since argument number 2 may be a digit instead of a control sequence we add a `\relax`. Otherwise something like `\mit{1}` would switch to math group 11 (and back) instead of printing an oldstyle 1.

```

306   \math@bgroup
307     \expandafter\ifx\csname M@\f@encoding\endcsname#1\else
308       #1\fi
309     \mathgroup#2\relax

```

Before we reinsert the swallowed token (arg. three) into the input stream, in the case that the *math alphabet identifier* isn’t called in math mode, we remove the `\fi` with the `\expandafter` trick. This is necessary if the token is actually an macro with arguments. In such a case the `\fi` will be misinterpreted as the first argument which would be disastrous.

```

310   \expandafter\math@egroup\fi}%

```

The surrounding macros equal `\begingroup` and `\endgroup`. But using internal names makes it possible to overwrite their meaning in certain cases. This is for example used in \mathcal{S} -TeX macros for placing accents.

(End definition for `\use@mathgroup`.)

`\math@egroup` If the `margid` option is in force (which can be tested by looking at the definition of `\math@bgroup` we change the `\math@egroup` command a bit to display the current *math group number* after it closes the scope of *math alphabet* with `\endgroup`.

```

311 <*trace>
312   \ifx\math@bgroup\bgroup
313     \def\math@egroup#1{#1\egroup
314       \ifnum \tracingfonts>\tw@
315         \@font@info{Restoring \noexpand\mathgroup
316                   (\ifnum\mathgroup=\m@ne default\else \the\mathgroup \fi)%
317                   }\fi}
318   \fi
319 </trace>

```

(End definition for `\math@egroup`.)

`\getanddefine@fonts` `\getanddefine@fonts` has two arguments: the *math group number* and the *family/series/shape* name as a control sequence.

```

320 \def\getanddefine@fonts#1#2{%

```

First we turn of tracing when `\tracingfonts` is less than 4.

```

321 <+debug>   \pushtracing
322 <+debug>   \ifnum\tracingfonts<4 \tracingoff
323 <+debug>   \else \tracingon\getanddefine@fonts \fi

```

```

324 <*trace>
325   \ifnum \tracingfonts>\tw@
326   \count#1\relax
327   \@font@info{\noexpand\mathgroup (\the\count@) #1 :=\MessageBreak
328               \string#2 \tf@size/\sf@size/\ssf@size}\fi
329 </trace>

```

We append the current `\tf@size` to #2 to obtain the font name.²⁶ Again, `font@name` is defined globally, for the reasons explained in the description of `\wrong@fontshape`.

```

330   \xdef\font@name{\csname \string#2/\tf@size\endcsname}%

```

Then we call `\pickup@font` to load it if necessary. We remember the internal name as `\textfont@name`.

```

331   \pickup@font \let\textfont@name\font@name

```

Same game for `\scriptfont` and `\scriptscriptfont`:

```

332   \xdef\font@name{\csname \string#2/\sf@size\endcsname}%
333   \pickup@font \let\scriptfont@name\font@name
334   \xdef\font@name{\csname \string#2/\ssf@size\endcsname}%
335   \pickup@font

```

Then we append the new `\textfont...` assignments to the `\math@fonts`.

```

336   \edef\math@fonts{\math@fonts
337                   \textfont#1\textfont@name
338                   \scriptfont#1\scriptfont@name
339                   \scriptscriptfont#1\font@name}%

```

Just before ending this macro we have to pop the tracing stack if it was pushed before.

```

340 <+debug> \poptracing
341   }
342 </2ekernel | package>

```

(End definition for `\getanddefine@fonts`.)

6 Scaled font extraction

`\ifnot@nil` We begin with a simple auxiliary macro. It checks whether its argument is the token `\@nil`. If so, it expands to `\@gobble` which discards the following argument, otherwise it expands to `\@firstofone` which reproduces its argument.

```

343 <*2ekernel>
344 \def\ifnot@nil#1{\def\reserved@a{#1}%
345   \ifx\reserved@a\@nnil \expandafter\@gobble
346   \else \expandafter\@firstofone\fi}

```

(End definition for `\ifnot@nil`.)

`\remove@to@nnil` Three other auxiliary macros will be needed in the following: `\remove@to@nnil` gobbles up everything up to, and including, the next `\@nnil` token, and `\remove@angles` and `\remove@star` do the same for the character `>` and `*`, respectively, instead of `\@nnil`.

```

347 \def\remove@to@nnil#1\@nnil{}
348 \def\remove@angles#1>{\set@simple@size@args}
349 \def\remove@star#1*{#1}

```

²⁶One might ask why this expansion does not generate a macro name that starts with an additional `\` character. The solution is that `\escapechar` is set to `-1` before `\getanddefine@fonts` is called.

(End definition for `\remove@to@nnil`, `\remove@angles`, and `\remove@star`.)

`\extract@sizefn` This macro takes a size specification and parses it into size function and the optional and mandatory arguments.

```

350 \def\extract@sizefn#1*#2\@nil{%
351   \if>#2>\set@size@funct@args#1\@nil
352     \let\sizefn@info\@empty
353   \else\expandafter\set@size@funct@args\remove@star#2\@nil
354     \def\sizefn@info{#1}\fi
355 }
```

(End definition for `\extract@sizefn`.)

`\try@simple@size` This function tries to extract the given size (specified by `\f@size`) for the requested font shape. The font information must already be present in `\font@info`. The central macro that does the real work is `\extract@fontinfo`. We will first give a simple example how this macro works, and describe it in full generality later.

Assume that the requested parameters are: *encoding scheme* ‘OT1’, *family* ‘cm’, *series* ‘sansserif’, *shape* ‘normal’, and *size* ‘12’. The corresponding font definitions have already been extracted from the macro `\OT1/cm/sansserif/normal` and stored in `font@info`. (Otherwise `\extract@fontinfo` doesn’t get called.) This information consists of a token list made of characters of category code 12 of the form

```
<10*>cmss10<12*>cmss12<17*>cmss17
```

For reasonable packages one usually needs more sizes but this is sufficient to get the flavour. We will define a macro `\extract@fontinfo` to find the external font name (‘cmss12’) for us:

```

\def\extract@fontinfo#1<12*#2>#3<#4\@nnil{%
  \set@simple@size@args#3<#4\@nnil
  \execute@size@function{#2}}

```

so that when it gets called via

```
\extract@fontinfo<10*>cmss10<12*>cmss12<17*>cmss17\@nnil
```

#1 will contain all characters before `<12*>`, #2 will be empty, #3 will be exactly `cmss12`, and #4 will be `17>cmss17`. The expansion is therefore

```

\set@simple@size@args cmss12<17*>cmss17\@nnil
\execute@size@function{}

```

This means: the default (empty) size function will be executed, with its optional argument set to empty and its mandatory argument set to `cmss12` by `\set@simple@size@args`. As we discussed earlier, the effect of the default size function is to load the given external font (`cmss12`) at the specified size (12)—which is exactly what was intended.

But this is only part of the whole story. It may be that the size requested does not occur in the token list `\font@info`. And the simple definition of `\extract@fontinfo` we gave above does not allow to specify give more than one size specification in front of the external font name.

Let’s address these two problems separately. The first one is solved with the following trick: We define `\extract@fontinfo` as follows:

```

\def\extract@fontinfo#1<12*#2>#3<#4\@nnil{%
  \ifnot@nil{#3}%
    {\set@simple@size@args#3<#4\@nnil
      \execute@size@function{#2}%
    }%
  }%
}

```

How does this work? We call `\extract@fontinfo` via

```

\expandafter\extract@fontinfo\font@info<12*>\@nil\@nnil

```

i.e. by appending `<12*>\@nil\@nnil`. If the size (`'12'` in this case) appears in `\font@info` everything works as explained above, the only difference being that argument `#4` of `\extract@fontinfo` additionally gets the tokens `<12*>\@nil\@nnil`. However, if the size is not found everything up to the final `<12*>` is in argument `#1`, `#3` gets `\@nil`, and `#2` and `#4` are empty. The macro `\ifnot@nil` will discard the calls to `\set@simple@size@args` and `\execute@size@function`, and hence `\font@info` will continue to be equal to `\@empty`. This means that no simple size specification matching the requested size could be found.

The second problem (more than one simple size specification for one external font name) will be addressed in `\set@simple@size@args` below.

The macros are hidden inside other control sequences so that we have to build `\extract@fontinfo` in several steps.

So here's the actual definition of `\extract@font` in `\try@simple@size`.

```

356 % % this could be replaced by \try@size@range making the subst slower!
357 \def\try@simple@size{%

```

`\reserved@a` is made an abbreviation for the head of the definition of the macro `\extract@fontinfo`.

```

358   \def\reserved@a{\def\extract@fontinfo####1}%

```

Now we can define `\extract@fontinfo`. Here we handle a small but convenient variation: in case of the default (empty) size function it is allowed to omit the `*` character.

```

359   \expandafter\reserved@a\expandafter<\f@size>##2<##3\@nnil{%
360     \ifnot@nil{##2}%
361       {\set@simple@size@args##2<##3\@nnil
362         \execute@size@function\sizefn@info
363       }%

```

Now we call `\extract@fontinfo`. Note the `<\@nil` tokens at the end.

```

364   \expandafter\expandafter
365   \expandafter\extract@fontinfo\expandafter\font@info
366   \expandafter<\f@size>\@nil\@nnil
367 }

```

(End definition for `\try@simple@size`.)

`\set@simple@size@args` As promised above, the macro `\set@simple@size@args` will handle the case of several size specifications in a row. If another size specification follows, the very first token of its argument list is the character `<`. By starting the definition as follows,

```

368 \def\set@simple@size@args#1<{%

```

parameter #1 is empty in this case, and contains the size function's arguments otherwise. We distinguish these two cases (Note that the character < cannot appear in #1) by calling `\remove@angles` for empty #1 and `\extract@sizefn` otherwise. In the latter case we have to take care of the remaining character tokens and discard them. This is done by `\remove@to@nnil`. Note also the use of Kabelschacht's method.

```

369         \if<#1<%
370             \expandafter\remove@angles
371         \else
372             \extract@sizefn#1*\@nil
373             \expandafter\remove@to@nnil
374         \fi}

```

(End definition for \set@simple@size@args.)

Now, we are through with the case of a simple size, except for calling the size function. This will be handled later, as it is the same mechanism for all types of size specification. We will now proceed to macros for extraction of size range specification.

\extract@rangefontinfo `\extract@rangefontinfo` goes through a font shape definition in the input until it recognizes the tokens `<\@nil->`. It looks for font ranges with font size functions. It's operation is rather simple: it discards everything up to the next size specification and passes this on to `\is@range` for inspection. The specification (parameter #2 is inserted again, in case it is needed later.

```

375 \def\extract@rangefontinfo#1<#2>{%
376     \is@range#2->\@nil#2>}

```

(End definition for \extract@rangefontinfo.)

\is@range `\is@range` is again a sort of dispatcher macro: if the size specification it is looking at is not a range specification it discards it and calls `\extract@rangefontinfo` to continue the search. Otherwise it calls `\check@range` to check the requested size against the specified range.

From the way `\is@range` is called inside `\extract@rangefontinfo` we see that #2 is the character > if the size specification found is a simple one (as it does not contain a - character. This is checked easily enough and `\extract@rangefontinfo` called again. Note that the extra tokens inserted after the `\@nil` in the call to `\is@range` appear at the beginning of the first argument to `\extract@rangefontinfo` and are hence ignored.

```

377 \def\is@range#1-#2\@nil{%
378     \if>#2\expandafter\check@single\else
379     \expandafter\check@range\fi}

```

(End definition for \is@range.)

\check@range `\check@range` takes lower bound as parameter #1, upper bound as #2, size function as #3 and the size function's arguments as #4. If #3 is the special token `\@nil \font@info` is exhausted and we can stop searching.

```

380 \def\check@range#1-#2>#3<#4\@nnil{%
381     \ifnot@nil{#3}{%

```

If #3 wasn't `\@nil` we have a range. We start by assuming that we have to recurse. Note that we have to reinsert an < as it was already removed by scanning.

```

382     \def\reserved@f{\extract@rangefontinfo<#4\@nnil}%

```

We have to make sure that both boundaries are present, if not we have to set them. Here we check the upper bound. If `\upper@bound` is zero after the assignment we set it to `\maxdimen` (upper open range). We need to use a *<dimen>* register for the scan since we may have a decimal number as the boundary.

```
383      \upper@bound0#2\p@
384      \ifdim\upper@bound=\z@ \upper@bound\maxdimen\fi
```

Now we check the upper boundary against `\f@size`. If it is larger or equal than `\f@size` this range is no good and we have to recurse.

```
385      \ifdim \f@size \p@<\upper@bound
```

Otherwise we have to check the lower bound. This time it is not necessary to scan the boundary value into a register because if it is empty we get zero as desired. We could even omit the 0 which would result in 1pt as default lower boundary. If `\f@size` is smaller than the boundary we have to recurse.

```
386      \lower@bound0#1\p@
387      \ifdim \f@size \p@<\lower@bound
388      \else
```

If both tests are passed we can try executing the size function.

```
389      \set@simple@size@args#3<#4\@nnil
390      \execute@size@function\sizefn@info
```

If the function was successful it should have left an external font name in `\external@font`. We use this to see if we can stop scanning. Otherwise we recurse.

```
391      \ifx\external@font\@empty
392      \else
393      \let\reserved@f\@empty
394      \fi
395      \fi
396      \fi
397      \reserved@f}}
```

(End definition for `\check@range`.)

`\lower@bound` We use two *dimen* registers `\lower@bound` and `\upper@bound` to store the lower and
`\upper@bound` upper endpoints of the range we found.

```
398 \newdimen\lower@bound
399 \newdimen\upper@bound
```

(End definition for `\lower@bound` and `\upper@bound`.)

`\check@single` `\check@single` takes the size as parameter #1, size function as #2 and the size function's arguments as #3. We can assume that there is always something in the pipeline since the very last entry is a faked range (see above).

```
400 \def\check@single#1>#2<#3\@nnil{%
```

We start by assuming that we have to recurse. Note that we have to reinsert an < as it was already removed by scanning.

```
401      \def\reserved@f{\extract@rangefontinfo<#3\@nnil}%
```

Now we check the size against `\f@size`. If it is not equal `\f@size` it is no good and we have to recurse.

```
402      \ifdim \f@size \p@=#1\p@
```


Otherwise if this test is passed we can try executing the size function.

```
403      \set@simple@size@args#2<#3\@nnil
404      \execute@size@function\sizefn@info
```

If the function was successful it should have left an external font name in `\external@font`. We use this to see if we can stop scanning. Otherwise we recurse.

```
405      \ifx\external@font\@empty
406      \else
407      \let\reserved@f\@empty
408      \fi
409      \fi
410      \reserved@f}
```

(End definition for \check@single.)

\set@size@funct@args This macro sets the optional and mandatory arguments for a size function. If the optional argument is not present it is set to the empty token list. The mandatory argument is delimited by the token `\@nil`.

```
411 \def\set@size@funct@args{\@ifnextchar[%
412   \set@size@funct@args@{\set@size@funct@args@[]}}
413 \def\set@size@funct@args@[#1]#2\@nil{%
414   \def\mandatory@arg{#2}%
415   \def\optional@arg{#1}}
416 \</2ekernel>
```

(End definition for \set@size@funct@args and \set@size@funct@args@.)

\DeclareSizeFunction This function defines a new size function hiding the internal from the designer. The body of the size function may use `\optional@arg` and `\mandatory@arg` denoting the optional and mandatory argument that may follow the size specification `<...>`.

```
417 \<2ekernel>
418 \def\DeclareSizeFunction#1#2{\@namedef{s@fct@#1}{#2}}
419 \@onlypreamble\DeclareSizeFunction
420 \</2ekernel>
```

(End definition for \DeclareSizeFunction.)

\execute@size@function This macro is very simple. The only point worth noting is that calling an undefined size function will do nothing (actually execute a `\relax`).

```
421 \<2ekernel | package>
422 \def\execute@size@function#1{%
423   \<trace>
424     \ifundefined{s@fct@#1}%
425       {\errmessage{Undefined font size function #1}%
426        \s@fct@}%
427     {\csname s@fct@#1\endcsname}%
428   \</trace>
429   \<-trace> \csname s@fct@#1\endcsname
430 }
431 \</2ekernel | package>
```

(End definition for \execute@size@function.)

`\try@size@range` This macro tries to find a suitable range for requested size (specified by `\f@size`) in `\font@info`. All the relevant action is done in `\extract@rangefontinfo`. All that needs to be done is to stuff in the token list in `\font@info` so that `\extract@rangefontinfo` can inspect it. Note the `<-*\@nil>` token at the end to stop scanning.

```

432 {*2ekernel}
433 \def\try@size@range{%
434     \expandafter\extract@rangefontinfo\font@info <-*\@nil<\@nnil
435 }

```

(End definition for \try@size@range.)

`\try@size@substitution` This is the last thing that can be tried. If the desired `\f@size` is found neither among the simple size specifications nor in one of the ranges the whole list of size specifications is searched for a nearby simple size.

```

436 \gdef\try@size@substitution{%

```

First we do some initializations. `\@tempdimb` will hold the difference between the wanted size and the best solution found so far, so we initialise it with `\maxdimen`. The macro `\best@size` will hold the best size found, nothing found is indicated by the empty value.

```

437     \@tempdimb \maxdimen
438     \let \best@size \empty

```

Now we loop over the specification

```

439     \expandafter \try@simples \font@info <\number\M>\@nil<\@nnil
440 }

```

(End definition for \try@size@substitution.)

`\font@submax` The macro `\font@submax` records the maximal deviation from the desired size encountered so far. Its value is used in a warning message at `\end{document}`. The macro `\fontsubfuzz` contains the amount that will not cause terminal warnings (warnings still go into the transcript file).

```

441 \def\font@submax{0pt}
442 \def\fontsubfuzz{.4pt}
443 {/2ekernel}
444 {+package}\def\fontsubfuzz{0pt}

```

(End definition for \font@submax and \fontsubfuzz.)

`\try@simples` `\try@simples` goes through a font shape definition in the input until it recognizes the tokens `<*\@nil>`. It looks for simple sizes to determine the two closest sizes. It is assumed that simple sizes are in increasing order.

```

445 {*2ekernel}
446 \gdef\try@simples#1<#2>{%
447     \tryif@simple#2->\tryif@simple}

```

(End definition for \try@simples.)

`\tryis@simple` `\tryis@simple` is similar to `\is@range`. If it sees a simple size, it checks it against the value of `\f@size` and sets `\lower@font@size` or `\higher@font@size`. In the latter case, it stops the iteration. By adding `<\number\M>` at the end of the line we always have an end point. This is a hack which probably should be corrected.

First it checks whether it is finished already, then whether the size specification in question is a simple one.

```

448 \gdef\tryif@simple#1-#2\tryif@simple{%

```

Most common case for `\reserved@f` first:

```
449 \let \reserved@f \try@simples
450 \if>#2%
```

If so, it compares it to the value of `\f@size`. This is done using a `dimen` register since there may be fractional numbers.

```
451 \dimen@ #1\p@
452 \ifdim \dimen@<\M\p@
```

If `\dimen@` is `\M\p@` we have reached the end of the fontspec (hopefully) otherwise we compare the value with `\f@size` and compute in `\@tempdimc` the absolute value of the difference between the two values.

```
453 \ifdim \f@size\p@<\dimen@
454 \@tempdimc \dimen@
455 \advance\@tempdimc -\f@size\p@
456 \else
457 \@tempdimc \f@size\p@
458 \advance\@tempdimc -\dimen@
459 \fi
```

The result is then compared with the smallest difference we have encountered, if the new value (in `\@tempdimc` is smaller) we have found a size which is a better approximation so we make it the `\best@size` and adjust `\@tempdimb`.

```
460 \ifdim \@tempdimc<\@tempdimb
461 \@tempdimb \@tempdimc
462 \def \best@size{#1}%
463 \fi
```

When we have reached the end of the fontspec we substitute the best size found (if any). We code this inline to save macro space; in the past this was done by a macro called `\subst@size`.

```
464 \else
```

This macro substitutes the size recorded in `\best@size` for the unavailable size `\f@size`. `\subst@size` `\font@submax` records the maximum difference between desired size and selected size in the whole run.

```
465 % \subst@size %% coded inline
466 % \def\subst@size{%
467 \ifx \external@font\@empty
468 \ifx \best@size\@empty
469 \else
470 \ifdim \@tempdimb>\font@submax \relax
471 \xdef \font@submax {\the\@tempdimb}%
472 \fi
473 \let \f@user@size \f@size
474 \let \f@size \best@size
475 \ifdim \@tempdimb>\fontsubfuzz\relax
476 \@font@warning{Font\space shape\space
477 '\curr@fontshape'\space in\space size\space
478 <\f@user@size>\space not\space available\MessageBreak
479 size\space <\f@size>\space substituted}%
480 \fi
481 \try@simple@size
482 \do@subst@correction
```

```

483     \fi
484 \fi
485 % %}

```

This brings us back into the main part of `\tryif@simple`. Finally we get rid of any rubbish left over on the input stack.

```

486     \let \reserved@f \remove@to@nnil
487     \fi
488 \fi

```

If it's a range iterate also.

```

489 \reserved@f}

```

(End definition for \tryis@simple and \subst@size.)

6.1 Sizefunctions

In the following we define some useful size functions.

`\s@fct@` This is the default size function. Mandatory argument is an external font name, optional argument a scale factor. The font is scaled to `\f@size` if no optional argument is present, and to `\f@size` multiplied by the optional argument otherwise.

```

490 \DeclareSizeFunction{}{\empty@sfcnt\@font@warning}
491 \DeclareSizeFunction{s}{\empty@sfcnt\@font@info}
492 \def\empty@sfcnt#1{%
493     \@tempdimb \f@size\p@
494     \ifx\optional@arg\@empty
495     \else
496         \@tempdimb \optional@arg\@tempdimb
497         #1{Font\space shape\space '\curr@fontshape'\space
498             will\space be\MessageBreak
499             scaled\space to\space size\space \the\@tempdimb}%
500     \fi
501     \edef\external@font{\mandatory@arg\space at\the\@tempdimb}}

```

(End definition for \s@fct@.)

`\s@fct@gen` This size function generates the external name from the mandatory argument and the requested user size, and thus can be used for external names where the size is encoded in the font name. The optional argument a scale factor. The font is scaled to `\f@size` if no optional argument is present, and to `\f@size` multiplied by the optional argument otherwise.

```

502 \DeclareSizeFunction{gen}{\gen@sfcnt\@font@warning}
503 \DeclareSizeFunction{sgen}{\gen@sfcnt\@font@info}
504 \def\gen@sfcnt{%
505     \edef\mandatory@arg{\mandatory@arg\f@size}%
506     \empty@sfcnt}

```

(End definition for \s@fct@gen and \s@fct@sgen.)

`\sfct@genb` This size function is similar to `gen`, but for fonts where the size is encoded in the font name in centipoints, as in the DC fonts version 1.2. The font is scaled to `\f@size` if `\sfct@sgenb` no optional argument is present, and to `\f@size` multiplied by the optional argument otherwise.

```

507 \DeclareSizeFunction{genb}{\genb@sfcnt\@font@warning}
508 \DeclareSizeFunction{sgenb}{\genb@sfcnt\@font@info}

509 \def\genb@sfcnt{%
510     \edef\mandatory@arg{\mandatory@arg\expandafter\genb@x\f@size..\@{}}%
511     \empty@sfcnt}

```

(End definition for `\sfct@genb` and `\sfct@sgenb`.)

`\genb@x` The auxiliary macros `\genb@x` and `\genb@y` are used to convert the `\f@size` into centi-
`\genb@y` points.

```

512 \def\genb@x#1.#2.#3\@{\two@digits{#1}\genb@y#200\@{}}
513 \def\genb@y#1#2#3\@{\#1#2}

```

(End definition for `\genb@x` and `\genb@y`.)

`\sfct@sub` This size function handles font substitution. The mandatory argument is a family/series/shape combination, the optional argument (if present) is ignored. The font encoding scheme cannot be changed. Therefore, the first thing we do is to prepend the encoding scheme.

```

514 \DeclareSizeFunction{sub}{\sub@sfcnt\@font@warning}
515 \DeclareSizeFunction{ssub}{\sub@sfcnt\@font@info}

516 \def\sub@sfcnt#1{%
517     \edef\mandatory@arg{\f@encoding/\mandatory@arg}%

```

Next action is split the arg into its individual components and allow for a late font shape load.

```

518     \begingroup
519     \expandafter\split@name\mandatory@arg/\@nil
520     \try@load@fontshape
521     \endgroup

```

Then we record the current `\f@size` since it may get clobbered.

```

522     \let\f@user@size\f@size

```

Then we check whether this new combination is defined and give an error message if not. In this case we also switch to `\error@fontshape`.

```

523     \expandafter
524     \ifx\csname\mandatory@arg\endcsname\relax
525         \errmessage{No\space declaration\space for\space
526             shape\space \mandatory@arg}%
527         \error@fontshape
528     \else

```

Otherwise we warn the user about the substitution taking place.

```

529         #1{Font\space shape\space '\curr@fontshape'\space in\space
530             size\space <\f@size>\space not\space available\MessageBreak
531             Font\space shape\space '\mandatory@arg'\space tried\space
532             instead}%
533         \expandafter\split@name\mandatory@arg/\@nil
534     \fi

```

Then we restart the font specification scan by calling `\get@external@font`.

```
535 \edef\font@size{\font@user@size}%
536 \get@external@font
```

Finally `\do@subst@correction` is called to get the font name right.

```
537 \do@subst@correction
538 }
```

(End definition for \sfct@sub.)

`\font@aliasinfo` Sometimes a substitution is only done to map a long font name to a standard shape or series, e.g.,

```
DeclareFontShape{T1}{Roboto-LF}{b}{it}{<-> alias * Roboto-LF/bold/it}{}
```

Using the `ssub` function in that case will give a strange (and incorrect) warning. As an alternative we therefore offer the size function `alias`. It will still add some info into the `.log` file, but no longer complains that the font shape is not available. It is implemented by grabbing the default warning text and replacing it with a new one.

```
539 </2ekernel>
540 <*2ekernel | latexrelease>
541 <latexrelease>\IncludeInRelease{2020/02/02}%
542 <latexrelease>{\font@aliasinfo}{alias size function}%
543 \DeclareSizeFunction{alias}{\sub@sfcnt\font@aliasinfo}
544 \def\font@aliasinfo#1{%
545 \font@info{Font\space shape\space '\curr@fontshape'\space
546 aliased\space to\MessageBreak '\mandatory@arg'}%
547 }
548 </2ekernel | latexrelease>
549 <latexrelease>\EndIncludeInRelease
550 <latexrelease>\IncludeInRelease{0000/00/00}%
551 <latexrelease>{\font@aliasinfo}{alias size function}%
552 <latexrelease>\let\sfct@alias\undefined
553 <latexrelease>\let\font@aliasinfo\undefined
554 <latexrelease>
555 <latexrelease>\EndIncludeInRelease
556 <*2ekernel>
```

(End definition for \font@aliasinfo.)

`\sfct@subf` The `subf` size function allows substitution of another font. The mandatory argument is the external name of the font to be substituted, the optional argument a size scaling factor like in the default size function. The main difference to the default size function is the warning message.

```
557 \DeclareSizeFunction{subf}{\sub@sfcnt\font@warning}
558 \DeclareSizeFunction{ssubf}{\sub@sfcnt\font@info}

559 \def\sub@sfcnt#1{%
560 #1{Font\space shape\space '\curr@fontshape'\space in\space
561 size\space \font@size\space not\space available\MessageBreak
562 external\space font\space '\mandatory@arg'\space used}%
563 \empty@sfcnt#1%
564 }
```

(End definition for \sfct@subf.)

`\s@fct@fixed` The `fixed` size function is for using a font at a different size than requested. A warning message is printed, and the external font to be used is taken from the mandatory argument. If an optional argument is present it is used as the ‘at’ size for the font. Otherwise the font is loaded at its design size.

```

565 \DeclareSizeFunction{fixed}{\fixed@sfcnt\@font@warning}
566 \DeclareSizeFunction{sfixed}{\fixed@sfcnt\@font@info}

567 \def\fixed@sfcnt#1{%
568   \ifx\optional@arg\@empty
569     \let\external@font\mandatory@arg
570   \else
571     \edef\external@font{\mandatory@arg\space at\optional@arg pt}%
572   \fi
573   #1{External\space font\space ‘\external@font’\space loaded\space
574     for\space size\MessageBreak
575     <\f@size>}%
576 }
577 \</2ekernel>

```

(End definition for \s@fct@fixed.)

File x

ltfsscmp.dtx

This file contains the implementation of commands giving compatibility with the original ‘NFSS1’ release of the Font Selection Scheme.

Warning: The macro documentation is still basically the documentation from the first NFSS release and therefore in some cases probably not completely accurate.

Version 1 of NFSS is obsolete now for about 20 years (and was “current” only for a short intermediate time) so with the 2015 release these internal interface commands are removed from the kernel and made available via `latexrelease` package so that backward compatibility remains ensured for very old documents.

```
1 (*latexrelease)
2 \IncludeInRelease{2015/01/01}{\new@fontshape}%
3                                     {NFSS version1 commands}%
4 \let\new@fontshape\@undefined
5 \let\warn@rel@i\@undefined
6 \let\scan@fontshape\@undefined
7 \let\scan@@fontshape\@undefined
8 \let\subst@fontshape\@undefined
9 \let\extra@def\@undefined
10 \let\default@mextra\@undefined
11 \let\preload@sizes\@undefined
12 \let\err@rel@i\@undefined
13 \let\newmathalphabet\@undefined
14 \let\newmathalphabet@\@undefined
15 \let\newmathalphabet@@@\@undefined
16 \let@if@no@font@opt\@undefined
17 \let@no@font@optfalse\@undefined
18 \let\define@mathalphabet\@undefined
19 \let\define@mathgroup\@undefined
20 \let\addtoversion\@undefined
21 \EndIncludeInRelease
```

In older releases we provide the original definitions.

```
22 \IncludeInRelease{0000/00/00}{\new@fontshape}%
23                                     {NFSS version1 commands}%
```

`\new@fontshape` The interface is now `\DeclareFontShape`.

```
24 \gdef\new@fontshape#1#2#3#4{%
25     \warn@rel@i\new@fontshape\DeclareFontShape
26     \expandafter\scan@fontshape\@gobble#4<\@nil><<%
27     \DeclareFontShape U{#1}{#2}{#3}\reserved@f}%
28 \@onlypreamble\new@fontshape
```

(End definition for \new@fontshape.)

`\warn@rel@i` The warning message used above.

```
29 \gdef\warn@rel@i#1#2{%
30     \@font@warning{*** NFSS release 1 command
```



```

31          \noexpand#1found\MessageBreak
32      *** Update by using release 2 command
33          \string#2.\MessageBreak
34      *** Recovery is probably possible}%
35 }%
36  \@onlypreamble\warn@rel@i

```

(End definition for \warn@rel@i.)

\scan@fontshape This will scan the old font shape definition syntax.

```

37  \gdef\scan@fontshape{%
38      \let\reserved@f\@empty
39      \let\reserved@e\@empty %           holds last info
40      \scan@@fontshape
41  }%
42  \@onlypreamble\scan@fontshape

```

(End definition for \scan@fontshape.)

\scan@@fontshape

```

43  \gdef\scan@@fontshape#1>#2#3<{%
44      \ifx\@nil#1%
45          \edef\reserved@f{\reserved@f\reserved@e}%
46      \else
47          \def\reserved@b{#1}%           nick names
48          \def\reserved@c{#3}%
49          \in@{ at}{#3}%
50          \ifin@
51              \in@{pt}{#3}% not a proof but a good chance
52          \ifin@

```

We grab also everything after pt and discard it if people have forgotten to place a percent sign there.

```

53      \def\reserved@a##1 at##2pt##3\@nil{%
54          \def\reserved@b{##2}%
55          \def\reserved@c{##1}%
56      }%
57      \reserved@a#3\@nil
58      \fi
59      \fi
60      \ifnum 0<0#2
61          \edef\reserved@d{subf*\reserved@c}%
62          \ifcase #2\or
63              \or
64              \else
65                  \errmessage{*** What's this? NFSS release 0? ***}%
66              \fi
67          \else
68              \edef\reserved@d{#2\reserved@c}%
69          \fi
70          \ifx\reserved@d\reserved@e
71              \edef\reserved@f{\reserved@f<\reserved@b>}%
72          \else
73              \edef\reserved@f{\reserved@f\reserved@e<\reserved@b>}%add old info
74              \let\reserved@e\reserved@d

```

```

75     \fi
76     \expandafter\scan@@fontshape
77     \fi
78 }%
79 \@onlypreamble\scan@@fontshape

```

(End definition for \scan@@fontshape.)

\subst@fontshape This is now also handled by the extend syntax of \DeclareFontShape.

```

80 \gdef\subst@fontshape#1#2#3#4#5#6{%
81     \warn@rel@i\subst@fontshape\DeclareFontShape
82     \DeclareFontShape{U}{#1}{#2}{#3}{<->sub*#4/#5/#6}{}}%
83 \@onlypreamble\subst@fontshape

```

(End definition for \subst@fontshape.)

\extra@def This was replaced by \DeclareFontFamily.

```

84 \gdef\extra@def#1#2#3{%
85     \warn@rel@i\extra@def\DeclareFontFamily
86     \DeclareFontFamily{U}{#1}{}%
87 }%
88 \@onlypreamble\extra@def

```

(End definition for \extra@def.)

\default@mextra The new name is \DeclareFontEncodingDefaults but in this case we don't feel comfortable with this either.

```

89 \gdef\default@mextra{%
90     \warn@rel@i\default@mextra\DeclareFontEncodingDefaults

```

We pick up the argument to \default@mextra implicitly as the second argument of \DeclareFontEncodingDefaults.

```

91     \DeclareFontEncodingDefaults\relax
92 }%
93 \@onlypreamble\default@mextra

```

(End definition for \default@mextra.)

\preload@sizes The new interface is \DeclarePreloadSizes.

```

94 \gdef\preload@sizes{%
95     \warn@rel@i\preload@sizes\DeclarePreloadSizes
96     \DeclarePreloadSizes U%
97 }%
98 \@onlypreamble\preload@sizes

```

(End definition for \preload@sizes.)

\err@rel@i This macro is used in cases where emulation with NFSS2 features is not really possible.

```

99 \gdef\err@rel@i#1#2{%
100     \@latex@error{*** NFSS release 1 command \noexpand#1found%
101         ^^J*** Recovery not possible. Use \string#2}%
102     {The new release of NFSS doesn't support the
103     \noexpand#1command^^Jany longer.
104     Please upgrade your file to the syntax of NFSS
105     release 2^^Jusing the \noexpand#2command.}%

```

Let's die.

```
106 \batchmode\input.\relax
107 }%
108 \@onlypreamble\err@rel@i
```

(End definition for \err@rel@i.)

```
\newmathalphabet \newmathalphabet is the old form.
\newmathalphabet@@ 109 \gdef\newmathalphabet{%
\newmathalphabet@@@ 110 \if@no@font@opt
111 \latex@error{*** NFSS release 1 command
112 \noexpand\newmathalphabet found%
113 ^^J \space*** Automatic recovery not possible.%
114 ^^J \space*** TYPE H for Help%
115 }%
116 {Please look at the file usrguide.tex for hints on
117 how to resolve this problem.}%
118 \else
119 \warn@rel@i\newmathalphabet\DeclareMathAlphabet
120 \fi
121 \@ifstar\newmathalphabet@@@
122 \newmathalphabet@@}%
123 \gdef\newmathalphabet@@#1{\DeclareMathAlphabet#1{U}{-}{-}{-}}%
124 \gdef\newmathalphabet@@@#1#2#3#4{%
125 \DeclareMathAlphabet{#1}{U}{#2}{#3}{#4}}%
126 \@onlypreamble\newmathalphabet
127 \@onlypreamble\newmathalphabet@@
128 \@onlypreamble\newmathalphabet@@@
```

(End definition for \newmathalphabet, \newmathalphabet@@, and \newmathalphabet@@@.)

```
\if@no@font@opt
\@no@font@optfalse 129 \global\let\if@no@font@opt\iftrue
130 \gdef\@no@font@optfalse{\let\if@no@font@opt\iffalse}%
131
```

(End definition for \if@no@font@opt and \@no@font@optfalse.)

```
\define@mathalphabet This is a case where dying is best.
131 \gdef\define@mathalphabet{%
132 \err@rel@i\define@mathalphabet\DeclareMathAlphabet
133 }%
134 \@onlypreamble\define@mathalphabet
```

(End definition for \define@mathalphabet.)

```
\define@mathgroup And here is another one
135 \gdef\define@mathgroup{%
136 \err@rel@i\define@mathgroup\DeclareSymbolFont
137 }%
138 \@onlypreamble\define@mathgroup
```

(End definition for \define@mathgroup.)

```

\addtoversion \addtoversion is the old form.
139 \def\addtoversion#1#2{%
140   \warn@rel@i\addtoversion\SetMathAlphabet
141   \SetMathAlphabet#2{#1}{U}}%
142 \@onlypreamble\addtoversion

(End definition for \addtoversion.)
    Finishing off this huge \IncludeInRelease argument:
143 \EndIncludeInRelease
144 </latexrelease>

```

File y

ltfssdcl.dtx

This file contains the main implementation of the font selection scheme commands. See other parts of the L^AT_EX distribution, or *The L^AT_EX Companion* for higher level documentation of these commands.

Warning: The macro documentation is still basically the documentation from the first NFSS release and therefore in some cases probably not completely accurate.

1 Interface Commands

`\in@` `\@in` is a utility macro with two arguments. It determines whether its first argument occurs in its second and sets the switch `\ifin@` accordingly. The first argument may not contain braces nor # (more precisely, tokens of category code 1, 2, or 6).

```

1  <*2kernel>
2  \def\in@#1#2%
3  {%
4    \begingroup
5      \def\in@@##1#1{%
6        \toks@{\expandafter{\in@@#2{}}#1}%
7        \edef\in@@{\the\toks@}%
8        \expandafter\endgroup
9        \ifx\in@@\@empty
10         \in@false
11       \else
12         \in@true
13       \fi
14     }
15 \newif\ifin@

```

(End definition for `\in@` and `\ifin@`.)

Before the `\begin{document}` command several *math versions* and *math alphabet identifiers* may be declared. In principle, there should be exactly one family/series/shape combination be declared for each version/alphabet pair. But we want to allow for defaults as well for automagical filling of holes.

While building the tables for math alphabet identifiers and math versions we keep several lists:

- the list of all math versions, `\version@list`, each entry prefixed by the control sequence `\version@elt`, i.e. this list has the following form

```

\version@elt<version1>\version@elt<version2>...
\version@elt<versionn>

```

- the list of all math alphabet identifiers. Here every entry has the form:

```

\group@elt<math group number>
{\{<default family>\}{<default series>\}{<default shape>}}.

```

- Each defined math alphabet identifier holds a list containing Information about the *versions* for which it is defined. This list has a more complicated structure: it looks as follows:

```

\set@alpha<the alphabet identifier itself>
\reserved@c<math version><font info>
...
\@nil

```

where ** is either `\reserved@e` (if the combination is not defined yet) or

```
{\family}{\series}{\shape}}
```

`\version@list` We initialize the version list to be empty.

```

16 \let\version@list=\@empty
17 \@onlypreamble\version@list

```

(End definition for `\version@list`.)

`\version@elt`

```

18 \let\version@elt\relax
19 \@onlypreamble\version@elt

```

(End definition for `\version@elt`.)

`\new@mathversion` The macro `\new@mathversion` is called with the version control sequence as its argument.

```

20 %\def\new@mathversion#1{%

```

The first thing this macro does is to check if the version identifier is already present in `\version@list`. We enclose `\version@list` in braces since it might be empty (if no *version* is defined yet). But this means that we need a suitable number of `\expandafter` primitives.

```

21 % \expandafter\in@\expandafter#1\expandafter{\version@list}%
22 % \ifin@

```

If so it prints an error message. The `\next` macro is used to get rid of the four characters `\mv@` that would otherwise appear at the begin of the version name in the error message.

```

23 % \latex@error{Math version
24 % '\expandafter\@gobblefour\string#1'
25 % already defined}\@eha

```

Otherwise we have a new version, and we can proceed with entering it into the tables. We add it to `\version@list`. This is very easy: we define `\version@elt` (which is the delimiter in `\version@list`) to protect itself and the following token from being expanded and simply redefine `\version@list`.

```

26 % \else
27 % \global\expandafter\newcount\csname c@\expandafter
28 % \gobble\string#1\endcsname
29 % \global\csname c@\expandafter
30 % \gobble\string#1\endcsname\@ne
31 % \def\version@elt{\noexpand\version@elt\noexpand}%
32 % \edef\version@list{\version@list\version@elt#1}%

```

Then we prepare to enter the new version into all math alphabet identifier lists. Remember that these lists use `\reserved@c` as delimiter, and that there appears the control sequence `\reserved@e` that must not be expanded. Therefore we take suitable precautions.

```
33 %      \def\reserved@c{\noexpand\reserved@c\noexpand}%
34 %      \let\reserved@e\relax
```

We will now go through the `\alpha@list` to process every *math alphabet identifier* in turn. Since this list has `\group@elt` as a delimiter we define this control sequence. It has three arguments as every entry consists of three items (as explained above).

```
35 %      \def\group@elt##1##2##3{%
```

The first of these arguments is the *math alphabet identifier*. We redefine it by appending the information about the new version at the end of the list contained in it. However, there is one subtlety: the definitions for `\reserved@c` and `\reserved@e` made above prevent the main part of the list from being expanded. But we still have to take care of the header and the trailer. To do this we remove the trailer by means of the macro `\remove@nil` which also protect the header from being expanded. Its definition is given below. Now we can prepare to add the new version.

```
36 %          \edef##1{\expandafter\remove@nil##1%
37 %              \reserved@c
38 %              #1%
39 %              \reserved@e
40 %              \noexpand\@nil}}%
```

Finally we call `\alpha@list` which will now execute the macro `\group@elt` once for every defined *math alphabet identifier*. And that's all for now.

```
41 %      \alpha@list
42 %  \fi}
```

(End definition for `\new@mathversion`.)

`\alpha@list` As we explained above every entry in `\alpha@list` has the form
`\alpha@elt`
*alphabet identifier**internal group number**default font assignments*...
We initialize it to `\@empty`.

```
43 \let\alpha@list\@empty
44 \@onlypreamble\alpha@list
```

(End definition for `\alpha@list`.)

`\alpha@elt`

```
45 \let\alpha@elt\relax
46 \@onlypreamble\alpha@elt
```

(End definition for `\alpha@elt`.)

`\newgroup` Start the group (fam) allocation at 0. (Doesn't belong here.)

```
47 \count18=-1
```

(End definition for `\newgroup`.)

`\stepcounter`

(End definition for `\stepcounter`.)

`\select@group` We surround `\select@group` with braces so that functions using it can be used directly after `_` or `^`. However, if we use oldstyle syntax where the math alphabet doesn't have arguments (ie if `\math@bgroup` is not `\bgroup`) we need to get rid of the extra group.

```

48 </2ekernel>
49 <latexrelease>\IncludeInRelease{2015/01/01}
50 <latexrelease>          {\select@group}{\select@group}%
51 <*2ekernel | latexrelease>
52 \def\select@group#1#2#3#4{%
53   \ifx\math@bgroup\bgroup\else\relax\expandafter\@firstofone\fi
54   {%
55     \ifmmode
56       \ifnum\csname c@mv@\math@version\endcsname<\e@mathgroup@top
57         \begingroup
58           \escapechar\m@ne
59           \getanddefine@fonts{\csname c@mv@\math@version\endcsname}#3%
60           \globaldefs\@ne \math@fonts
61         \endgroup
62         \init@restore@version
63         \xdef#1{\noexpand\use@mathgroup\noexpand#2%
64           {\number\csname c@mv@\math@version\endcsname}}}%
65         \global\advance\csname c@mv@\math@version\endcsname\@ne
66       \else
67         \let#1\relax
68         \@latex@error{Too many math alphabets used in
69           version \math@version}%
70         \@eha
71       \fi
72     \else \expandafter\non@alpherr\fi
73     #1{#4}%
74   }%
75 }
76 </2ekernel | latexrelease>
77 <latexrelease>\EndIncludeInRelease
78 <latexrelease>\IncludeInRelease{0000/00/00}
79 <latexrelease>          {\select@group}{\select@group}%
80 <latexrelease>\def\select@group#1#2#3#4{%
81 <latexrelease> \ifx\math@bgroup\bgroup\else\relax\expandafter\@firstofone\fi
82 <latexrelease> {%
83 <latexrelease> \ifmmode
84 <latexrelease>   \ifnum\csname c@mv@\math@version\endcsname<\sixt@@n
85 <latexrelease>     \begingroup
86 <latexrelease>       \escapechar\m@ne
87 <latexrelease>       \getanddefine@fonts
88 <latexrelease>         {\csname c@mv@\math@version\endcsname}#3%
89 <latexrelease>       \globaldefs\@ne \math@fonts
90 <latexrelease>     \endgroup
91 <latexrelease>     \init@restore@version
92 <latexrelease>     \xdef#1{\noexpand\use@mathgroup\noexpand#2%
93 <latexrelease>       {\number\csname c@mv@\math@version\endcsname}}}%
94 <latexrelease>     \global\advance\csname c@mv@\math@version\endcsname\@ne
95 <latexrelease>   \else
96 <latexrelease>     \let#1\relax
97 <latexrelease>     \@latex@error{Too many math alphabets used in
98 <latexrelease>       version \math@version}%

```



```

99 <latexrelease>      \@eha
100 <latexrelease>      \fi
101 <latexrelease> \else \expandafter\non@alpherr\fi
102 <latexrelease> #1{#4}%
103 <latexrelease> }%
104 <latexrelease>}
105 <latexrelease>\EndIncludeInRelease
106 <*2ekernel>

107 \@onlypreamble\restore@mathversion

```

(End definition for \select@group.)

\init@restore@version

```

108 \def\init@restore@version{%
109     \global\let\init@restore@version\relax
110     \xdef\restore@mathversion
111         {\expandafter\noexpand\csname mv@\math@version\endcsname
112          \global\csname c@mv@\math@version\endcsname
113          \number\csname c@mv@\math@version\endcsname\relax}%
114     \aftergroup\dorestore@version
115 }
116 \@onlypreamble\init@restore@version

```

(End definition for \init@restore@version.)

\non@alpherr

```

117 \gdef\non@alpherr#1{\@latex@error{%

```

The command here will have a space at the end of its name, so we make sure not to insert an extra one.

```

118     \string#1allowed only in math mode}\@ehd}

```

(End definition for \non@alpherr.)

\dorestore@version

```

119 \def\dorestore@version
120 {\ifmmode
121     \aftergroup\dorestore@version
122 \else
123     \gdef\init@restore@version{%
124         \global\let\init@restore@version\relax
125         \xdef\restore@mathversion
126             {\expandafter\noexpand\csname mv@\math@version\endcsname
127              \global\csname c@mv@\math@version\endcsname
128              \number\csname c@mv@\math@version\endcsname\relax}%
129         \aftergroup\dorestore@version
130     }%
131     \begingroup
132         \let\getanddefine@fonts@gobbletwo
133         \restore@mathversion
134     \endgroup
135     \fi}%
136 \@onlypreamble\dorestore@version

```

(End definition for \dorestore@version.)

`\document@select@group` We surround `\select@group` with braces so that functions using it can be used directly after `_` or `^`.

```

137 </2ekernel>
138 <latexrelease>\IncludeInRelease{2020/10/01}
139 <latexrelease> {\document@select@group}{\document@select@group}%
140 <*2ekernel | latexrelease>
141 \def\document@select@group#1#2#3#4{%
142   \ifx\math@bgroup\bgroup\else\relax\expandafter\@firstofone\fi
143   {%
144     \ifmmode
145       \ifnum\csname c@mv@\math@version\endcsname<\e@mathgroup@top
146         \begingroup
147           \escapechar\m@ne
148           \getanddefine@fonts{\csname c@mv@\math@version\endcsname}#3%
149           \globaldefs\@ne \math@fonts
150         \endgroup
151         \expandafter\extract@alph@from@version
152           \csname mv@\math@version\expandafter\endcsname
153           \expandafter{\number\csname
154             c@mv@\math@version\endcsname}%
155           #1%
156         \global\advance\csname c@mv@\math@version\endcsname\@ne
157       \else
158         \let#1\relax
159         \@latex@error{Too many math alphabets used
160           in version \math@version}%
161         \@eha
162       \fi

```

extra `\expandafter` to remove the `\expandafter` added below

```

163   \else \expandafter\expandafter\expandafter\non@alpherr\fi

```

If the legacy interface is used, e.g., `\sf -1$` the math alphabet `#1` does not take an argument so we better do not surround `#4` with braces, because then we get `{\relax}` into the formula and introduce an extra Ord atom. The two different cases can be distinguished by looking at the current value of `\math@bgroup`.

```

164   \expandafter#1\ifx\math@bgroup\bgroup{#4}\else#4\fi
165 }%
166 }
167 </2ekernel | latexrelease>
168 <latexrelease>\EndIncludeInRelease
169 <latexrelease>\IncludeInRelease{2015/01/01}
170 <latexrelease> {\document@select@group}{\document@select@group}%
171 <latexrelease>
172 <latexrelease>\def\document@select@group#1#2#3#4{%
173 <latexrelease> \ifx\math@bgroup\bgroup\else\relax\expandafter\@firstofone\fi
174 <latexrelease> {%
175 <latexrelease> \ifmmode
176 <latexrelease>   \ifnum\csname c@mv@\math@version\endcsname<\e@mathgroup@top
177 <latexrelease>     \begingroup
178 <latexrelease>       \escapechar\m@ne
179 <latexrelease>       \getanddefine@fonts{\csname c@mv@\math@version\endcsname}#3%
180 <latexrelease>       \globaldefs\@ne \math@fonts
181 <latexrelease>     \endgroup

```

```

182 <latexrelease> \expandafter\extract@alph@from@version
183 <latexrelease> \csname mv@\math@version\expandafter\endcsname
184 <latexrelease> \expandafter{\number\csname
185 <latexrelease> c@mv@\math@version\endcsname}%
186 <latexrelease> #1%
187 <latexrelease> \global\advance\csname c@mv@\math@version\endcsname\@ne
188 <latexrelease> \else
189 <latexrelease> \let#1\relax
190 <latexrelease> \@latex@error{Too many math alphabets used
191 <latexrelease> in version \math@version}%
192 <latexrelease> \@eha
193 <latexrelease> \fi
194 <latexrelease> \else \expandafter\non@alpherr\fi
195 <latexrelease> #1{#4}%
196 <latexrelease> }%
197 <latexrelease>}
198 <latexrelease>\EndIncludeInRelease
199 <latexrelease>
200 <latexrelease>\IncludeInRelease{0000/00/00}
201 <latexrelease> {\document@select@group}{\document@select@group}%
202 <latexrelease>
203 <latexrelease>\def\document@select@group#1#2#3#4{%
204 <latexrelease> \ifx\math@bgroup\bgroup\else\relax\expandafter\@firstofone\fi
205 <latexrelease> {%
206 <latexrelease> \ifmmode
207 <latexrelease> \ifnum\csname c@mv@\math@version\endcsname<\sixt@@n
208 <latexrelease> \begingroup
209 <latexrelease> \escapechar\m@ne
210 <latexrelease> \getanddefine@fonts
211 <latexrelease> {\csname c@mv@\math@version\endcsname}#3%
212 <latexrelease> \globaldefs\@ne \math@fonts
213 <latexrelease> \endgroup
214 <latexrelease> \expandafter\extract@alph@from@version
215 <latexrelease> \csname mv@\math@version\expandafter\endcsname
216 <latexrelease> \expandafter{\number\csname
217 <latexrelease> c@mv@\math@version\endcsname}%
218 <latexrelease> #1%
219 <latexrelease> \global\advance\csname c@mv@\math@version\endcsname\@ne
220 <latexrelease> \else
221 <latexrelease> \let#1\relax
222 <latexrelease> \@latex@error{Too many math alphabets used
223 <latexrelease> in version \math@version}%
224 <latexrelease> \@eha
225 <latexrelease> \fi
226 <latexrelease> \else \expandafter\non@alpherr\fi
227 <latexrelease> #1{#4}%
228 <latexrelease> }%
229 <latexrelease>}
230 <latexrelease>\EndIncludeInRelease
231 <*2kernel>

```

(End definition for \document@select@group.)

\process@table

```

232 \def\process@table{%
233   \def\cdp@elt##1##2##3##4{%
234     \@font@info{Checking defaults for
235       ##1/##2/##3/##4}%
236     \expandafter
237     \ifx\csname##1/##2/##3/##4\endcsname\relax

```

Grouping is important for two reasons, first \cdp@elt will get redefined if \Declare... functions are executed within the external .fd file and secondly \try@load@fontshape changes a lot of catcodes without surrounding itself with a group.

```

238     \begingroup
239     \def\f@encoding{##1}\def\f@family{##2}%
240     \try@load@fontshape
241     \endgroup
242   \fi
243   \expandafter
244   \ifx\csname##1/##2/##3/##4\endcsname\relax
245     \@latex@error{This NFSS system isn't set up properly}%
246     {For encoding scheme ##1 the defaults
247       ##2/##3/##4 do not form a valid font shape}%
248   \else
249     \@font@info{... okay}%
250   \fi}%
251 \cdp@list

```

Now we make sure that \error@fontshape is okay.

```

252 \begingroup
253   \escapechar\m@ne
254   \error@fontshape
255   \expandafter\ifx\csname \curr@fontshape\endcsname\relax
256     \begingroup
257     \try@load@fontshape
258     \endgroup
259   \fi
260   \expandafter\ifx\csname \curr@fontshape\endcsname\relax
261     \@latex@error{This NFSS system isn't set up properly}%
262     {The system maintainer forgot to specify a suitable
263       substitution
264       font shape using the \noexpand\DeclareErrorFont
265       command}%
266   \fi
267 \endgroup

```

Set \select@group to its meaning used within the document body.

```

268 \let\select@group\document@select@group

```

Install the default font attributes as they are currently pointing to error font face. We can speed up the process by just using \edef, thereby avoiding all kind of extra processing. Don't use \reset@font since that would trigger \selectfont.

```

269 \fontencoding\encodingdefault
270 \edef\f@family{\familydefault}%
271 \edef\f@series{\seriesdefault}%
272 \edef\f@shape{\shapedefault}%

```

Drop stuff not longer needed. We need to add many more!!!!

```

273 \everyjob{}%

```

```

274 }
275 \@onlypreamble\process@table

(End definition for \process@table.)

276 %\@onlypreamble\set@mathradical

```

\DeclareMathVersion

```

277 \def\DeclareMathVersion#1{%
278   \expandafter\new@mathversion\csname mv@#1\endcsname}
279 \@onlypreamble\DeclareMathVersion

(End definition for \DeclareMathVersion.)

```

\new@mathversion

```

280 \def\new@mathversion#1{%
281   \expandafter\in@\expandafter#1\expandafter{\version@list}%
282   \ifin@
283     \@font@info{Redeclaring math version
284               '\expandafter\@gobblefour\string#1'}%
285   \else
286     \expandafter\newcount\csname c@\expandafter
287                               \@gobble\string#1\endcsname
288     \def\version@elt{\noexpand\version@elt\noexpand}%
289     \edef\version@list{\version@list\version@elt#1}%
290   \fi

```

\toks@ is used to gather all tokens for the math version. \count@ will be used to count the math groups we add to this version.

```

291   \toks@{}%
292   \count@\z@

```

Now we loop over \group@list to add all math groups defined so far to the version and at the same time to count them.

```

293   \def\group@elt##1##2{%
294     \advance\count@\@ne
295     \addto@hook\toks@{\getanddefine@fonts##1##2}%
296     }%
297   \group@list

```

We set the counter for this math version to the number of math groups found in \group@list.

```

298   \global\csname c@\expandafter\@gobble\string#1\endcsname\count@

```

Now we loop over \alpha@list to add all math alphabets known so far. We have to distinguish the case that an alphabet by default should produce an error in new versions.

```

299   \def\alpha@elt##1##2##3{%
300     \ifx##2\no@alphabet@error
301       \toks@\expandafter{\the\toks@\install@mathalphabet##1%
302         {\no@alphabet@error##1}}%
303     \else
304       \toks@\expandafter{\the\toks@\install@mathalphabet##1%
305         {\select@group##1##2##3}}%
306     \fi
307     }%
308   \alpha@list

```

Finally we define the math version to expand to the contents of \toks@.

```

309 \xdef#1{\the\toks@}%
310 }
311 \@onlypreamble\new@mathversion

```

(End definition for \new@mathversion.)

\DeclareSymbolFont

```

312 \def\DeclareSymbolFont#1#2#3#4#5{%
313 \tempwafalse
314 \edef\reserved@b{#2}%
315 \def\cdp@elt##1##2##3##4{\def\reserved@c{##1}%
316 \ifx\reserved@b\reserved@c \@tempwattrue\fi}%
317 \cdp@list
318 \if@tempswa
319 \@ifundefined{sym#1}{%
320 \ifnum\count18<15 %
321 \expandafter\new@mathgroup\csname sym#1\endcsname
322 \expandafter\new@symbolfont\csname sym#1\endcsname
323 {#2}{#3}{#4}{#5}%
324 \else
325 \@latex@error{Too many symbol fonts declared}\@eha
326 \fi
327 }%
328 {%
329 \@font@info{Redeclaring symbol font ‘#1’}%

```

Update the group list.

```

330 \def\group@elt##1##2{%
331 \noexpand\group@elt\noexpand##1%
332 \expandafter\ifx\csname sym#1\endcsname##1%
333 \expandafter\noexpand\csname#2/#3/#4/#5\endcsname
334 \else
335 \noexpand##2%
336 \fi}%
337 \xdef\group@list{\group@list}%

```

Update the version list.

```

338 \def\version@elt##1{%
339 \expandafter
340 \SetSymbolFont@\expandafter##1\csname#2/#3/#4/#5\endcsname
341 \endcsname \csname sym#1\endcsname
342 }%
343 \version@list
344 }%
345 \else
346 \@latex@error{Encoding scheme ‘#2’ unknown}\@eha
347 \fi
348 }
349 \@onlypreamble\DeclareSymbolFont

```

(End definition for \DeclareSymbolFont.)

`\group@list`

```
350 \let\group@list\@empty
351 \@onlypreamble\group@list
```

(End definition for \group@list.)

`\group@elt`

```
352 \let\group@elt\relax
353 \@onlypreamble\group@elt
```

(End definition for \group@elt.)

`\new@symbolfont`

```
354 \def\new@symbolfont#1#2#3#4#5{%
355   \toks@\expandafter{\group@list}%
356   \edef\group@list{\the\toks@\noexpand\group@elt\noexpand#1%
357     \expandafter\noexpand\csname#2/#3/#4/#5\endcsname}%
358   \def\version@elt##1{\toks@\expandafter{##1}%
359     \edef##1{\the\toks@\noexpand\getanddefine@fonts
360       #1\expandafter\noexpand\csname#2/#3/#4/#5\endcsname}%
361     \global\advance\csname c@\expandafter
362       \@gobble\string##1\endcsname\@ne
363     }%
364   \version@list
365 }
366 \@onlypreamble\new@symbolfont
```

(End definition for \new@symbolfont.)

`\SetSymbolFont`

```
367 \def\SetSymbolFont#1#2#3#4#5#6{%
368   \@tempswafalse
369   \edef\reserved@b{#3}%
370   \def\cdp@elt##1##2##3##4{\def\reserved@c{##1}%
371     \ifx\reserved@b\reserved@c \@tempswatrue\fi}%
372   \cdp@list
373   \if@tempswa
374     \expandafter\SetSymbolFont@
375     \csname mv@#2\expandafter\endcsname\csname#3/#4/#5/#6\expandafter
376     \endcsname \csname sym#1\endcsname
377   \else
378     \@latex@error{Encoding scheme ‘#3’ unknown}\@eha
379   \fi
380 }
381 \@onlypreamble\SetSymbolFont
```

(End definition for \SetSymbolFont.)

`\SetSymbolFont@`

```
382 \def\SetSymbolFont@#1#2#3{%
383   \expandafter\in@\expandafter#1\expandafter{\version@list}%
384   \ifin@
385     \expandafter\in@\expandafter#3\expandafter{\group@list}%
386     \ifin@
387       \begingroup
```

```

388 \expandafter\get@cdp\string#2\@nil\reserved@a
389 \toks@{}%
390 \def\install@mathalphabet##1##2{%
391 \addto@hook\toks@{\install@mathalphabet##1{##2}}%
392 }%
393 \def\getanddefine@fonts##1##2{%
394 \ifnum##1=#3%
395 \addto@hook\toks@{\getanddefine@fonts#3#2}%
396 \expandafter\get@cdp\string##2\@nil\reserved@b
397 \ifx\reserved@a\reserved@b\else
398 \@font@info{Encoding ‘\reserved@b’ has changed
399 to ‘\reserved@a’ for symbol font\MessageBreak
400 ‘\expandafter\@gobblefour\string#3’ in the
401 math version ‘\expandafter
402 \@gobblefour\string#1’}%
403 \fi
404 \@font@info{%
405 Overwriting symbol font
406 ‘\expandafter\@gobblefour\string#3’ in
407 version ‘\expandafter
408 \@gobblefour\string#1’\MessageBreak
409 \@spaces \expandafter\@gobble\string##2 -->
410 \expandafter\@gobble\string#2}%
411 \else
412 \addto@hook\toks@{\getanddefine@fonts##1##2}%
413 \fi}%
414 #1%
415 \xdef#1{\the\toks@}%
416 \endgroup
417 \else
418 \@latex@error{Symbol font ‘\expandafter\@gobblefour\string#3’
419 not defined}\@eha
420 \fi
421 \else
422 \@latex@error{Math version ‘\expandafter\@gobblefour\string#1’
423 is not
424 defined}{You probably misspelled the name of the math
425 version.^^JOr you have to specify an additional package.}%
426 \fi
427 }
428 \@onlypreamble\SetSymbolFont@

```

(End definition for \SetSymbolFont@.)

\get@cdp

```

429 \def\get@cdp#1#2/#3\@nil#4{\def#4{#2}}
430 \@onlypreamble\get@cdp

```

(End definition for \get@cdp.)

\DeclareMathAlphabet

```

431 \def\DeclareMathAlphabet#1#2#3#4#5{%
432 \@tempswafalse
433 \edef\reserved@b{#2}%
434 \def\cdp@elt##1##2##3##4{\def\reserved@c{##1}%

```



```

435     \ifx\reserved@b\reserved@c \@tempswattrue\fi}%
436 \cdp@list
437 \if@tempswa
438   \expandafter\ifx
439   \csname\expandafter\@gobble\string#1\endcsname
440   \relax
441   \new@mathalphabet#1{#2}{#3}{#4}{#5}%
442 \else

```

Check if it is already a math alphabet.

```

443   \edef\reserved@a{\noexpand\in@{\string\select@group}%
444     {\expandafter\meaning\csname \expandafter
445       \@gobble\string#1\space\endcsname}}%
446 \reserved@a
447 \ifin@
448   \@font@info{Redeclaring math alphabet \string#1}%
449   \def\version@elt##1{%
450     \expandafter\SetMathAlphabet@\expandafter
451     ##1\csname#2/#3/#4/#5\expandafter\endcsname

452     \csname M@#2\expandafter\endcsname
453     \csname \expandafter\@gobble\string#1\space\endcsname#1}%
454   \version@list
455 \else

```

Check if it is a math alphabet defined via \DeclareSymbolFontAlphabet.

```

456   \edef\reserved@a{\noexpand\in@{\string\use@mathgroup}%
457     {\expandafter\meaning\csname \expandafter
458       \@gobble\string#1\space\endcsname}}%
459 \reserved@a
460 \ifin@

```

In that case overwriting is simple since there is nothing inserted in the math version macros.

```

461   \@font@info{Redeclaring math alphabet \string#1}%
462   \new@mathalphabet#1{#2}{#3}{#4}{#5}%

```

Otherwise panic.

```

463   \else
464     \@latex@error{Command ‘\string#1’ already defined}\@eha
465   \fi
466 \fi
467 \fi
468 \else
469   \@latex@error{Encoding scheme ‘#2’ unknown}\@eha
470 \fi
471 }
472 \@onlypreamble\DeclareMathAlphabet

```

(End definition for \DeclareMathAlphabet.)

\new@mathalphabet

```

473 \def\new@mathalphabet#1#2#3#4#5{%
474   \toks@\expandafter{\alpha@list}%
475   \edef#1{\expandafter\noexpand\csname \expandafter
476     \@gobble\string#1\space\endcsname

```

```

477         \if/#5/%
478             \noexpand\no@alphabet@error
479             \noexpand\no@alphabet@error
480         \else
481             \expandafter\noexpand\csname M@#2\endcsname
482             \expandafter\noexpand\csname#2/#3/#4/#5\endcsname
483         \fi
484     }%
485     \toks2\expandafter{#1}%
486     \edef\alpha@list{\the\toks@\noexpand\alpha@elt\the\toks2}%
487     \def\version@elt##1{\toks@\expandafter{##1}%
488         \edef##1{\the\toks@\install@mathalphabet
489             \expandafter\noexpand
490             \csname \expandafter@gobble
491                 \string#1\space\endcsname
492             {\if/#5/%
493                 \noexpand\no@alphabet@error
494                 \noexpand#1%
495             \else
496                 \noexpand\select@group\the\toks2
497                 \fi}}%
498     }%
499     \version@list
500     \expandafter\edef\csname \expandafter@gobble
501         \string#1\space\endcsname{\if/#5/%
502             \noexpand\no@alphabet@error
503             \noexpand#1%
504         \else
505             \noexpand\select@group\the\toks2
506         \fi}%
507     \edef#1{\noexpand\protect
508         \expandafter\noexpand\csname \expandafter
509             \@gobble\string#1\space\endcsname}%
510 }
511 \@onlypreamble\new@mathalphabet

```

(End definition for \new@mathalphabet.)

\SetMathAlphabet

```

512 \def\SetMathAlphabet#1#2#3#4#5#6{%
513     \@tempswafalse
514     \edef\reserved@b{#3}%
515     \def\cdp@elt##1##2##3##4{\def\reserved@c{##1}%
516         \ifx\reserved@b\reserved@c \@tempswatrue\fi}%
517     \cdp@list
518     \if@tempswa
519         \expandafter\SetMathAlphabet@
520             \csname mv@#2\expandafter\endcsname\csname#3/#4/#5/#6\expandafter
521                 \endcsname \csname M@#3\expandafter\endcsname
522                 \csname \expandafter@gobble\string#1\space\endcsname#1%
523     \else
524         \@latex@error{Encoding scheme ‘#3’ unknown}\@eha
525     \fi
526 }
527 \@onlypreamble\SetMathAlphabet

```

(End definition for \SetMathAlphabet.)

\SetMathAlphabet@

```

528 \def\SetMathAlphabet@#1#2#3#4#5{%
529   \expandafter\in@\expandafter#1\expandafter{\version@list}%
530   \ifin@
531     \expandafter\in@\expandafter#4\expandafter{\alpha@list}%
532     \ifin@
533       \begingroup
534         \toks@{}%
535         \def\getanddefine@fonts##1##2{%
536           \addto@hook\toks@{\getanddefine@fonts##1##2}%
537         }%
538         \def\reserved@c##1##2##3##4{%           % for message below
539           \expandafter\@gobble\string##4}%
540         \def\install@mathalphabet##1##2{%
541           \ifx##1#4%
542             \addto@hook\toks@
543               {\install@mathalphabet#4{\select@group#4#3#2}}%
544             \@font@info{Overwriting math alphabet
545               '\string#5' in version '\expandafter
546               \@gobblefour\string#1'\MessageBreak
547               \@spaces \reserved@c##2 -->
548               \expandafter\@gobble\string#2}%
549           \else
550             \addto@hook\toks@{\install@mathalphabet##1{##2}}%
551           \fi
552         }%
553         #1%
554         \xdef#1{\the\toks@}%
555       \endgroup
556     \else

```

If the math alphabet was defined via \DeclareSymbolFontAlphabet we have remove its external definition and add it as a normal math alphabet to every version before trying to change it in one version.

```

557   \edef\reserved@a{%
558     \noexpand\in@{\string\use@mathgroup}{\meaning#4}}%
559   \reserved@a
560   \ifin@
561     \def\reserved@b##1\use@mathgroup##2##3{%
562       \def\reserved@b{##3}\def\reserved@c{##2}}%
563     \expandafter\reserved@b#4%
564     \begingroup
565       \def\install@mathalphabet##1##2{%
566         \addto@hook\toks@{\install@mathalphabet##1{##2}}%
567       }%
568       \def\getanddefine@fonts##1##2{%
569         \addto@hook\toks@{\getanddefine@fonts##1##2}%
570         \ifnum##1=\reserved@b
571           \expandafter
572             \addto@hook\expandafter\toks@
573             \expandafter{\expandafter\install@mathalphabet
574             \expandafter#4\expandafter

```

```

575         {\expandafter\select@group\expandafter
576          #4\reserved@c##2}}}%
577     \fi
578   }%
579   \def\version@elt##1{%
580     \toks@{}%
581     ##1%
582     \xdef##1{\the\toks@}%
583   }%
584   \version@list
585 \endgroup

```

Put it into the `\alpha@list` with default ‘error’

```

586   \expandafter\gdef\expandafter\alpha@list\expandafter
587   {\alpha@list
588     \alpha@elt #4\no@alphabet@error \no@alphabet@error}%
589   \gdef#4{\no@alphabet@error #5}% fake things :-)

```

Then call the internal setting routine again:

```

590     \SetMathAlphabet@{#1}{#2}{#3}#4#5%
591   \else
592     \@latex@error{Command ‘\string#5’ not defined as a
593       math alphabet}%
594     {Use \noexpand\DeclareMathAlphabet to define it.}%
595   \fi
596 \fi
597 \else
598   \@latex@error{Math version ‘\expandafter\@gobblefour\string#1’
599     is not
600     defined}{You probably misspelled the name of the math
601     version.^^JOr you have to specify an additional package.}%
602 \fi
603 }
604 \onlypreamble\SetMathAlphabet@

```

(End definition for `\SetMathAlphabet@`.)

`\DeclareMathAccent` Could do with more checks like allowing single number in #4 lowercase in #4 etc

```

605 </2ekernel>
606 <*2ekernel | latexrelease>
607 <latexrelease>\IncludeInRelease{2019/10/01}%
608 <latexrelease>          {DeclareMathAccent}{Make math accents robust}%
609 \def\DeclareMathAccent#1#2#3#4{%
610   \expandafter\in@\csname sym#3\expandafter\endcsname
611   \expandafter{\group@list}%
612   \ifin@
613     \begingroup
614     \count\z@=#4\relax
615     \count\tw@\count\z@
616     \divide\count\z@\sist@@n
617     \count@\count\z@
618     \multiply\count@\sist@@n
619     \advance\count\tw@-\count@
620     \if\relax\noexpand#1% is command?
621     \edef\reserved@a{\noexpand\in@

```

```

622         {\expandafter\@gobble\string\mathaccent}
623         {\expandafter\meaning
624         \csname\expandafter\@gobble\string#1\space\endcsname}}%
625     \reserved@a
626     \ifin@
627         \expandafter\let
628         \csname\expandafter\@gobble\string#1\space\endcsname
629         \@undefined
630         \expandafter\set@mathaccent
631         \csname sym#3\endcsname#1#2%
632         {\hexnumber@{\count\z@}\hexnumber@{\count\tw@}}%
633         \@font@info{Redefining math accent \string#1}%
634     \else
635         \expandafter\ifx
636         \csname\expandafter\@gobble\string#1\endcsname
637         \relax
638         \expandafter\set@mathaccent
639         \csname sym#3\endcsname#1#2%
640         {\hexnumber@{\count\z@}\hexnumber@{\count\tw@}}%
641     \else
642         \@latex@error{Command '\string#1' already defined}\@eha
643     \fi
644 \fi
645 \else
646 \@latex@error{Not a command name: '\noexpand#1'}\@eha
647 \fi
648 \endgroup
649 \else
650 \@latex@error{Symbol font '#3' is not defined}\@eha
651 \fi
652 }
653 </2ekernel | latexrelease>
654 <latexrelease>\EndIncludeInRelease
655 <latexrelease>\IncludeInRelease{0000/00/00}%
656 <latexrelease>          {DeclareMathAccent}{Make math accents robust}%
657 <latexrelease>\def\DeclareMathAccent#1#2#3#4{%
658 <latexrelease>  \expandafter\in@ \csname sym#3\endcsname\expandafter\endcsname
659 <latexrelease>  \expandafter{\group@list}%
660 <latexrelease>  \ifin@
661 <latexrelease>    \begingroup
662 <latexrelease>    \count\z@=#4\relax
663 <latexrelease>    \count\tw@\count\z@
664 <latexrelease>    \divide\count\z@\sixt@@n
665 <latexrelease>    \count@\count\z@
666 <latexrelease>    \multiply\count@\sixt@@n
667 <latexrelease>    \advance\count\tw@-\count@
668 <latexrelease>    \if\relax\noexpand#1% is command?
669 <latexrelease>      \edef\reserved@a{\noexpand\in@
670 <latexrelease>        {\expandafter\@gobble\string\mathaccent}\{meaning#1}}%
671 <latexrelease>      \reserved@a
672 <latexrelease>    \ifin@
673 <latexrelease>      \expandafter\set@mathaccent
674 <latexrelease>      \csname sym#3\endcsname#1#2%
675 <latexrelease>      {\hexnumber@{\count\z@}\hexnumber@{\count\tw@}}%

```

```

676 <latexrelease>      \@font@info{Redeclaring math accent \string#1}%
677 <latexrelease>      \else
678 <latexrelease>      \expandafter\ifx
679 <latexrelease>      \csname\expandafter\@gobble\string#1\endcsname
680 <latexrelease>      \relax
681 <latexrelease>      \expandafter\set@mathaccent
682 <latexrelease>      \csname sym#3\endcsname#1#2%
683 <latexrelease>      {\hexnumber@{\count\z@}\hexnumber@{\count\tw@}}%
684 <latexrelease>      \else
685 <latexrelease>      \@latex@error{Command ‘\string#1’ already defined}\@eha
686 <latexrelease>      \fi
687 <latexrelease>      \fi
688 <latexrelease>      \else
689 <latexrelease>      \@latex@error{Not a command name: ‘\noexpand#1’}\@eha
690 <latexrelease>      \fi
691 <latexrelease>      \endgroup
692 <latexrelease>      \else
693 <latexrelease>      \@latex@error{Symbol font ‘#3’ is not defined}\@eha
694 <latexrelease>      \fi
695 <latexrelease> }
696 <latexrelease> \EndIncludeInRelease
697 <*2ekernel>

698 \@onlypreamble\DeclareMathAccent

(End definition for \DeclareMathAccent.)

```

`\set@mathaccent`

```

699 </2ekernel>
700 <*2ekernel | latexrelease>
701 <latexrelease> \IncludeInRelease{2019/10/01}%
702 <latexrelease>      {\set@mathaccent}{makemath accents robust}%
703 \def\set@mathaccent#1#2#3#4{%
704   \xdef#2{\mathaccent"\mathchar@type#3\hexnumber@#1#4\relax}%
705   \MakeRobust#2%
706 }
707 \@onlypreamble\set@mathaccent
708 </2ekernel | latexrelease>
709 <latexrelease> \EndIncludeInRelease
710 <latexrelease> \IncludeInRelease{0000/00/00}%
711 <latexrelease>      {\set@mathaccent}{makemath accents robust}%
712 <latexrelease>
713 <latexrelease> \def\set@mathaccent#1#2#3#4{%
714 <latexrelease>   \xdef#2{\mathaccent"\mathchar@type#3\hexnumber@#1#4\relax}}
715 <latexrelease>
716 <latexrelease> \EndIncludeInRelease
717 <*2ekernel>

(End definition for \set@mathaccent.)

```

`\DeclareMathSymbol`

```

718 \def\DeclareMathSymbol#1#2#3#4{%
719   \expandafter\in@\csname sym#3\expandafter\endcsname
720   \expandafter{\group@list}%
721   \ifin@

```

```

722 \begingroup
723 \count\z@=#4\relax
724 \count\tw@\count\z@
725 \divide\count\z@\sist@@n
726 \count@\count\z@
727 \multiply\count@\sist@@n
728 \advance\count\tw@-\count@
729 \if\relax\noexpand#1% is command?

```

Store the command name with a space attached inside `\reserved@@b` in case we look at a robust definition.

```

730 \edef\reserved@b{\expandafter\noexpand
731 \csname\expandafter@gobble\string#1\space\endcsname}%

```

Test both `#1` and `#1_` for containing `mathchar`.

```

732 \edef\reserved@a
733 {\noexpand\in@{\expandafter@gobble\string\mathchar}%
734 {\meaning#1\expandafter\meaning\reserved@b}}%
735 \reserved@a

```

Drop `#1_` in case it was defined before.

```

736 \global\expandafter\let\reserved@b\@undefined
737 \ifin@
738 \expandafter\set@mathsymbol
739 \csname sym#3\endcsname#1#2%
740 {\hexnumber@{\count\z@}\hexnumber@{\count\tw@}}%
741 \@font@info{Redeclaring math symbol \string#1}%
742 \else
743 \expandafter\ifx
744 \csname\expandafter@gobble\string#1\endcsname
745 \relax
746 \expandafter\set@mathsymbol
747 \csname sym#3\endcsname#1#2%
748 {\hexnumber@{\count\z@}\hexnumber@{\count\tw@}}%
749 \else
750 \@latex@error{Command '\string#1' already defined}\@eha
751 \fi
752 \fi
753 \else
754 \expandafter\set@mathchar
755 \csname sym#3\endcsname#1#2
756 {\hexnumber@{\count\z@}\hexnumber@{\count\tw@}}%
757 \fi
758 \endgroup
759 \else
760 \@latex@error{Symbol font '#3' is not defined}\@eha
761 \fi
762 }
763 \@onlypreamble\DeclareMathSymbol

```

(End definition for `\DeclareMathSymbol`.)

`\set@mathchar`

```

764 \def\set@mathchar#1#2#3#4{%
765 \global\mathcode'#2="\mathchar@type#3\hexnumber@#1#4\relax}
766 \@onlypreamble\set@mathchar

```

(End definition for \set@mathchar.)

\set@mathsymbol

```
767 \def\set@mathsymbol#1#2#3#4{%
768   \global\mathchardef#2"\mathchar@type#3\hexnumber@#1#4\relax}
769 \@onlypreamble\set@mathsymbol
```

(End definition for \set@mathsymbol.)

```
770 %\def\mathsymbol#1#2#3{%
771 %   \@tempcnta=#3\relax
772 %   \@tempcntb\@tempcnta
773 %   \divide\@tempcnta\sixt@@n
774 %   \count@\@tempcnta
775 %   \multiply\count@\sixt@@n
776 %   \advance\@tempcntb-\count@
777 %   \mathchar"\mathchar@type#1\hexnumber@#2%
778 %       \hexnumber@\@tempcnta\hexnumber@\@tempcntb\relax}
779 %
780 %\def\DeclareMathAlphabetCharacter#1#2#3{%
781 %   \DeclareMathSymbol{#1}7{#2}{#3}
```

\DeclareMathDelimiter

```
782 \def\DeclareMathDelimiter#1{%
783   \if\relax\noexpand#1%
784     \expandafter\@DeclareMathDelimiter
785   \else
786     \expandafter\@xxDeclareMathDelimiter
787   \fi
788   #1}
789 \@onlypreamble\DeclareMathDelimiter
```

(End definition for \DeclareMathDelimiter.)

\@xxDeclareMathDelimiter

This macro checks if the second arg is a “math type” such as `\mathopen`. The undocumented original code didn’t use math types when the delimiter was a single letter. For this reason the coding is a bit strange as it tries to support the undocumented syntax for compatibility reasons.

```
790 \def\@xxDeclareMathDelimiter#1#2#3#4{%
```

7 is the default value returned in the case that `\mathchar@type` is passed something unexpected, like a math symbol font name. We locally move `\mathalpha` out of the way so if you use that the right branch is taken. This will still fail if an explicit number 7 is used!

```
791 \begingroup
792   \let\mathalpha\mathord
793   \ifnum7=\mathchar@type{#2}%
794     \endgroup
```

If this branch is taken we have old syntax (5 arguments).

```
795     \expandafter\@firstofone
796   \else
```


If this branch is taken `\mathchar@type` is different from 7 so we assume new syntax. In this case we also use the arguments to set up the letter as a math symbol for the case where it is not used as a delimiter.

```
797     \endgroup
798     \DeclareMathSymbol#1{#2}{#3}{#4}%
```

Then we arrange that `\@xDeclareMathDelimiter` only gets #1, #3, #4 ... as it does not expect a math type as argument.

```
799     \expandafter\@firstoftwo
800     \fi
801     {\@xDeclareMathDelimiter#1}{#2}{#3}{#4}}
802 \@onlypreamble\@xxDeclareMathDelimiter
```

(End definition for `\@xxDeclareMathDelimiter`.)

`\@DeclareMathDelimiter`

```
803 \def\@DeclareMathDelimiter#1#2#3#4#5#6{%
804   \expandafter\in@\csname sym#3\expandafter\endcsname
805   \expandafter{\group@list}%
806   \ifin@
807     \expandafter\in@\csname sym#5\expandafter\endcsname
808     \expandafter{\group@list}%
809   \ifin@
810     \begingroup
811       \count\z@=#4\relax
812       \count\tw@\count\z@
813       \divide\count\z@\sist@@n
814       \count@\count\z@
815       \multiply\count@\sist@@n
816       \advance\count\tw@-\count@
817       \edef\reserved@c{\hexnumber@{\count\z@}\hexnumber@{\count\tw@}}%
818     %
819     \count\z@=#6\relax
820     \count\tw@\count\z@
821     \divide\count\z@\sist@@n
822     \count@\count\z@
823     \multiply\count@\sist@@n
824     \advance\count\tw@-\count@
825     \edef\reserved@d{\hexnumber@{\count\z@}\hexnumber@{\count\tw@}}%
826   %
827   \edef\reserved@a{\noexpand\in@
828     {\expandafter\@gobble\string\delimiter}{\meaning#1}}%
829   \reserved@a
830   \ifin@
831     \expandafter\set@mathdelimiter
832     \csname sym#3\expandafter\endcsname
833     \csname sym#5\endcsname#1#2%
834     \reserved@c\reserved@d
835     \@font@info{Redeclaring math delimiter \string#1}%
836   \else
837     \expandafter\ifx
838     \csname\expandafter\@gobble\string#1\endcsname
839     \relax
840     \expandafter\set@mathdelimiter
```

```

841         \csname sym#3\expandafter\endcsname
842         \csname sym#5\endcsname#1#2%
843         \reserved@c\reserved@d
844     \else
845         \@latex@error{Command ‘\string#1’ already defined}\@eha
846     \fi
847 \fi
848 \endgroup
849 \else
850     \@latex@error{Symbol font ‘#5’ is not defined}\@eha
851 \fi
852 \else
853     \@latex@error{Symbol font ‘#3’ is not defined}\@eha
854 \fi
855 }
856 \@onlypreamble\@DeclareMathDelimiter

```

(End definition for \@DeclareMathDelimiter.)

\@xDeclareMathDelimiter

```

857 \def\@xDeclareMathDelimiter#1#2#3#4#5{%
858     \expandafter\in@\csname sym#2\expandafter\endcsname
859     \expandafter{\group@list}%
860     \ifin@
861         \expandafter\in@\csname sym#4\expandafter\endcsname
862         \expandafter{\group@list}%
863     \ifin@
864         \begingroup
865             \count\z@=#3\relax
866             \count\tw@\count\z@
867             \divide\count\z@\sixt@@n
868             \count@\count\z@
869             \multiply\count@\sixt@@n
870             \advance\count\tw@-\count@
871             \edef\reserved@c{\hexnumber@{\count\z@}\hexnumber@{\count\tw@}}%
872             %
873             \count\z@=#5\relax
874             \count\tw@\count\z@
875             \divide\count\z@\sixt@@n
876             \count@\count\z@
877             \multiply\count@\sixt@@n
878             \advance\count\tw@-\count@
879             \edef\reserved@d{\hexnumber@{\count\z@}\hexnumber@{\count\tw@}}%
880             \expandafter\set@@mathdelimiter
881             \csname sym#2\expandafter\endcsname\csname sym#4\endcsname#1%
882             \reserved@c\reserved@d
883         \endgroup
884     \else
885         \@latex@error{Symbol font ‘#4’ is not defined}\@eha
886     \fi
887 \else
888     \@latex@error{Symbol font ‘#2’ is not defined}\@eha
889 \fi
890 }

```

```
891 \@onlypreamble\@xDeclareMathDelimiter
```

(End definition for \@xDeclareMathDelimiter.)

`\set@mathdelimiter` We have to end the definition of a math delimiter like `\lfloor` with a space and not with `\relax` as we did before, because otherwise constructs involving `\abovewithdelims` will prematurely end (pr/1329)

```
892 </2ekernel>
893 <*2ekernel | latexrelease>
894 <latexrelease>\IncludeInRelease{2019/10/01}%
895 <latexrelease>          {\set@mathdelimiter}{make delimiters robust}%
896 \def\set@mathdelimiter#1#2#3#4#5#6{%
```

We use `\protected` not `\MakeRobust` so that `\bigl\lfloor` etc. works inside the argument of `\protected@edef`.

```
897   \protected
898   \xdef#3{\delimiter"\mathchar@type#4\hexnumber@#1#5%
899     \hexnumber@#2#6 }%
900 %   \MakeRobust#3%
901 }
902 \@onlypreamble\set@mathdelimiter
903 </2ekernel | latexrelease>
904 <latexrelease>\EndIncludeInRelease
905 <latexrelease>\IncludeInRelease{0000/00/00}%
906 <latexrelease>          {\set@mathdelimiter}{make delimiters robust}%
907 <latexrelease>
908 <latexrelease>\def\set@mathdelimiter#1#2#3#4#5#6{%
909 <latexrelease>   \xdef#3{\delimiter"\mathchar@type#4\hexnumber@#1#5%
910 <latexrelease>     \hexnumber@#2#6 }}
911 <latexrelease>
912 <latexrelease>\EndIncludeInRelease
913 <*2ekernel>
```

(End definition for `\set@mathdelimiter`.)

`\set@@mathdelimiter`

```
914 \def\set@@mathdelimiter#1#2#3#4#5{%
915   \global\delcode'#3="\hexnumber@#1#4\hexnumber@#2#5\relax}
916 \@onlypreamble\set@@mathdelimiter
```

(End definition for `\set@@mathdelimiter`.)

`\DeclareMathRadical`

```
917 \def\DeclareMathRadical#1#2#3#4#5{%
```

Below is a crude fix to make this macro work if `#1` is undefined or `\relax`. Should be improved!

```
918   \expandafter\ifx
919     \csname\expandafter\@gobble\string#1\endcsname
920     \relax
921     \let#1\radical
922   \fi
923   \edef\reserved@a{\noexpand\in@
924     {\expandafter\@gobble\string\radical}{\meaning#1}}%
925   \reserved@a
```

```

926 \ifin@
927 \expandafter\in@\csname sym#2\expandafter\endcsname
928 \expandafter{\group@list}%
929 \ifin@
930 \expandafter\in@\csname sym#4\expandafter\endcsname
931 \expandafter{\group@list}%
932 \ifin@
933 \begingroup
934 \count\z@=#3\relax
935 \count\tw@\count\z@
936 \divide\count\z@\sist@@n
937 \count@\count\z@
938 \multiply\count@\sist@@n
939 \advance\count\tw@-\count@
940 \edef\reserved@c{%
941 \hexnumber@\count\z@}\hexnumber@\count\tw@}%
942 \count\z@=#5\relax
943 \count\tw@\count\z@
944 \divide\count\z@\sist@@n
945 \count@\count\z@
946 \multiply\count@\sist@@n
947 \advance\count\tw@-\count@
948 \edef\reserved@d{%
949 \hexnumber@\count\z@}\hexnumber@\count\tw@}%
Coded inline instead of using \set@mathradical
950 % \expandafter\set@mathradical
951 % \csname sym#2\expandafter\endcsname
952 % \csname sym#4\endcsname#1%
953 % \reserved@c\reserved@d
954 \xdef#1{\radical"\expandafter\hexnumber@
955 \csname sym#2\endcsname\reserved@c
956 \expandafter\hexnumber@
957 \csname sym#4\endcsname\reserved@d
958 \relax}%
959 \endgroup
960 \else
961 \@latex@error{Symbol font ‘#4’ is not defined}\@eha
962 \fi
963 \else
964 \@latex@error{Symbol font ‘#2’ is not defined}\@eha
965 \fi
966 \else
967 \@latex@error{Command ‘\string#1’ already defined}\@eha
968 \fi
969 }
970 \@onlypreamble\DeclareMathRadical

```

(End definition for \DeclareMathRadical.)

Definition below was wrong it contained \delimiter !

```

def\set@mathradical#1#2#3#4#5{%
\xdef#3{\radical"\hexnumber@#1#4\hexnumber@#2#5\relax}}

```

\mathalpha just a dummy currently

```

971 \let\mathalpha\relax
(End definition for \mathalpha.)

```

\mathchar@type

```

972 \def\mathchar@type#1{%
973   \ifodd 2#1#1 #1\else           % is this non-negative number?
974   \ifx#1\mathord 0\else
975   \ifx#1\mathop 1\else
976   \ifx#1\mathbin 2\else
977   \ifx#1\mathrel 3\else
978   \ifx#1\mathopen 4\else
979   \ifx#1\mathclose 5\else
980   \ifx#1\mathpunct 6\else
981   7%                             % anything else is variable ord
982   \fi
983   \fi
984   \fi
985   \fi
986   \fi
987   \fi
988   \fi
989 \fi}
990 \@onlypreamble\mathchar@type
(End definition for \mathchar@type.)

```

\DeclareSymbolFontAlphabet

```

991 \def\DeclareSymbolFontAlphabet#1#2{%
992   \expandafter\DeclareSymbolFontAlphabet@
993   \csname \expandafter\@gobble\string#1\space\endcsname{#2}#1}
994 \@onlypreamble\DeclareSymbolFontAlphabet
(End definition for \DeclareSymbolFontAlphabet.)

```

\DeclareSymbolFontAlphabet@

```

995 \def\DeclareSymbolFontAlphabet@#1#2#3{%
We use the switch \if@tempswa to decide if we can declare this symbol font alphabet.
996   \@tempswatrue
First check if #2 is known to be a symbol font
997   \expandafter\in@\csname sym#2\expandafter\endcsname
998   \expandafter{\group@list}%
999   \ifin@
Check if #1 is defined as a math alphabet defined via \DeclareMathAlphabet:
1000   \expandafter\in@\expandafter#1\expandafter{\alpha@list}%
1001   \ifin@
If so remove it from the \alpha@list and from all math version macros.
1002   \@font@info{Redeclaring math alphabet \string#3}%
1003   \toks@{}%
1004   \def\alpha@elt##1##2##3{%
1005     \ifx##1#1\else\addto@hook\toks@{\alpha@elt##1##2##3}\fi}%
1006   \alpha@list
1007   \xdef\alpha@list{\the\toks@}%

```

Now we loop over all versions and remove the math alphabet:

```

1008     \def\version@elt##1{%
1009         \begingroup
1010         \toks@{}%
1011         \def\getanddefine@fonts####1####2{%
1012             \addto@hook\toks@{\getanddefine@fonts####1####2}}%
1013         \def\install@mathalphabet####1####2{%
1014             \ifx####1#1\else
1015                 \addto@hook\toks@{\install@mathalphabet
1016                                     ####1{####2}}\fi}%
1017         ##1%
1018         \xdef##1{\the\toks@}%
1019     \endgroup
1020 }%
1021 \version@list
1022 \else

```

If #3 is not defined as a math alphabet check if it is defined at all:

```

1023     \expandafter\ifx
1024     \csname\expandafter\@gobble\string#1\space\endcsname
1025     \relax

```

If it is undefined, fine otherwise check if it is a math alphabet defined via `\DeclareSymbolFontAlphabet`:

```

1026     \else
1027         \edef\reserved@a{%
1028             \noexpand\in@{\string\use@mathgroup}{\meaning#1}}%
1029         \reserved@a
1030         \ifin@
1031             \@font@info{Redeclaring math alphabet \string#3}%
1032         \else

```

Since the command #3 is defined to be something which is not a math alphabet we have to skip redefining it.

```

1033         \@tempswafalse
1034         \@latex@error{Command ‘\string#3’ already defined}\@eha
1035     \fi
1036 \fi
1037 \fi
1038 \else

```

Since the symbol font is not known we better skip defining this alphabet.

```

1039     \@tempswafalse
1040     \@latex@error{Unknown symbol font ‘#2’}\@eha
1041 \fi
1042 \if@tempswa

```

When we reach this point we are allowed to define #1 to be a symbol font math alphabet. This means that we have to set it to

`\use@mathgroup <math-settings> \sym<name>`

The `<math-settings>` are the one for the encoding that is used in the font shape where `\sym<name>` is pointing to. This means that we have to get it from the information stored in `\group@list`. Thus we loop through that list after defining `\group@elt` in a suitable way.

```

1043 \def\group@elt##1##2{%
1044     \expandafter\ifx\csname sym#2\endcsname##1%
1045     \expandafter\reserved@a\string##2\@nil
1046     \fi}%
1047 \def\reserved@a##1##2/##3\@nil{%
1048     \def\reserved@a{##2}}%
1049 \group@list
1050 \toks@{\relax\ifmmode \else \non@alpherr#1\fi}%
1051 \edef#1{\the\toks@
1052     \noexpand\use@mathgroup
1053     \expandafter\noexpand\csname M@\reserved@a\endcsname
1054     \csname sym#2\endcsname}%
1055 \def#3{\protect#1}%
1056 \fi
1057 }
1058 \@onlypreamble\DeclareSymbolFontAlphabet@
1059 </2ekernel>

```

(End definition for \DeclareSymbolFontAlphabet@.)

File z

ltfssini.dtx

This file contains the top level L^AT_EX interface to the font selection scheme commands. See other parts of the L^AT_EX distribution, or *The L^AT_EX Companion* for higher level documentation of these commands.

1 NFSS Initialization

Finally, there are six commands that are to be used in L^AT_EX and that we will therefore protect against expansion at the wrong point: `\fontfamily`, `\fontseries`, `\fontshape`, `\fontsize`, `\selectfont`, and `\mathversion`.

```
1 \langle*2ekernel\rangle
```

1.1 Providing math *versions*

L^AT_EX provides two *versions*. We call them *normal* and *bold*, respectively.

```
2 \DeclareMathVersion{normal}
3 \DeclareMathVersion{bold}
```

Now we define the standard font change commands. We don't allow the use of `\rmfamily` etc. in math mode.

(Actually most are now defined further down in the file.)

First the changes to another *family*:

```
4 %\DeclareRobustCommand\rmfamily
5 %      {\not@math@alphabet\rmfamily\mathrm
6 %      \fontfamily\rmdefault\selectfont}
7 %\DeclareRobustCommand\sffamily
8 %      {\not@math@alphabet\sffamily\mathsf
9 %      \fontfamily\sfddefault\selectfont}
10 %\DeclareRobustCommand\ttfamily
11 %      {\not@math@alphabet\ttfamily\mathtt
12 %      \fontfamily\ttdefault\selectfont}
```

Then the commands changing the *series*:

```
13 %\DeclareRobustCommand\bfseries
14 %      {\not@math@alphabet\bfseries\mathbf
15 %      \fontseries\bfdefault\selectfont}
16 %\DeclareRobustCommand\mdseries
17 %      {\not@math@alphabet\mdseries\relax
18 %      \fontseries\mddefault\selectfont}
19 \DeclareRobustCommand\upshape
20      {\not@math@alphabet\upshape\relax
21      \fontshape\updefault\selectfont}
```

Then the commands changing the *shape*:

```
22 \DeclareRobustCommand\slshape
23      {\not@math@alphabet\slshape\relax
24      \fontshape\sldefault\selectfont}
25 \DeclareRobustCommand\scshape
26      {\not@math@alphabet\scshape\relax
```



```

27         \fontshape\scdefault\selectfont}
28 \DeclareRobustCommand\itshape
29     {\not@math@alphabet\itshape\mathit
30     \fontshape\itdefault\selectfont}

```

2 Custom series settings for main document families

This section was introduced 2020/02/02 and for now we support a full rollback (may need splitting later).

```

31 </2ekernel>
32 <*2ekernel | latexrelease>
33 <latexrelease>\IncludeInRelease{2020/02/02}%
34 <latexrelease>          {\DeclareFontSeriesDefault}{Custom series}%

```

One problem with the NFSS approach of handling the series axis turned out to be that (especially with respect to “boldness”) different font families implemented different strategies. For example, with Computer Modern fonts you normally only have **bx** whereas most PostScript fonts offered only **b** but not **bx**. As a result L^AT_EX’s standard setting for `\bfdefault` didn’t work with such fonts, but if it got changed to produce **b**, then that didn’t work with Computer Modern if the fonts got combined (e.g., using Computer Modern Typewriter with such fonts).

The solution back then was to provide substitution rules in the font `.fd` such that if a **bx** series got requested the **b** series got used. While this works in that particular case, it isn’t a very general solution. For example, if you happen to have a font family that has several weights you may want to typeset the whole document in a somewhat lighter or darker font but if you then modify `\mddefault` to allow for this, then of course your change only works with that particular family but not with the typewriter or sans serif family you also want to use.

A better solution was provided by the `mweights` package by Bob Tennent that offers defaults on the level of the three main font families in the document: for “rm”, “sf” and “tt” so that font packages could define defaults for the sans serif document font by providing `\bfseries@sf` which then was used when `\bfseries` got executed and the current family was the `\sffamily`.

`\DeclareFontSeriesDefault`

We now support this concept directly from within L^AT_EX and for use in font packages (or the document preamble) we offer `\DeclareFontSeriesDefault`. This declaration takes three arguments:

document family interface: Can either be `rm`, `sf` or `tt`. This is optional and if not given the overall default.

document series interface: Can be `md` or `bf`.

series value: This is the value that is going to be used with the combination is requested.

For example, `\DeclareFontSeriesDefault[rm]{bf}{sb}` would use **sb** (semi-bold) when `\rmfamily \bfseries` is asked for.

If used without the optional argument, e.g., `\DeclareFontSeriesDefault{bf}{b}` then this is like redefining `\bfdefault` or `\mddefault`.

If some family specific defaults aren't given, e.g. if there are no declarations for, say, `tt` then the format defaults of `\mddefault` and `\bfdefault` are assumed. If those are later changed this is *not* reflected!²⁷

`\DeclareFontSeriesDefault` The command to declare font series defaults for the “rm”, “sf” or “tt” family.

```

35 \let\DeclareFontSeriesDefault\@undefined      % for rollback
36 \newcommand\DeclareFontSeriesDefault[3][]{%
37   \def\reserved@a{#1}%

```

No optional argument: set up general default.

```

38   \ifx\reserved@a\@empty
39     \ifcsname #2series\endcsname                % supported are
40                                           % \[md/bf]default

```

Adding `\@empty` allows us to detect if the default gets redefined with `\renewcommand` or `\def` by the user.

```

41     \expandafter\def
42       \csname #2default\endcsname{#3\@empty}%
43     \expandafter\def
44       \csname #2default@previous\endcsname{#3\@empty}%
45   \else
46
47     \@latex@error{Wrong syntax for \string\DeclareFontSeriesDefault}%
48     {Mandatory first argument must be 'md' or 'bf'.}

```

Optional argument given, set up specific default.

```

49   \else
50     \ifcsname #2series@#1\endcsname              % supported are
51                                           % \[md/bf]series@[rm/sf/tt]
52     \expandafter\edef
53       \csname #2series@#1\endcsname{#3}%

```

If the interface is used we remove the frozen kernel default. This way, we know that something was explicitly set up (even if the setup has the same value as the default).

```

54     \expandafter\let
55       \csname #2series@#1@kernel\endcsname\@undefined
56   \else
57     \@latex@error{Wrong syntax for \string\DeclareFontSeriesDefault}%
58     {Optional argument must be 'rm', 'sf', or 'tt'. \MessageBreak
59       Mandatory first argument must be 'md' or 'bf'.}
60   \fi
61 \fi
62 }

```

(End definition for `\DeclareFontSeriesDefault`.)

`\mdseries@rm` We initialize the family specific default at the end of the format generation. Later on they may get overwritten in the preamble or a package via `\DeclareFontSeriesDefault` (or possibly directly).

`\mdseries@sf` Conceptual change: The `\bfdefault` will be `b` not `bx` because that is what it should be really for nearly every font except Computer/Latin Modern.

`\mdseries@tt`

²⁷I see no easy way to achieve this without compromising compatibility with existing packages that currently use `mweights` and directly define (some) of the `\mdseries@...` commands but not others.

To account for the fact that by default we typeset in CM or LM we set up the `\bfseries@..` defaults to use `bx` instead.

This means that it behaves like before because if the default fonts are used then `\bfseries@rm` etc kick in and make `\textbf` use `bx`. However, if the font gets changed then `\bfdefault` will get used.

```
63 \def\bfseries@rm{bx}
64 \def\bfseries@sf{bx}
65 \def\bfseries@tt{bx}
```

Frozen version of the kernel defaults so we can see if they have changed.

```
66 \let\bfseries@rm@kernel\bfseries@rm
67 \let\bfseries@sf@kernel\bfseries@sf
68 \let\bfseries@tt@kernel\bfseries@tt
```

The default for the medium series is `m` and this will be interpreted as resetting both weight and width. To reset only one of them the virtual value `?m` and `m?` are available.

```
69 \def\mdseries@rm{m}
70 \def\mdseries@sf{m}
71 \def\mdseries@tt{m}
```

(End definition for \mdseries@rm and others.)

`\series@change@debug` For debugging, but right now none of this code is extracted. The idea is to have a separate package with debugging code one day.

```
72 <*debug>
73 \let\series@change@debug\typeout
74 \let\series@change@debug\@gobble
75 </debug>
```

(End definition for \series@change@debug.)

`\prepare@family@series@update` This is core command that prepares for the family update. The big difference to the documented code above is that the nested `\ifx` statements seem to be missing. Instead we loop through an internal list that holds the names of the three meta families. This approach allows us to extend the mechanism at a later stage to allow for additional named meta families.

Here is the current definition of that list:

```
\@meta@family@list
76 \def\@meta@family@list{\@elt{rm}\@elt{sf}\@elt{tt}}

77 \def\prepare@family@series@update#1#2{%

78   \if@forced@series
79   <+debug> \series@change@debug{No series preparation (forced \f@series)\on@line}%
80     \fontfamily#2%
81   \else
82   <+debug> \series@change@debug{Preparing for switching to #1 (#2)\on@line}%
83     \expand@font@defaults
```

We prepare for changing the current series. We have to find it before changing the family as discussed above.

```
84   \let\target@series@value\@empty
85   \def\target@meta@family@value{#1}%
```

As the very last item in the meta family list we add `\@elt{??}` and define this pseudo meta family to be the current font family. So if none of the real meta families matched then this will match. This will cover the following case:

- `\bfseries` is called for a family using `bx` (e.g., CMR)
- Switch to a font family that is none of the meta families, e.g., via `\fontfamily{ptm}\allowbreak\vf`
- Then none of the real meta families, match but the final `\@elt{??}` will.
- Therefore if the current series is `\mddefault` or `\bfdefault` it will be detected and the corresponding target series selected.

```
86 \expandafter\edef\csname ??def@ult\endcsname{\f@family}%
```

To find it we loop over the meta family list with a suitable definition of `\@elt`.

```
87 \let\@elt\update@series@target@value
88 \meta@family@list
```

Last resort pseudo meta family. Will only be looked at if none of the real ones have matched.

```
89 \@elt{??}%
90 \let\@elt\relax
```

That will figure out the correct series value to use without updating it. Now we can change the family.

```
91 \fontfamily#2%
```

After that we update the series. That code is again like the one above.

```
92 \ifx\target@series@value\@empty
93 <+debug> \series@change@debug{Target series still empty ...}%
94 \else
95 \ifx \f@series\target@series@value
96 <+debug> \series@change@debug{Target series unchanged:
97 <+debug> \f@series \space = \target@series@value}%
98 \else
99 \maybe@load@fontshape
100 <+debug> \series@change@debug{Target series:
101 <+debug> \f@series \space -> \target@series@value}%
```

The `\target@series@value` may contain something like `cm` (coming from a default) and so we can't directly assign it to `\f@series` we have to drop any surplus `m` first.

```
102 % \let\f@series\target@series@value
103 \series@maybe@drop@one@m\target@series@value\f@series
104 \fi
105 \fi
106 \fi
107 }
```

(End definition for `\prepare@family@series@update` and `\meta@family@list`.)

`\update@series@target@value` In this macro used in the look you basically find the nested `\ifx`s from the outline above. The only difference is that it is parameterized instead of being written out and only for one block of tests because the code is called repeatedly when looping over the meta family list. From the list we get each meta family name in turn.

```
108 \def\update@series@target@value#1{%
```

There is one additional test at the beginning, because the list contains all meta families and we need to ignore the case where current one from the list and target one are identical.

```

109 \def\reserved@a{#1}%
110 \ifx\target@meta@family@value\reserved@a % rm -> rm do nothing
111 \else
112 <+debug> \series@change@debug{Trying to match #1: \csname#1def@ult\endcsname
113 <+debug> \space = \f@family\space ?}%

```

We only “do” something if the current font family matches the current meta family.

```

114 \expandafter\ifx\csname#1def@ult\endcsname\f@family

```

If that’s the case we know that this is the block that applies (only one meta family can match). So to speed things up we change \@elt so that the rest of the loop gets gobbled.

```

115 \let\@elt\@gobble

```

Then we try to find the right new value for the series (as explained above). The two macros defined first are only there because we now need to use \csname and this way the code will be a little faster.

```

116 \expandafter\let\expandafter\reserved@b
117 \csname mdseries@\target@meta@family@value\endcsname
118 \expandafter\let\expandafter\reserved@c
119 \csname bfseries@\target@meta@family@value\endcsname
120 <+debug>\series@change@debug{Targets for mdseries and bfseries:
121 <+debug> \reserved@b\space and \reserved@c}%

```

This here is now identical to the nested \ifx block from the outline, except that it there appeared twice in \rmfamily. This is now covered by looping and stopping the loop when a match was found.

We have to sanitize the default value first because it may contain something like mc and that would never match \f@series because there it would be called c with the m dropped. It would be probably better to do that differently these days, but it is hard to adjust without causing a lot of issues, so we do the dropping in various places instead.

```

122 \expandafter\series@maybe@drop@one@m
123 \csname mdseries@#1\endcsname\reserved@d
124 \ifx\reserved@d\f@series
125 <+debug> \series@change@debug{mdseries@#1 matched -> \reserved@b}%
126 \let\target@series@value\reserved@b
127 \else

```

Again do some sanitizing.

```

128 \expandafter\series@maybe@drop@one@m
129 \csname bfseries@#1\endcsname\reserved@d
130 \ifx\reserved@d\f@series
131 <+debug> \series@change@debug{bfseries@#1 matched -> \reserved@c}%
132 \let\target@series@value\reserved@c
133 \else\ifx\f@series\mdef@ult \let\target@series@value\reserved@b
134 <+debug> \series@change@debug{mdef@ult matched -> \reserved@b}%
135 \else\ifx\f@series\bfdef@ult \let\target@series@value\reserved@c
136 <+debug> \series@change@debug{bfdef@ult matched -> \reserved@c}%
137 \fi\fi\fi\fi
138 \fi
139 \fi
140 }

```

(End definition for \update@series@target@value.)

`\init@series@setup` This is code to be run at begin document ...

```
141 \def\init@series@setup{%
```

We only want `bx` in `\bfseries@rm` if the roman font is Computer Modern or Latin Modern, otherwise it should be `b`. It was set to `bx` in the kernel so that any font use with the default families in the preamble get this value. Now at the real document start we check if the fonts have been changed. If there was a `\DeclareFontSeriesDefault` declaration or `\bfseries@rm` was directly altered then it differs from `\bfseries@rm@kernel` and we do nothing. Otherwise we check if `\rmdefault` is one of the CM/LM font families and if so we keep `bx` otherwise we change it to `b`.

This approach doesn't cover one case: CM/LM got changed to a different family that supports `bx`, but the support package for that family used `\def\bfseries@rm{bx}` instead of using `\DeclareFontSeriesDefault`. In that case the code here changes it to `b`. Solution: use the `\DeclareFontSeriesDefault` interface.

```
142 \ifx\bfseries@rm@kernel\bfseries@rm
143   \expandafter\in@\expandafter{\rmdefault}%
144   {cmr,cmss,cmtt,lcmss,lcmtt,lmr,lmss,lmnt}%
145 \ifin@ \else \def\bfseries@rm{b}\fi\fi
```

Same approach for `\bfseries@sf` and `\bfseries@tt`:

```
146 \ifx\bfseries@sf@kernel\bfseries@sf
147   \expandafter\in@\expandafter{\sfdefault}%
148   {cmr,cmss,cmtt,lcmss,lcmtt,lmr,lmss,lmnt}%
149 \ifin@ \else \def\bfseries@sf{b}\fi\fi
150 \ifx\bfseries@tt@kernel\bfseries@tt
151   \expandafter\in@\expandafter{\ttdefault}%
152   {cmr,cmss,cmtt,lcmss,lcmtt,lmr,lmss,lmnt}%
153 \ifin@ \else \def\bfseries@tt{b}\fi\fi
```

If the document preamble has changed the `\familydefault` or if the if the `\rmdefault` contains a new font family, we may have to adjust the series defaults accordingly, before starting typesetting.

Similarly, if the user has changed the `\mddefault` or the medium series for the family selected as document font we may also have to adjust the `\seriesdefault`.

On the other hand if the document font is still CM or LM then `\bfdefault` is wrong, because it is now saying `b` and not `bx` as it should for such fonts.

To fix all this we first run `\reset@font` (the internal kernel name for `\normalfont`). This will set up the document encoding, family, series and shape based on the current values of `\encodingdefault`, `\familydefault`, `\seriesdefault` and `\shapedefault`. However, if the family (from `\familydefault`) has special medium default we should switch to that (and not use what is current value from `\seriesdefault`). This can be achieved by afterwards calling `\mediumseries` and then changing `\seriesdefault` to the now current series value (in `\f@series`).

But what should happen if `\seriesdefault` got explicitly changed? In that case the explicit change should survive and we should not alter `\seriesdefault`. This is solved by comparing the current value of `\seriesdefault` with a kernel version saved in the format and if they differ we do not call `\mdseries` or change `\seriesdefault`.

```
154 \reset@font
155 \ifx\seriesdefault\seriesdefault@kernel
156   \mdseries
157   \let\seriesdefault\f@series
158 \fi
159 }%
```

(End definition for `\init@series@setup`.)

As the kernel code now implements the same functionality as `mweights`, albeit internally coded slightly differently, that package shouldn't be loaded any more. We therefore pretend that it already got loaded. Thus, a font package that tries to load it and then sets `\mdseries@.`, etc. will continue to work but will now use the kernel code.

Of course, mid-term such package should probably use `\DeclareFontSeriesDefault` instead of making using low-level definitions.

```

160 \expandafter\let\csname ver@mweights.sty\endcsname\fmtversion
161 </2ekernel | latexrelease>
162 <latexrelease>\EndIncludeInRelease
163 <latexrelease>\IncludeInRelease{0000/00/00}%
164 <latexrelease>          {\DeclareFontSeriesDefault}{Custom series}%
165 <latexrelease>
166 <latexrelease>\let\DeclareFontSeriesDefault\@undefined
167 <latexrelease>\let\bfseries@rm\@undefined
168 <latexrelease>\let\bfseries@sf\@undefined
169 <latexrelease>\let\bfseries@tt\@undefined
170 <latexrelease>\let\bfseries@rm@kernel\@undefined
171 <latexrelease>\let\bfseries@sf@kernel\@undefined
172 <latexrelease>\let\bfseries@tt@kernel\@undefined
173 <latexrelease>\let\mdseries@rm\@undefined
174 <latexrelease>\let\mdseries@sf\@undefined
175 <latexrelease>\let\mdseries@tt\@undefined
176 <latexrelease>\expandafter\let\csname ver@mweights.sty\endcsname\@undefined
177 <latexrelease>
178 <latexrelease>\let\@meta@family@list\@undefined
179 <latexrelease>\let\prepare@family@series@update\@undefined
180 <latexrelease>\let@update@series@target@value\@undefined
181 <latexrelease>

```

This is always called in `\document` so don't make it undefined.

```

182 <latexrelease>\let\init@series@setup\relax
183 <latexrelease>
184 <latexrelease>\EndIncludeInRelease
185 <*2ekernel>
186 </2ekernel>
187 <*2ekernel | latexrelease>
188 <latexrelease>\IncludeInRelease{2020/10/01}%
189 <latexrelease>          {\bfseries}{Custom series with hooks}%

```

`\expand@font@defaults` The family specific defaults are fully expanded, i.e., they are defined via `\edef` inside `\DeclareFontSeriesDefault`. However, the overall defaults, e.g., `\bfdefault` may have been redefined by the user and thus may not be fully expanded. So to enable reliable comparison we make expanded versions of them. That we rerun each time. The alternative would be to only allow for changes before begin document.

```

\rm@def@ult
\sfb@def@ult
\tt@def@ult
\md@def@ult
\bf@def@ult
190 \def\expand@font@defaults{%
191   \edef\rmdef@ult{\rmdefault}%
192   \edef\sfbdef@ult{\sfdefault}%
193   \edef\ttdef@ult{\ttdefault}%

```

The series defaults may contain some surplus m that we need to drop here.

```

194   \series@maybe@drop@one@m\bfdefault\bfdef@ult
195   \series@maybe@drop@one@m\mddefault\mddef@ult

```

Formats that set up parallel fonts, e.g., for Japanese, can use this hook to add additional code here.

```
196 \UseHook{expand@font@defaults}%
197 }
```

(End definition for `\expand@font@defaults` and others.)

\bfseries This document command switches to the bold series.

```
198 \DeclareRobustCommand\bfseries{%
199 \not@math@alphabet\bfseries\mathbf
```

In the original NFSS definition it then called `\fontseries` with the value `\bfdefault`. In the new scheme we have more alternatives and therefore check if the current family (`\f@family`) is the current `\rmdef@ult`, `\sfdef@ult` or `\ttdef@ult` and then select the correct family default in that case.

```
200 \expand@font@defaults
```

If `\bfdefault` and `\bfdefault@previous` are different then the default got changed directly through the legacy interface (i.e., via `\def` or `\renewcommand`. In that case we reset all meta family defaults so that the document behaves like it was the case before the new mechanism was introduced.

```
201 \ifx\bfdefault\bfdefault@previous\else
```

We add `\@empty` and then let `\bfdefault@previous` to `\bfdefault` so that we can detect any further change.

```
202 \expandafter\def\expandafter\bfdefault
203 \expandafter{\bfdefault\@empty}%
204 \let\bfdefault@previous\bfdefault
```

And we reset the meta family defaults (`\bfdef@ult` is an expanded version of `\bfdefault`).

```
205 \let\bfseries@rm\bfdef@ult
206 \let\bfseries@sf\bfdef@ult
207 \let\bfseries@tt\bfdef@ult
```

Formats that set up parallel fonts, e.g., for Japanese, can use this hook to add resets here. Not that this hook is only run when resets are necessary.

```
208 \UseHook{bfseries/defaults}%
209 \fi
210 \ifx\f@family\rmdef@ult \fontseries\bfseries@rm
211 \else\ifx\f@family\sfdef@ult \fontseries\bfseries@sf
212 \else\ifx\f@family\ttdef@ult \fontseries\bfseries@tt
```

If not `\bfdefault` is used.

```
213 \else \fontseries\bfdefault
214 \fi\fi\fi
```

This hook in contrast is always executed.

```
215 \UseHook{bfseries}%
216 \selectfont
217 }
```

(End definition for `\bfseries`.)

`\mdseries` This document command switches to the medium series.

```
218 \DeclareRobustCommand\mdseries{%
219   \not@math@alphabet\mdseries\relax
220   \expand@font@defaults
221   \ifx\mddefault\mddefault@previous\else
222     \expandafter\def\expandafter\mddefault\expandafter{\mddefault\@empty}%
223     \let\mddefault@previous\mddefault
224     \let\mdseries@rm\mddef@ult
225     \let\mdseries@sf\mddef@ult
226     \let\mdseries@tt\mddef@ult
```

Formats that set up parallel fonts, e.g., for Japanese, can use this hook to add resets here.

```
227   \UseHook{mdseries/defaults}%
228   \fi
229   \ifx\f@family\rmdef@ult      \fontseries\mdseries@rm
230   \else\ifx\f@family\sfdef@ult \fontseries\mdseries@sf
231   \else\ifx\f@family\ttdef@ult \fontseries\mdseries@tt
232   \else                        \fontseries\mddefault
233   \fi\fi\fi
234   \UseHook{mdseries}%
235   \selectfont
236 }
```

(End definition for \mdseries.)

`\rmfamily` Here are the document level commands for changing the main font families, or rather, here is a documented outline of the code, the actual code is then streamlined and somewhat generalized.

```
DeclareRobustCommand\rmfamily{%
  \not@math@alphabet\rmfamily\mathrm
```

If families are changed then we have to do a bit more work. In the original NFSS implementation a family change kept encoding, series shape and size unchanged but now we can't any longer simply reuse the current series value. Instead we may have to change it from one family default to the next.

```
\expand@font@defaults
```

We have to do the testing while the current family is still unchanged but we have to do the adjustment of the series after it got changed (because the new family might have different sets of shapes available and we certainly don't want to see substitution going on. So we use `\target@series@value` to hold the target series (if any).

```
\let\target@series@value\@empty
```

Thus, if the current family is the sans family

```
\ifx\f@family\sfdef@ult
```

and if we using the medium series of the sans family

```
\ifx\f@series\mdseries@sf
```

then lets switch to the medium series for the serif family

```
\let\target@series@value\mdseries@rm
```

and if we use the bold series of the sans family switch to the bold default of the serif family:

```
\else\ifx\f@series\bfseries@sf \let\target@series@value\bfseries@rm
```

However, the sans family may not have any specific defaults set, so we also compare with the overall defaults.

```
\else\ifx\f@series\mddef@ult \let\target@series@value\mdseries@rm
\else\ifx\f@series\bfdef@ult \let\target@series@value\bfseries@rm
```

If neither test was true we leave the series alone. This way a special manual setting such as `\fontseries{lc}` is not undone if the family changes (of course there may not be any support for it in the new family but then the NFSS substitution kicks in and sorts it out).

```
\fi\fi\fi\fi
```

We need to do the same if the current family is the typewriter family:

```
\else\ifx\f@family\ttdef@ult
\ifx\f@series\mdseries@tt \let\target@series@value\mdseries@rm
\else\ifx\f@series\bfseries@tt \let\target@series@value\bfseries@rm
\else\ifx\f@series\mddef@ult \let\target@series@value\mdseries@rm
\else\ifx\f@series\bfdef@ult \let\target@series@value\bfseries@rm
\fi\fi\fi\fi
\fi\fi
```

With these preparations for series out of the way we can now change the font family to `\rmdefault`.

```
\fontfamily\rmdefault
```

If `\target@series@value` is still empty there is nothing more to do other than selecting the new family. However, if not then we should update the font series now as well. But there is one further subtle issue. We may not have loaded an `.fd` file for our target font family yet. In the past that was done in `\selectfont` if necessary but since we are now doing all the comparisons in `\fontseries` we need to make sure that the font family specifications are already loaded prior to calling `\fontseries`.

```
\ifx\target@series@value\@empty \else
\maybe@load@fontshape
```

Updating the series in this case means directly changing `\f@series` to the target value. We don't want to go through `\fontseries` because that would apply the mappings and then `bx + b` would keep `bx` instead of changing to `b` as desired. as

```
\let\f@series\target@series@value
\fi
\selectfont}
```

So now for the real definition: most of the code above gets delegated to a helper command `\prepare@family@series@update` so that the definition becomes again fairly short. In addition we add a hook, mainly for our Japanese friends so that the code can be extended prior to the call to `\selectfont`.

```
237 \DeclareRobustCommand\rmfamily{%
238   \not@math@alphabet\rmfamily\mathrm
```

This holds all the code discussed above, first argument is the meta family, i.e., `rm` in this case, and second argument is the default family name, e.g., `cmr` indirectly accessed via `\rmdefault`. This is calling `\fontfamily` and if necessary `\fontseries` as outline above.

```
239   \prepare@family@series@update{rm}\rmdefault
```

Then comes the hook code (by default a no-op) and finally the call to `\selectfont`.

```
240   \UseHook{rmfamily}%
241   \selectfont}
```

The definitions for `\sffamily` and `\ttfamily` are similar, the differences are only in what font families get checked.

```
\sffamily
\sffamily 242 \DeclareRobustCommand\sffamily{%
243   \not@math@alphabet\sffamily\mathsf
244   \prepare@family@series@update{sf}\sfdefault
245   \UseHook{sffamily}%
246   \selectfont}

247 \DeclareRobustCommand\ttfamily{%
248   \not@math@alphabet\ttfamily\mathtt
249   \prepare@family@series@update{tt}\ttdefault
250   \UseHook{ttfamily}%
251   \selectfont}
```

(End definition for \rmfamily, \sffamily, and \ttfamily.)

```
rmfamily  Declare the hooks used above.
sffamily  252 \NewHook{rmfamily}
ttfamily  253 \NewHook{sffamily}
normalfont 254 \NewHook{ttfamily}
expand@font@defaults 255 \NewHook{normalfont}
bfseries  256 \NewHook{expand@font@defaults}
bfseries/defaults 257 \NewHook{bfseries}
mdseries  258 \NewHook{bfseries/defaults}
mdseries/defaults 259 \NewHook{mdseries}
260 \NewHook{mdseries/defaults}
```

(End definition for rmfamily and others.)

```
\@rmfamilyhook These four hooks have legacy versions used in 2020/02/02 so we should support them
\@sffamilyhook until they aren't any longer used.
\@ttfamilyhook By default the hooks do nothing.
\@defaultfamilyhook
261 \let\@rmfamilyhook\@empty
262 \let\@sffamilyhook\@empty
263 \let\@ttfamilyhook\@empty
264 \let\@defaultfamilyhook\@empty %Fmi sort out
```

(End definition for \@rmfamilyhook and others.)

```
265 </2ekernel | latexrelease>
266 <latexrelease>\EndIncludeInRelease
267 <latexrelease>\IncludeInRelease{2020/02/02}%
268 <latexrelease>          {\bfseries}{Custom series with hooks}%
269 <latexrelease>
270 <latexrelease>\def\expand@font@defaults{%
271 <latexrelease>  \edef\rmdef@ult{\rmdefault}%
272 <latexrelease>  \edef\sfdef@ult{\sfdefault}%
273 <latexrelease>  \edef\ttdef@ult{\ttdefault}%
274 <latexrelease>  \edef\bfdef@ult{\bfdefault}%
275 <latexrelease>  \edef\mddef@ult{\mddefault}%
276 <latexrelease>  \edef\famdef@ult{\familydefault}%
277 <latexrelease>}
278 <latexrelease>
279 <latexrelease>\DeclareRobustCommand\bfseries{%
280 <latexrelease>  \not@math@alphabet\bfseries\mathbf
281 <latexrelease>  \expand@font@defaults
282 <latexrelease>    \ifx\f@family\rmdef@ult      \fontseries\bfseries@rm
283 <latexrelease>    \else\ifx\f@family\sfdef@ult  \fontseries\bfseries@sf
284 <latexrelease>    \else\ifx\f@family\ttdef@ult  \fontseries\bfseries@tt
285 <latexrelease>    \else                                \fontseries\bfdefault
286 <latexrelease>    \fi\fi\fi
287 <latexrelease>  \selectfont
288 <latexrelease>}
289 <latexrelease>
290 <latexrelease>\DeclareRobustCommand\mdseries{%
291 <latexrelease>  \not@math@alphabet\mdseries\relax
292 <latexrelease>  \expand@font@defaults
293 <latexrelease>    \ifx\f@family\rmdef@ult      \fontseries\mdseries@rm
294 <latexrelease>    \else\ifx\f@family\sfdef@ult  \fontseries\mdseries@sf
295 <latexrelease>    \else\ifx\f@family\ttdef@ult  \fontseries\mdseries@tt
296 <latexrelease>    \else                                \fontseries\mddefault
297 <latexrelease>    \fi\fi\fi
298 <latexrelease>  \selectfont
299 <latexrelease>}
300 <latexrelease>
301 <latexrelease>\DeclareRobustCommand\rmfamily{%
302 <latexrelease>  \not@math@alphabet\rmfamily\mathrm
303 <latexrelease>  \prepare@family@series@update{rm}\rmdefault
304 <latexrelease>  \@rmfamilyhook
305 <latexrelease>  \selectfont}
306 <latexrelease>\DeclareRobustCommand\sffamily{%
307 <latexrelease>  \not@math@alphabet\sffamily\mathsf
308 <latexrelease>  \prepare@family@series@update{sf}\sfdefault
309 <latexrelease>  \@sffamilyhook
310 <latexrelease>  \selectfont}
311 <latexrelease>\DeclareRobustCommand\ttfamily{%
312 <latexrelease>  \not@math@alphabet\ttfamily\mathtt
313 <latexrelease>  \prepare@family@series@update{tt}\ttdefault
314 <latexrelease>  \@ttfamilyhook
315 <latexrelease>  \selectfont}
316 <latexrelease>\let\@rmfamilyhook\@empty
317 <latexrelease>\let\@sffamilyhook\@empty
```

```

318 <latexrelease>\let\@ttfamilyhook\@empty
319 <latexrelease>
320 <latexrelease>
321 <latexrelease>\EndIncludeInRelease
322 <latexrelease>\IncludeInRelease{0000/00/00}%
323 <latexrelease>                {\bfseries}{Custom series with hooks}%
324 <latexrelease>
325 <latexrelease>\let\expand@font@defaults\@undefined
326 <latexrelease>
327 <latexrelease>\DeclareRobustCommand\bfseries
328 <latexrelease>        {\not@math@alphabet\bfseries\mathbf}
329 <latexrelease>        \fontseries\bfdefault\selectfont}
330 <latexrelease>\DeclareRobustCommand\mdseries
331 <latexrelease>        {\not@math@alphabet\mdseries\relax
332 <latexrelease>        \fontseries\mddefault\selectfont}
333 <latexrelease>\DeclareRobustCommand\rmfamily
334 <latexrelease>        {\not@math@alphabet\rmfamily\mathrm
335 <latexrelease>        \fontfamily\rmdefault\selectfont}
336 <latexrelease>\DeclareRobustCommand\sffamily
337 <latexrelease>        {\not@math@alphabet\sffamily\mathsf
338 <latexrelease>        \fontfamily\sfdefault\selectfont}
339 <latexrelease>\DeclareRobustCommand\ttfamily
340 <latexrelease>        {\not@math@alphabet\ttfamily\mathtt
341 <latexrelease>        \fontfamily\ttdefault\selectfont}
342 <latexrelease>
343 <latexrelease>\let\@rmfamilyhook\@undefined
344 <latexrelease>\let\@sffamilyhook\@undefined
345 <latexrelease>\let\@ttfamilyhook\@undefined
346 <latexrelease>
347 <latexrelease>\EndIncludeInRelease
348 <*2ekernel>

```

`\IfFontSeriesContextTF` With the ability for `\bfseries` or `\mdseries` to be mapped to different NFSS axis values it becomes important to have the ability to determine the current context as we can no longer look at `\f@series` to answer a question such as “am I currently typesetting in a bold typeface?”

This is provided by the test `\IfFontSeriesContextTF`. It takes three arguments:

- The context we try to check (either `bf` for bold or `md` for medium, i.e., the same that can go into the first mandatory argument of `\DeclareFontSeriesDefault`),
- what to do if we are in this context (true case) and
- what to do if we are not (false case).

This allows you to define commands like `\IfBold`, e.g.,

```
\newcommand\IfBold[2]{\IfSeriesContextTF{bf}{#1}{#2}}
```

and then do

```
This is \IfBold{bold}{non-bold} text.
```

and get the appropriate result.

```

349 </2ekernel>
350 <*2ekernel | latexrelease>
351 <latexrelease>\IncludeInRelease{2020/10/01}%
352 <latexrelease>{\IfFontSeriesContextTF}{Font series context}%
353 \DeclareRobustCommand\IfFontSeriesContextTF[1]{%
354   \expand@font@defaults

```

In the beginning we haven't found the context we are looking for.

```

355   \@font@series@contextfalse

```

We store the requested context away for use in the tests.

```

356   \def\requested@test@context{#1}%

```

The next definition is there to ensure that get a final match during testing even if the current family is non of the meta families (`rm`, `sf` or `tt`). This will then basically tests if the current font family matches the overall default.

```

357   \expandafter\edef\csname ??def@ult\endcsname{\f@family}%

```

Then we run through the meta family list (currently containing just the three values) followed by the artificial meta family `??` and test each of them in turn using `\test@font@series@context` as the testing command.

```

358   \let\@elt\test@font@series@context
359   \@meta@family@list
360   \@elt{??}%
361   \let\@elt\relax

```

Following that we evaluate the status of `\if@font@series@context` to determine which of the remaining arguments (true/false case) we have to execute.

```

362   \if@font@series@context
363   \expandafter\@firstoftwo
364   \else
365   \expandafter\@secondoftwo
366   \fi
367 }

```

(End definition for `\IfFontSeriesContextTF`.)

`\test@font@series@context` This tests the context (stored in `\requested@test@context`) and updates the boolean if the right context is found.

```

368 \def\test@font@series@context#1{%

```

First task is to figure out whether the current family matches `\rmfamily`, `\sffamily`, etc. so in `\reserved@a` we store the value of `\rmdef@ult` (or whatever the given meta family is) and compare that to `\f@family`.

```

369   \edef\reserved@a{\csname #1def@ult\endcsname}%
370   \ifx\f@family\reserved@a

```

If they match we have found the right meta family so we don't need to test any of the remaining meta family and therefore change `\@elt` to `\@gobble`.

```

371   \let\@elt\@gobble

```

Now we have to test if `\f@series` matches the requested context (e.g., whether `\bfseries@rm` has that value if the current meta family is `rm` and we are looking for the `bf` context).

```
372 \expandafter\ifx
373 \csname\requested@test@context series@#1\endcsname\f@series
```

If yes we change the boolean and are done.

```
374 \@font@series@contexttrue
```

If not then maybe the reason is that nothing special was set up for that meta family so we also check now check if `\f@series` matches the overall default (e.g., `\bfdef@ult` if we are looking for the bold context). If that matches we change the boolean.

```
375 \else
376 \expandafter\ifx
377 \csname\requested@test@context def@ult\endcsname\f@series
378 \@font@series@contexttrue
379 \fi\fi\fi
380 }
```

(End definition for `\test@font@series@context`.)

`\if@font@series@context` The boolean to signal if we found the requested font series context.

```
381 \newif\if@font@series@context
```

(End definition for `\if@font@series@context`.)

```
382 </2kernel | latexrelease>
383 <latexrelease>\EndIncludeInRelease
384 <latexrelease>\IncludeInRelease{0000/00/00}%
385 <latexrelease> \IfFontSeriesContextTF}{Font series context}%
386 <latexrelease>
387 <latexrelease>\let\IfFontSeriesContextTF\@undefined
388 <latexrelease>\let\test@font@series@context\@undefined
389 <latexrelease>\let\if@font@series@context\@undefined
390 <latexrelease>\let\@font@series@contexttrue\@undefined
391 <latexrelease>\let\@font@series@contextfalse\@undefined
392 <latexrelease>\EndIncludeInRelease
393 <*2kernel>
```

3 Supporting nested emphasis

By default $\text{\LaTeX} 2_{\epsilon}$ supports two levels of nested emphasis: if the current font has an upright shape then it switches to `\itshape` otherwise to `\emminnershape` (which defaults to `\upshape`). This means nested emphasis will oscillate between italic and upright shapes.

Sometimes it would be nice to allow for a more lengthy sequence, but instead of providing a fixed one \LaTeX now offers a general mechanism that allows to define arbitrary sequences.

```
\DeclareEmphSequence
\emforce
```

This declaration expects a comma separated list of (font) change declarations corresponding to increasing levels of emphasis. The mechanism tries to be “smart” and verifies that the declarations actually alter the font. If not it will ignore this level and tries the next one—the assumption being that there was a manual font change in the document

to the font that is now supposed to be used for emphasis. Of course, this only works if the declarations in the list actually change the font and not, say, just the color. In such a case one has to use `\emforce` to which directs the mechanism to use the level even if the font attributes haven't changed.

`\emreset` If the nesting is so deep, that the specified levels are exhausted then `\emreset` is used as a final set of declarations (which by default returns back to the upright shape). Any additional nesting levels will then reuse the list from its beginning.

`\DeclareEmphSequence` `\DeclareEmphSequence` expects a list of declaration. Spaces in the argument are dropped to avoid spurious spaces in the output. The declarations are additive. At the very end the shape is reset using `\emreset` and `\emforce` so that this case is never skipped.²⁸ Further nested calls restart at the beginning.

```
394 </2ekernel>
395 <*2ekernel | latexrelease>
396 <latexrelease>\IncludeInRelease{2020/02/02}%
397 <latexrelease>{\DeclareEmphSequence}{Nested emph}%
398 \def\DeclareEmphSequence#1{%
399   \protected@edef\emfontdeclare@clist{\zap@space#1, \@empty\emforce\emreset}%
400 }
```

By default the it is empty, in which case `\eminnershape` is used by L^AT_EX.

```
401 \let\emfontdeclare@clist\@empty
```

(End definition for \DeclareEmphSequence.)

`\emrest` Reset the font to upright and upper/lower case. With the default rules using `\shapedefault` does that for us but to be on the safe side we do it like this:

```
402 \DeclareRobustCommand\emreset{\upshape\ulcshape}
```

(End definition for \emrest.)

`\em` The new definition for `\em` (and implicitly `\emph`) is the same as before as long as `\emfontdeclare@clist` is empty.

```
403 \DeclareRobustCommand\em{%
404   \@nomath\em
405   \ifx\emfontdeclare@clist\@empty
406     \ifdim \fontdimen\@ne\font >\z@
407       \eminnershape \else \itshape \fi
408   \else
```

But if not we use the list to decide how to do emphasis.

We use the current font to check if the declarations have any effect, so even a size change is allowed and identified as a modification (but a color change, for example, isn't). So first we save the current status.

```
409   \edef\em@currfont{\csname\curr@fontshape/\f@size\endcsname}%

```

Then we grab the next element from the list and check if it can be used.

```
410   \expandafter\do@emfont@update\emfontdeclare@clist\do@emfont@update
411   \fi
412 }
```

²⁸Maybe we should not add `\emforce` but allow that case to be skipped as well. Of course, that might result in an endless loop if somebody defines a sequence without any font change and without `\emforce` but ...

413 \def\emminnershape{\upshape}

(End definition for \em.)

\do@emfont@update We know that the list (if not empty) has at least 2 elements separated by a comma, so we pick up the first in #1 and the rest in #2.

414 \def\do@emfont@update#1,#2\do@emfont@update{%

First action is to alter the list and move the first entry to the end

415 \def\emfontdeclare@clist{#2,#1}%

Then we execute current declaration. Appending \selectfont means one can write just \fontshape{it}} and that works then too.

416 % \typeout{Use: \detokenize{#1}}%

417 #1\selectfont

We then compare the current font with our saved version, but with a slight twist: we add \em@force at the end of the name. Normally this is empty so has no effect but if there was an \emforce as part of #1 it will append a / to the font name (making it invalid) thus this will then always fail the test.

If the test fails we are done and the declarations will be used. Otherwise we will try the next declaration in the sequence.

418 \expandafter\ifx\csname \curr@fontshape/\f@size\em@force

For the comparison with \ifx we have to expand \em@currfont once as the relevant info is inside.

419 \expandafter\endcsname

420 \em@currfont

421 \expandafter\do@emfont@update\emfontdeclare@clist\do@emfont@update

If \emforce was used, we have to undo its effect:

422 \else

423 \let\em@force\@empty

424 \fi

425 }

(End definition for \do@emfont@update.)

\emforce The definition of \emforce is simple: change \em@force to make the above test always invalid.

426 \protected\def\emforce{\def\em@force{/}}

427 \let\em@force\@empty

428 </2ekernel | latexrelease>

429 <latexrelease>\EndIncludeInRelease

(End definition for \emforce and \em@force.)

\em These are the older definitions for \em, prior to 2020.

\emminnershape We also have to define the *emphasize* font change command (i.e. \em). This command will look is the current font is sloped (i.e. has a positive \fontdimen1) and will then select either \upshape or \itshape.

430 <latexrelease>\IncludeInRelease{2015/01/01}{\DeclareEmphSequence}{Nested emph}%

431 <latexrelease>\let\DeclareEmphSequence\@undefined

432 <latexrelease>\let\emfontdeclare@clist\@undefined

433 <latexrelease>\let\emreset\@undefined

434 <latexrelease>\let\do@emfont@update\@undefined

```

435 \let\emforce\@undefined
436 \let\em@force\@undefined
437 \let\em\em
438 \DeclareRobustCommand\em
439 {\@nomath\em \ifdim \fontdimen\@ne\font >\z@
440 \emminnershape \else \itshape \fi}%
441 \EndIncludeInRelease
442 \let\em\em
443 \IncludeInRelease{0000/00/00}{\DeclareEmphSequence}{Nested emph}%
444 \DeclareRobustCommand\em
445 {\@nomath\em \ifdim \fontdimen\@ne\font >\z@
446 \emminnershape \else \itshape \fi}%
447 \let\emminnershape\@undefined
448 \EndIncludeInRelease
449 \*2ekernel)

```

(End definition for `\em` and `\emminnershape`.)

`\not@math@alphabet` This function generates an error message when it is called in math mode. The same function should be defined in `newlfont.sty`.

```

450 \def\not@math@alphabet#1#2{%
451   \relax
452   \ifmmode
453     \@latex@error{Command \noexpand#1invalid in math mode}%
454     {%
455       Please
456       \ifx#2\relax
457         define a new math alphabet^^J%
458         if you want to use a special font in math mode%
459       \else
460         use the math alphabet \noexpand#2instead of
461         the #1command%
462       \fi
463       .
464     }%
465   \fi}

```

(End definition for `\not@math@alphabet`.)

Finally we provide two abbreviations to switch to the L^AT_EX versions.

```

466 \DeclareRobustCommand\boldmath{\@nomath\boldmath
467   \mathversion{bold}}
468 \DeclareRobustCommand\unboldmath{\@nomath\unboldmath
469   \mathversion{normal}}

```

Here we switch to the default math version by defining the internal macro `\math@version`. We dare not to call `\mathversion` at this place because this would call `\glb@settings`.

```

470 \def\math@version{normal}

```

3.1 Legacy

We start by defining a few macros that are part of standard L^AT_EX's user interface. The use of these functions is not encouraged, but they will allow to process older documents without changes to the source.

`\newfont`

```
471 \def\newfont#1#2{\@ifdefinable#1{\font#1=#2\relax}}
```

(End definition for \newfont.)

`\symbol`

```
472 </2ekernel>
473 <*2ekernel | latexrelease>
474 <latexrelease>\IncludeInRelease{2020/10/01}%
475 <latexrelease>          {\symbol}{XeTeX change for math}%
476 \ifdefined\XeTeXversion
477   \DeclareRobustCommand\symbols[1]{\Ucharcat#1 12\relax}
478 \else
479   \DeclareRobustCommand\symbols[1]{\char#1\relax}
480 \fi
481 </2ekernel | latexrelease>
482 <latexrelease>\EndIncludeInRelease
483 <latexrelease>\IncludeInRelease{0000/00/00}%
484 <latexrelease>          {\symbol}{XeTeX change for math}%
485 <latexrelease>
486 <latexrelease>\DeclareRobustCommand\symbols[1]{\char#1\relax}
487 <latexrelease>
488 <latexrelease>\EndIncludeInRelease
489 <*2ekernel>
```

(End definition for \symbol.)

3.2 Miscellaneous

`\@setfontsize`

This abbreviation is used by L^AT_EX's user level size changing commands, such as `\large`.

`\@setsize`

```
490 \def\@setfontsize#1#2#3{\@nomath#1%
```

For the benefit of people relying on keeping the name of the current font command saved in `\@currsz` we define it. To ensure that `\@setfontsize` keeps being robust we omit this assignment during times where `\protect` differs from `\@typeset@protect`.

```
491   \ifx\protect\@typeset@protect
492     \let\@currsz#1%
493   \fi
494   \fontsize{#2}{#3}\selectfont}
```

For compatibility we also define `\@setsize` the 209 command

```
495 <*compat>
496 \def\@setsize#1#2#3#4{\@setfontsize#1{#4}{#2}}
497 </compat>
```

(End definition for \@setfontsize and \@setsize.)

`\hexnumber@` To set up L^AT_EX's special math character definitions we first provide a macro to generate hexadecimal numbers. It is a rather simple `\ifcase`.

```

498 \def\hexnumber@#1{\ifcase\number#1
499 0\or 1\or 2\or 3\or 4\or 5\or 6\or 7\or 8\or
500 9\or A\or B\or C\or D\or E\or F\fi}

```

(End definition for `\hexnumber@`.)

`\nfss@text` In its simplest form `\nfss@text` is an `\mbox`. This will produce unbreakable text outside math and inside math you will get text with the same fonts as outside. The only drawback is that such item won't change sizes in subscripts. But this behavior can be easily changed. With the `amstex` style option one will get a sub style called `amstext` which will redefine the `\nfss@text` macro to produce correct text in all sizes.

We have to use `\def` instead of the shorter `\let` since `\mbox` is undefined when we reach this point.

```

501 \def\nfss@text#1{{\mbox{#1}}}

```

(End definition for `\nfss@text`.)

`\copyright` The definition of `\copyright` was changed so that it works in other type styles, and to make it robust. We leave the family untouched so that the copyright notice will come out differently if a different font family is in use. This command is commented out, since it is now defined in `ltoutenc.dtx`.

```

502 %\DeclareRobustCommand\copyright
503 %    {\ooalign{\hfil
504 %        \raise.07ex\hbox{\mdseries\upshape c}\hfil\crcr
505 %        \mathhexbox20D}}

```

(End definition for `\copyright`.)

`\normalfont` The macro `\reset@font` is used in L^AT_EX to switch to a standard font, in order to initialize the current font in situations where typesetting is done in a new visual context (e.g. in a footnote). We define it here to allow the test for the new L^AT_EX version above but nevertheless are able to run all kind of mixtures.

The user interface name for `\reset@font` is `\normalfont`:

```

506 </2ekernel>
507 <*2ekernel | latexrelease>
508 <latexrelease>\IncludeInRelease{2021/06/01}%
509 <latexrelease>          {\normalfont}{Add hook to \normalfont}%
510 \DeclareRobustCommand\normalfont{%

```

Instead of calling `\usefont`, as it was done in the past, we inline the code from `\usefont` as we want to add the hook before `\selectfont`, but after all the font attributes are set.

```

511 \fontencoding\encodingdefault
512 \edef\f@family{\familydefault}%
513 \edef\f@series{\seriesdefault}%
514 \edef\f@shape{\shapedefault}%

```

Any earlier `\fontseries`, etc. should be canceled and we should switch unconditionally to the requested font face so we drop any code that may have been stored in `\delayed@f@adjustment`.

```

515 \let\delayed@f@adjustment\@empty
516 \UseHook{normalfont}%

```

This is the old name for the hook introduced in 2020/02/02. It will be removed in one of the future releases!

```

517 \@defaultfamilyhook          % hookname from 2020/02 will vanish
518 \selectfont}

519 \let\reset@font\normalfont

(End definition for \normalfont and \reset@font.)

520 % \changes{v3.2g}{2021/03/18}
521 % {Add missing 2020/02/02 latexrelease entry.}
522 </2ekernel | latexrelease>
523 <latexrelease>\EndIncludeInRelease
524 <latexrelease>
525 <latexrelease>\IncludeInRelease{2020/10/01}%
526 <latexrelease> { \normalfont } {Add hook to \normalfont}%
527 <latexrelease>
528 <latexrelease>\DeclareRobustCommand\normalfont{%
529 <latexrelease> \fontencoding\encodingdefault
530 <latexrelease> \edef\f@family{\familydefault}%
531 <latexrelease> \edef\f@series{\seriesdefault}%
532 <latexrelease> \edef\f@shape{\shapedefault}%
533 <latexrelease> \UseHook{normalfont}%
534 <latexrelease> \@defaultfamilyhook          % hookname from 2020/02 will vanish
535 <latexrelease> \selectfont}
536 <latexrelease>
537 <latexrelease>\let\reset@font\normalfont
538 <latexrelease>
539 <latexrelease>\EndIncludeInRelease
540 <latexrelease>
541 <latexrelease>\IncludeInRelease{2020/02/02}%
542 <latexrelease> { \normalfont } {Add hook to \normalfont}%
543 <latexrelease>
544 <latexrelease>\DeclareRobustCommand\normalfont{%
545 <latexrelease> \fontencoding\encodingdefault
546 <latexrelease> \edef\f@family{\familydefault}%
547 <latexrelease> \edef\f@series{\seriesdefault}%
548 <latexrelease> \edef\f@shape{\shapedefault}%
549 <latexrelease> \@defaultfamilyhook
550 <latexrelease> \selectfont}
551 <latexrelease>
552 <latexrelease>\let\reset@font\normalfont
553 <latexrelease>
554 <latexrelease>\let\@defaultfamilyhook\@empty
555 <latexrelease>
556 <latexrelease>\EndIncludeInRelease
557 <latexrelease>
558 <latexrelease>\IncludeInRelease{0000/00/00}%
559 <latexrelease> { \normalfont } {Add hook to \normalfont}%
560 <latexrelease>
561 <latexrelease>\DeclareRobustCommand\normalfont
562 <latexrelease> { \usefont\encodingdefault
563 <latexrelease> \familydefault
564 <latexrelease> \seriesdefault
565 <latexrelease> \shapedefault

```

```

566 <latexrelease> \relax}
567 <latexrelease>\let\reset@font\normalfont
568 <latexrelease>
569 <latexrelease>\let\@defaultfamilyhook\@undefined
570 <latexrelease>
571 <latexrelease>\EndIncludeInRelease
572 <*2ekernel>

```

We left out the special L^AT_EX fonts which are not automatically included in the base version of the font selection since these fonts contain only a few characters which are also included in the AMS fonts so anybody who is using these fonts doesn't need them. But for compatibility reasons we will define these symbols.

```

573 \def\not@base#1{\@latex@error
574   {Command \noexpand#1not provided in base LaTeX2e}%
575   {Load the latexsym or the amsfonts package to
576    define this symbol}}
577 \def\mho{\not@base\mho}
578 \def\Join{\not@base\Join}
579 \def\Box{\not@base\Box}
580 \def\Diamond{\not@base\Diamond}
581 \def\leadsto{\not@base\leadsto}
582 \def\squsubset{\not@base\squsubset}
583 \def\sqsupset{\not@base\sqsupset}
584 \def\lhd{\not@base\lhd}
585 \def\unlhd{\not@base\unlhd}
586 \def\rhd{\not@base\rhd}
587 \def\unrhd{\not@base\unrhd}

```

We now initialize all variables set by `\DeclareErrorFont`. These values are not really important since they will be overwritten later on by the definition in `fontdef.ltx`.

However, if `fontdef.cfg` is corrupted then at least a hopefully suitable error font is present.

```

588 \DeclareErrorFont{OT1}{cmr}{m}{n}{10}  %% don't modify this setting
589                                         %% overwrite it in fontdef.cfg
590                                         %% if necessary

```

We also set some default values for `\f@family` etc. Note that we don't yet have any encodings that comes later. In the past this was implicitly done by `\DeclareErrorFont`.

```

591 \fontfamily{cmr}

```

Previously the default values for series and shape were set by calling `\fontseries` and `\fontshape`, but their action is now delayed until `\selectfont` which isn't called inside the format (to avoid unnecessarily loading a font that may never get used). We therefore have to set `\f@series` and `\f@shape` directly instead.

```

592 \def\f@series{m}          % \fontseries{m}
593 \def\f@shape{n}          % \fontshape{n}
594 \fontsize{10}{10}

```

The initial `fontenc` package load list. This will get overwritten in `fonttext` and is only provided in case an old `fonttext.cfg` does not define the command:

```

595 \def\@fontenc@load@list{\@elt{T1,OT1}}

```

We now load the customizable parts of NFSS.

```

596 \InputIfFileExists{fonttext.cfg}
597     {\typeout{=====^^J%
598             ^^J%
599             Local config file fonttext.cfg used^^J%
600             ^^J%
601             =====}%
602     \def\@addtofilelist##1{\xdef\@filelist{\@filelist,##1}}%
603     }
604     {\input{fonttext.ltx}}
605 \let\@addtofilelist\@gobble

```

Ditto for math although I don't think that we will get a lot of customisation :-)

```

606 \InputIfFileExists{fontmath.cfg}
607     {\typeout{=====^^J%
608             ^^J%
609             Local config file fontmath.cfg used^^J%
610             ^^J%
611             =====}%
612     \def\@addtofilelist##1{\xdef\@filelist{\@filelist,##1}}%
613     }
614     {\input{fontmath.ltx}}
615 \let\@addtofilelist\@gobble

```

Then we preload several fonts. This file might be customized *without* changing the behavior of the format (i.e. necessary font definitions will be loaded at runtime if they are not preloaded). This is done in the file `preload.ltx`.

```

616 \InputIfFileExists{preload.cfg}
617     {\typeout{=====^^J%
618             ^^J%
619             Local config file preload.cfg used^^J%
620             ^^J%
621             =====}%
622     \def\@addtofilelist##1{\xdef\@filelist{\@filelist,##1}}%
623     }
624     {\input{preload.ltx}}
625 \let\@addtofilelist\@gobble

```

`\seriesdefault` After `\seriesdefault` got defined inside `fonttext.ltx` or a `.cfg` file overwriting it, we
`\seriesdefault@kernel` alter its value by appending `\@empty` to it. This will vanish if expanded but allows us to check if the default gets altered (even to the same value) in the document preamble. All we have to do is to save the current value somewhere and later compare the two. For this we use `\seriesdefault@kernel`.

```

626 \expandafter\def\expandafter\seriesdefault\expandafter{\seriesdefault\@empty}
627 \let\seriesdefault@kernel\seriesdefault

```

(End definition for `\seriesdefault` and `\seriesdefault@kernel`.)

`\@accii` We also save the values of some accents in `\@accii`, `\@accii` and `\@acciii` so they can
`\@accii` be restored by a `minipage` inside a `tabbing` environment.
`\@acciii` 628 \let\@accii\' \let\@accii\' \let\@acciii\=

(End definition for `\@accii`, `\@accii`, and `\@acciii`.)

`\cal` Here were the two old *alphabet identifiers*.
`\mit`

(End definition for \cal and \mit.)

629 \end{kernel}

File A

fontdef.dtx

<-latexrelease> [2021/01/15 v3.0i LaTeX Kernel (<-latexrelease> font setup)]

1 Introduction

This file is used to generate the files `fonttext.ltx` (text font declarations) and `fontmath.ltx` (math font declarations), which are used during the format generation. It contains the declaration of the standard text encodings used at the site as well as a minimal subset of font shape groups that NFSS will look at to ensure that the specified encodings are valid.

The math part contains the setup for math encodings as well as the default math symbol declarations that belong to the encoding.

It is possible to change this setup (by using other fonts, or defaults) without losing the ability to process documents written at other sites. Portability in this sense means that a document will compile without errors. It does not mean, however, that identical output will be produced. For this it is necessary that the distributed setup is used at both installations.

2 Customization

You are not allowed to change this source file! If you want to change the default encodings and/or the font shape groups preloaded you should create a copy of `fonttext.ltx` under the name `fonttext.cfg` and change this copy. If $\text{\LaTeX} 2_{\epsilon}$ finds a file of this name it will use it, otherwise it uses the standard file which is `fontdef.ltx`.

If you don't plan to use Computer Modern much or at all, it might (!) be a good idea to make your own `fonttext.cfg`. Look at the comments below (docstrip module 'text') to see what should go into such a file.

To change the math font setup use a copy of `fontmath.ltx` under the name `fontmath.cfg` and change this copy. However, dealing with this interface is even more a job for an expert than changing the text font setup — in short, we don't encourage either.

Warning: please note that we don't support customised \LaTeX versions. Thus, before sending in a bug report please try your test file with a \LaTeX format which is not customised and send in the log from that version (unless the problem goes away).

Please note: the following standard encodings have to be defined in all local variants of `font....cfg` to guarantee that all \LaTeX installations behave in the same way.

T1	Cork \TeX text encoding
OT1	old \TeX text encoding
U	unknown encoding
OML	old \TeX math letters encoding
OMS	old \TeX math symbols encoding
OMX	old \TeX math extension symbols encoding
TU	Unicode

Notice that some of these encodings are ‘old’ in the sense that we hope that they will be superseded soon by encoding standards defined by the T_EX user community. Therefore this set of default encodings may change in the future.

The first candidate is OT1 which will soon be replaced by T1, the official T_EX text encoding.

Warning: If you add additional encodings to this file there is no guarantee any longer that files processable at your installation will also be processable at other installations. Thus, if you make use of such an encoding in your document, e.g. if you intend to typeset in Cyrillic (OT2 encoding), you need to specify this encoding in the preamble of your document prior to sending it to another installation. Once the encoding is specified in that place in your document, the document is processable at all L^AT_EX installations (provided they have suitable fonts installed).

For this reason we suggest that you define a short package file that sets up an additional encoding used at your site (rather than putting the encoding into this file) since this package can easily be shipped with your document.

3 The docstrip modules

The following modules are used to direct **docstrip** in generating external files:

driver	produce a documentation driver file
text	produce the file fonttext.ltx
math	produce the file fontmath.ltx
cfgtext	produce a dummy fonttext.cfg file
cfgmath	produce a dummy fontmath.cfg file

A typical **docstrip** command file would then have entries like:

```
generateFile{fonttext.ltx}{t}{\from{fontdef.dtx}{text}}
```

4 A driver for this document

The next bit of code contains the documentation driver file for T_EX, i.e. the file that will produce the documentation you are currently reading. It will be extracted from this file by the DOCSTRIP program.

```
1 <*driver>
2 \documentclass{ltxdoc}
3 \GetFileInfo{fontdef.dtx}
4 \begin{document}
5   \DocInput{fontdef.dtx}
6 \end{document}
7 </driver>
```

5 The fonttext.ltx file

The identification is done earlier on with a \ProvidesFile declaration.

```
8 <*text>
9 \typeout{=== Don't modify this file, use a .cfg file instead ===^~J}
```

5.1 Encodings

This file declares the standard encodings for text and math fonts. All others should be declared in packages or in the documents directly.

For every text encoding there are normally a number of encoding specific commands, e.g. accents, special characters, etc. (The definition for such a command might have to change when the encoding is changed, because the character is in a different position, or not available at all, or the accent is produced in a different way.) This is handled by a general mechanism which is described in `ltoutenc.dtx`.

By convention, text encoding specific declarations, including the `\DeclareFontEncoding` declaration, are kept in separate file of the form `<enc>enc.def`, e.g. `ot1enc.def`. This allows other applications to make use of the declarations as well.

Similar to the default encoding, the loading of the encoding files for the two major text encodings shouldn't be changed. In particular, the `inputenc` package depends on this.

```
10 \input {omlenc.def}
11 \input {omsenc.def}
```

Documents containing a lot of accented characters should really be using T1 fonts. We therefore load this last so that T1 encoding specific commands are executed as fast as possible (encoding files are no longer reloaded in `fontenc`).

```
12 \input {ot1enc.def}
13 \input {t1enc.def}
14 \input{ts1enc.def}

15 \ifx\Umathcode\@undefined
```

We then set the default text font encoding. This will hopefully change some day to T1. This setting should *not* be changed to produce a portable format.

```
16 \fontencoding{OT1}

The initial fontenc package load list if an 8-bit TEX engine is used:
17 \def\@fontenc@load@list{\@elt{T1,OT1}}
18 \def\rmsubstdefault{cmr}
19 \def\sfsubstdefault{cmss}
20 \def\ttsubstdefault{cmtt}
21 \LoadFontDefinitionFile{TS1}{cmr}

22 \else
```

Unicode.

```
23 \input {tuenc.def}
24 \fontencoding{TU}

The initial fontenc package load list if a Unicode engine is used:
25 \def\@fontenc@load@list{\@elt{TU}}
26 \DeclareFontSubstitution{TU}{lmr}{m}{n}
27 \LoadFontDefinitionFile{TU}{lmr}
28 \LoadFontDefinitionFile{TU}{lmss}
29 \LoadFontDefinitionFile{TU}{lmtt}

30 \def\rmsubstdefault{lmr}
31 \def\sfsubstdefault{lmss}
32 \def\ttsubstdefault{lmtt}
33 \LoadFontDefinitionFile{TS1}{lmr}
```

```
34 \DeclareFontSubstitution{TU}{lmr}{m}{n}
```

End of Unicode branch.

```
35 \fi
```

If different encodings for text fonts are in use one could put the common setup into `\DeclareFontEncodingDefaults`. There is now a better mechanism so using this interface is discouraged!

```
36 \DeclareFontEncodingDefaults{}{}
```

Then we define the default substitution for every encoding. This release of $\text{\LaTeX 2}_{\epsilon}$ assumes that the ec fonts are available. It is possible to change this to point to some other font family (e.g., Times with the appropriate encoding if it is available) without making documents non-portable. However, in such a case documents will produce different page breaks at other sites. The substitution defaults can all be changed without losing portability as long as there are font shape definitions for the selected substitutions.

```
37 \DeclareFontSubstitution{T1}{cmr}{m}{n}
```

```
38 \DeclareFontSubstitution{OT1}{cmr}{m}{n}
```

For every encoding declaration, $\text{\LaTeX 2}_{\epsilon}$ will try to verify that the given substitution information makes sense, i.e. that it is impossible to go into an endless loop if font substitution happens. This is done at the moment the `\begin{document}` is encountered. $\text{\LaTeX 2}_{\epsilon}$ will then check that for every encoding the substitution defaults form a valid font shape group, which means that it will check if there is a `\DeclareFontShape` declaration for this combination. We will therefore load the corresponding `.fd` files now. If we don't do this they would be loaded at verification time (i.e. at `\begin{document}`) which would delay processing unnecessarily.

Warning: Please note that this means that you have to regenerate the format whenever you change any of these `.fd` files since $\text{\LaTeX 2}_{\epsilon}$ will not read `.fd` files if it already knows about the encoding/family combination.

The `\nfss@catcodes` ensures that white space is ignored in any definitions made in the fd files.

```
39 \begingroup
```

```
40 \nfss@catcodes
```

```
41 \input {t1cmr.fd}
```

```
42 \input {ot1cmr.fd}
```

```
43 \endgroup
```

We also load some other font definition files which are normally needed in a document. This is only done for processing speed and you can comment the next two lines out to save some memory. If necessary these files are then loaded when your document is processed. (Loading `.fd` files is a less drastic step compared to preloading fonts because the number of fonts is limited 255 at (nearly) every \TeX installation, while the amount of main memory is not a limiting factor at most installations.)

```
44 \begingroup
```

```
45 \nfss@catcodes
```

```
46 \input {ot1cmss.fd}
```

```
47 \input {ot1cmtt.fd}
```

```
48 \endgroup
```

Even with all the precautions it is still possible that NFSS will run into problems, for example, when a .fd file contains corrupted data. To guard against such cases NFSS has a very low-level fallback font that is installed with the following line.

```
49 \DeclareErrorFont{OT1}{cmr}{m}{n}{10}
```

This means, “if everything else fails use Computer Modern Roman normal shape at 10pt in the old text encoding”. You can change the font used but the encoding should be the same as the one specified with `\fontencoding` above.

5.2 Defaults

To allow the use of `\rmfamily`, `\sffamily`, etc. in documents even if non-standard families are used we provide nine macros which hold the name of the corresponding families, series, and so on. This makes it easy to use other font families (like Times Roman, etc.). One simply has to redefine these defaults.

All these hooks have to be defined in this file but you can change their meaning (except for `\encodingdefault`) without making documents non-portable.

```
\encodingdefault The following three definitions set up the meaning for \rmfamily, \sffamily, and
\rmdefault      \ttfamily.
\sfdefault
\ttdefault
50 \ifx\Umathcode\@undefined
51 \newcommand\encodingdefault{OT1}
52 \newcommand\rmdefault{cmr}
53 \newcommand\sfdefault{cmss}
54 \newcommand\ttdefault{cmtt}
55 \else
56 \newcommand\encodingdefault{TU}
57 \newcommand\rmdefault{lmr}
58 \fontfamily{\rmdefault}
59 \newcommand\sfdefault{lmss}
60 \newcommand\ttdefault{lmtt}
61 \fi
62 </text>
63 <latexrelease>\IncludeInRelease{2017/01/01}%
64 <latexrelease>          {\encodingdefault}{TU encoding default}%
65 <latexrelease>\ifx\Umathcode\@undefined
66 <latexrelease>\renewcommand\encodingdefault{OT1}
67 <latexrelease>\fontencoding{\encodingdefault}
68 <latexrelease>\renewcommand\rmdefault{cmr}
69 <latexrelease>\fontfamily{\rmdefault}
70 <latexrelease>\renewcommand\sfdefault{cmss}
71 <latexrelease>\renewcommand\ttdefault{cmtt}
72 <latexrelease>\else
73 <latexrelease>\renewcommand\encodingdefault{TU}
74 <latexrelease>%done in everyjob\fontencoding{\encodingdefault}
75 <latexrelease>\renewcommand\rmdefault{lmr}
76 <latexrelease>\fontfamily{\rmdefault}
77 <latexrelease>\renewcommand\sfdefault{lmss}
78 <latexrelease>\renewcommand\ttdefault{lmtt}
79 <latexrelease>\fi
80 <latexrelease>\EndIncludeInRelease
81 <latexrelease>\IncludeInRelease{0000/00/00}%
82 <latexrelease>          {\encodingdefault}{TU encoding default}%
```

```

83 <latexrelease>\fontencoding{OT1}
84 <latexrelease>\renewcommand\encodingdefault{OT1}
85 <latexrelease>\fontencoding{\encodingdefault}
86 <latexrelease>\renewcommand\rmdefault{cmr}
87 <latexrelease>\fontfamily{\rmdefault}
88 <latexrelease>\renewcommand\sfddefault{cmss}
89 <latexrelease>\renewcommand\ttdefault{cmtt}
90 <latexrelease>\EndIncludeInRelease
91 <*text>

```

(End definition for \encodingdefault and others.)

```

\bfdefault Series changing commands are influenced by the following hooks.
\mddefault 92 \newcommand\bfdefault{b} % overwritten below (for rollback)
          93 \newcommand\mddefault{m} % overwritten below (for rollback)

```

(End definition for \bfdefault and \mddefault.)

```

\itdefault Shape changing commands use the following hooks.
\sldefault 94 \newcommand\itdefault{it}
\scdefault 95 \newcommand\sldefault{sl}
\updefault 96 \newcommand\scdefault{sc}
          97 \newcommand\updefault{up} % overwritten below (for rollback)

```

(End definition for \itdefault and others.)

```

98 </text>
99 <*text | latexrelease>
100 <latexrelease>\IncludeInRelease{2020/02/02}%
101 <latexrelease>          {\updefault}{font defaults change}%
102 % \begin{macrocode}
103 \renewcommand\updefault{up}

```

We append \@empty to the series value so that we can detect if it got changed via \def or \renewcommand later.

```

104 \renewcommand\bfdefault{b\@empty}
105 \renewcommand\mddefault{m\@empty}

106 \let\bfdefault@previous\bfdefault
107 \let\mddefault@previous\mddefault
108 </text | latexrelease>
109 <latexrelease>\EndIncludeInRelease
110 <latexrelease>\IncludeInRelease{0000/00/00}%
111 <latexrelease>          {\updefault}{font defaults change}%
112 <latexrelease>
113 <latexrelease>\renewcommand\updefault{n}
114 <latexrelease>\renewcommand\bfdefault{bx}
115 <latexrelease>
116 <latexrelease>\let\bfdefault@previous\undefined
117 <latexrelease>\let\mddefault@previous\undefined
118 <latexrelease>\EndIncludeInRelease
119 <*text>

```

`\familydefault` Finally we have the hooks that describe the behaviour of the `\normalfont` command.
`\seriesdefault` To stay portable, the definition of `\encodingdefault` should *not* be changed and should
`\shapedefault` match the setting above for `\fontencoding`. All other values can be set according to your taste.

```
120 \newcommand\familydefault{\rmdefault}
121 \newcommand\seriesdefault{\mddefault}
```

In previous releases `\shapedefault` pointed to `\updefault` which resolved to `n`, but these days that is no longer the case (and `up` is wrong when you want to do a reset. So we now use `n` explicitly.

```
122 \newcommand\shapedefault{n}
```

(End definition for `\familydefault`, `\seriesdefault`, and `\shapedefault`.)

This finishes the low-level setup in `fonttext.ltx`.

```
123 </text>
```

6 The fontmath.ltx file

The identification is done earlier on with a `\ProvidesFile` declaration.

```
124 <*math>
125 \typeout{=== Don't modify this file, use a .cfg file instead ===^~J}
```

6.1 The font encodings used

```
126 \DeclareFontEncoding{OML}{-}{-}
127 \DeclareFontEncoding{OMS}{-}{-}
128 \DeclareFontEncoding{OMX}{-}{-}
```

Finally a declaration for U encoding which serves for all fonts that do not fit standard encodings. For math this sets up `\noaccents@` providing for AMS- \LaTeX . This macro is used therein to handle accented characters if they are not supported by the font. In other words, if fonts with U encoding are used in math, all accents (like from `\breve`) are obtained from some other font that has them.

```
129 \DeclareFontEncoding{U}{-}{\noaccents@}
```

The encodings for math are next:

```
130 \DeclareFontSubstitution{OML}{cmm}{m}{it}
131 \DeclareFontSubstitution{OMS}{cmsy}{m}{n}
132 \DeclareFontSubstitution{OMX}{cmex}{m}{n}
133 \DeclareFontSubstitution{U}{cmr}{m}{n}
134 \begingroup
135 \nfss@catcodes
136 \input {omlcmm.fd}
137 \input {omscmsy.fd}
138 \input {omxcmex.fd}
139 \input {ucmr.fd}
140 \endgroup
```

6.1.1 Symbolfont and Alphabet declarations

We now define the basic symbol fonts used by \LaTeX . These four symbol fonts must be defined by this file.

It is possible to make the symbol fonts point to other external fonts without losing the ability to process documents written at other sites, as long as one defines the same

symbol font names with the same encodings, e.g. `operators` with OT1 etc. If other encodings are used documents become non-portable. Such a change should therefore be done in a package file.

```

141 \DeclareSymbolFont{operators}    {OT1}{cmr} {m}{n}
142 \DeclareSymbolFont{letters}     {OML}{cmm} {m}{it}
143 \DeclareSymbolFont{symbols}     {OMS}{cmsy}{m}{n}
144 \DeclareSymbolFont{largesymbols}{OMX}{cmex}{m}{n}

145 \SetSymbolFont{operators}{bold}{OT1}{cmr} {bx}{n}
146 \SetSymbolFont{letters}  {bold}{OML}{cmm} {b}{it}
147 \SetSymbolFont{symbols}  {bold}{OMS}{cmsy}{b}{n}

```

Below are the seven math alphabets which are defined by NFSS. Again they must be defined by this file. However, as before you can change the fonts used without losing portability, but you should be careful when changing the encoding since that may make documents come out wrong.

```

148 \DeclareSymbolFontAlphabet{\mathrm}    {operators}
149 \DeclareSymbolFontAlphabet{\mathnormal}{letters}
150 \DeclareSymbolFontAlphabet{\mathcal}    {symbols}
151 \DeclareMathAlphabet          {\mathbf}  {OT1}{cmr}{bx}{n}
152 \DeclareMathAlphabet          {\mathsf}  {OT1}{cmss}{m}{n}
153 \DeclareMathAlphabet          {\mathit}  {OT1}{cmr}{m}{it}
154 \DeclareMathAlphabet          {\mathtt}   {OT1}{cmtt}{m}{n}

```

Given the currently available fonts we cannot bold-en `\mathbf` and `\mathtt` but in principle one could use ‘ultra bold’ or something. The alphabets defined via `\DeclareSymbolFontAlphabet` will change automatically in a new math version if the corresponding symbol font changes.

```

155 \SetMathAlphabet\mathsf{bold}{OT1}{cmss}{bx}{n}
156 \SetMathAlphabet\mathit{bold}{OT1}{cmr}{bx}{it}

```

6.2 Math font sizes

The declarations below declare the text, script and scriptscript size to be used for each text font size.

All occurrences of sizes longer than a single character are replaced with the macro name that holds them, saving a number of tokens (but losing a bit of speed, so this may not stay this way).

```

157 \DeclareMathSizes{5}{5}{5}{5}
158 \DeclareMathSizes{6}{6}{5}{5}
159 \DeclareMathSizes{7}{7}{5}{5}
160 \DeclareMathSizes{8}{8}{6}{5}
161 \DeclareMathSizes{9}{9}{6}{5}
162 \DeclareMathSizes{\@xpt}{\@xpt}{7}{5}
163 \DeclareMathSizes{\@xipt}{\@xipt}{8}{6}
164 \DeclareMathSizes{\@xiipt}{\@xiipt}{8}{6}
165 \DeclareMathSizes{\@xivpt}{\@xivpt}{\@xpt}{7}
166 \DeclareMathSizes{\@xvipt}{\@xvipt}{\@xipt}{\@xpt}
167 \DeclareMathSizes{\@xxpt}{\@xxpt}{\@xivpt}{\@xiipt}
168 \DeclareMathSizes{\@xxvpt}{\@xxvpt}{\@xxpt}{\@xvipt}

```


6.3 The math symbol assignments

We start by setting up math codes for most of the characters typed in directly from the keyboard. Most of them are normally already setup up in the same way by `IniTeX`. However, we repeat them here to have a complete setup which can be exchanged with another if desired.

6.3.1 The letters

```
169 \DeclareMathSymbol{a}{\mathalpha}{letters}{'a}
170 \DeclareMathSymbol{b}{\mathalpha}{letters}{'b}
171 \DeclareMathSymbol{c}{\mathalpha}{letters}{'c}
172 \DeclareMathSymbol{d}{\mathalpha}{letters}{'d}
173 \DeclareMathSymbol{e}{\mathalpha}{letters}{'e}
174 \DeclareMathSymbol{f}{\mathalpha}{letters}{'f}
175 \DeclareMathSymbol{g}{\mathalpha}{letters}{'g}
176 \DeclareMathSymbol{h}{\mathalpha}{letters}{'h}
177 \DeclareMathSymbol{i}{\mathalpha}{letters}{'i}
178 \DeclareMathSymbol{j}{\mathalpha}{letters}{'j}
179 \DeclareMathSymbol{k}{\mathalpha}{letters}{'k}
180 \DeclareMathSymbol{l}{\mathalpha}{letters}{'l}
181 \DeclareMathSymbol{m}{\mathalpha}{letters}{'m}
182 \DeclareMathSymbol{n}{\mathalpha}{letters}{'n}
183 \DeclareMathSymbol{o}{\mathalpha}{letters}{'o}
184 \DeclareMathSymbol{p}{\mathalpha}{letters}{'p}
185 \DeclareMathSymbol{q}{\mathalpha}{letters}{'q}
186 \DeclareMathSymbol{r}{\mathalpha}{letters}{'r}
187 \DeclareMathSymbol{s}{\mathalpha}{letters}{'s}
188 \DeclareMathSymbol{t}{\mathalpha}{letters}{'t}
189 \DeclareMathSymbol{u}{\mathalpha}{letters}{'u}
190 \DeclareMathSymbol{v}{\mathalpha}{letters}{'v}
191 \DeclareMathSymbol{w}{\mathalpha}{letters}{'w}
192 \DeclareMathSymbol{x}{\mathalpha}{letters}{'x}
193 \DeclareMathSymbol{y}{\mathalpha}{letters}{'y}
194 \DeclareMathSymbol{z}{\mathalpha}{letters}{'z}

195 \DeclareMathSymbol{A}{\mathalpha}{letters}{'A}
196 \DeclareMathSymbol{B}{\mathalpha}{letters}{'B}
197 \DeclareMathSymbol{C}{\mathalpha}{letters}{'C}
198 \DeclareMathSymbol{D}{\mathalpha}{letters}{'D}
199 \DeclareMathSymbol{E}{\mathalpha}{letters}{'E}
200 \DeclareMathSymbol{F}{\mathalpha}{letters}{'F}
201 \DeclareMathSymbol{G}{\mathalpha}{letters}{'G}
202 \DeclareMathSymbol{H}{\mathalpha}{letters}{'H}
203 \DeclareMathSymbol{I}{\mathalpha}{letters}{'I}
204 \DeclareMathSymbol{J}{\mathalpha}{letters}{'J}
205 \DeclareMathSymbol{K}{\mathalpha}{letters}{'K}
206 \DeclareMathSymbol{L}{\mathalpha}{letters}{'L}
207 \DeclareMathSymbol{M}{\mathalpha}{letters}{'M}
208 \DeclareMathSymbol{N}{\mathalpha}{letters}{'N}
209 \DeclareMathSymbol{O}{\mathalpha}{letters}{'O}
210 \DeclareMathSymbol{P}{\mathalpha}{letters}{'P}
211 \DeclareMathSymbol{Q}{\mathalpha}{letters}{'Q}
212 \DeclareMathSymbol{R}{\mathalpha}{letters}{'R}
213 \DeclareMathSymbol{S}{\mathalpha}{letters}{'S}
```

```

214 \DeclareMathSymbol{T}{\mathalpha}{letters}{‘T}
215 \DeclareMathSymbol{U}{\mathalpha}{letters}{‘U}
216 \DeclareMathSymbol{V}{\mathalpha}{letters}{‘V}
217 \DeclareMathSymbol{W}{\mathalpha}{letters}{‘W}
218 \DeclareMathSymbol{X}{\mathalpha}{letters}{‘X}
219 \DeclareMathSymbol{Y}{\mathalpha}{letters}{‘Y}
220 \DeclareMathSymbol{Z}{\mathalpha}{letters}{‘Z}

```

6.3.2 The digits

```

221 \DeclareMathSymbol{0}{\mathalpha}{operators}{‘0}
222 \DeclareMathSymbol{1}{\mathalpha}{operators}{‘1}
223 \DeclareMathSymbol{2}{\mathalpha}{operators}{‘2}
224 \DeclareMathSymbol{3}{\mathalpha}{operators}{‘3}
225 \DeclareMathSymbol{4}{\mathalpha}{operators}{‘4}
226 \DeclareMathSymbol{5}{\mathalpha}{operators}{‘5}
227 \DeclareMathSymbol{6}{\mathalpha}{operators}{‘6}
228 \DeclareMathSymbol{7}{\mathalpha}{operators}{‘7}
229 \DeclareMathSymbol{8}{\mathalpha}{operators}{‘8}
230 \DeclareMathSymbol{9}{\mathalpha}{operators}{‘9}

```

6.3.3 Punctuation, brace, etc. keys

```

231 \DeclareMathSymbol{!}{\mathclose}{operators}{"21}
232 \DeclareMathSymbol{*}{\mathbin}{symbols}{"03} % \ast
233 \DeclareMathSymbol{+}{\mathbin}{operators}{"2B}
234 \DeclareMathSymbol{,}{\mathpunct}{letters}{"3B}
235 \DeclareMathSymbol{-}{\mathbin}{symbols}{"00}
236 \DeclareMathSymbol{.}{\mathord}{letters}{"3A}
237 \DeclareMathSymbol{:}{\mathrel}{operators}{"3A}
238 \DeclareMathSymbol{;}{\mathpunct}{operators}{"3B}
239 \DeclareMathSymbol{=}{\mathrel}{operators}{"3D}
240 \DeclareMathSymbol{?}{\mathclose}{operators}{"3F}

```

The following symbols are defined as delimiters below which automatically defines them as math symbols.

```

241 %\DeclareMathSymbol{(){\mathopen}{operators}{"28}
242 %\DeclareMathSymbol{)}{\mathclose}{operators}{"29}
243 %\DeclareMathSymbol{/}{\mathord}{letters}{"3D}
244 %\DeclareMathSymbol{[(){\mathopen}{operators}{"5B}
245 %\DeclareMathSymbol{](){\mathclose}{operators}{"5D}
246 %\DeclareMathSymbol{|}{\mathord}{symbols}{"6A}
247 %\DeclareMathSymbol{<}{\mathrel}{letters}{"3C}
248 %\DeclareMathSymbol{>}{\mathrel}{letters}{"3E}

```

Should all of the following being activated by default? Probably not.

```

249 %\DeclareMathSymbol{\{}{\mathopen}{symbols}{"66}
250 %\DeclareMathSymbol{\}}{\mathclose}{symbols}{"67}
251 %\DeclareMathSymbol{\}\}{\mathord}{symbols}{"6E} % \backslash
252 \mathcode‘\ = "8000 % \space
253 \mathcode‘\` = "8000 % ^\prime
254 \mathcode‘\_ = "8000 % \_

```

6.3.4 Delimitercodes for characters

[to be completed]

Finally, `InitEX` sets all `\delcode` values to -1, except `\delcode‘.=0`

```

255 \DeclareMathDelimiter{\mathopen}{operators}{28}{largesymbols}{00}
256 \DeclareMathDelimiter{\mathclose}{operators}{29}{largesymbols}{01}
257 \DeclareMathDelimiter{\mathopen}{operators}{5B}{largesymbols}{02}
258 \DeclareMathDelimiter{\mathclose}{operators}{5D}{largesymbols}{03}

```

The next two are considered to be relations when not used in the context of a delimiter! And worse, they do even represent different glyphs when being used as delimiter and not as delimiter. This is a user level syntax inherited from plain T_EX. Therefore we explicitly redefine the math symbol definitions for these symbols afterwards.

```

259 \DeclareMathDelimiter{<}{\mathopen}{symbols}{68}{largesymbols}{0A}
260 \DeclareMathDelimiter{>}{\mathclose}{symbols}{69}{largesymbols}{0B}
261 \DeclareMathSymbol{<}{\mathrel}{letters}{3C}
262 \DeclareMathSymbol{>}{\mathrel}{letters}{3E}

```

And here is another case where the non-delimiter version produces a glyph different from the delimiter version.

```

263 \DeclareMathDelimiter{/}{\mathord}{operators}{2F}{largesymbols}{0E}
264 \DeclareMathSymbol{/}{\mathord}{letters}{3D}

265 \DeclareMathDelimiter{|}{\mathord}{symbols}{6A}{largesymbols}{0C}

266 \expandafter\DeclareMathDelimiter\@backslashchar
267 \mathord{symbols}{6E}{largesymbols}{0F}

```

N.B. { and } should NOT get delcodes; otherwise parameter grouping fails!

6.4 Symbols accessed via control sequences

6.4.1 Greek letters

```

268 \DeclareMathSymbol{\alpha}{\mathord}{letters}{0B}
269 \DeclareMathSymbol{\beta}{\mathord}{letters}{0C}
270 \DeclareMathSymbol{\gamma}{\mathord}{letters}{0D}
271 \DeclareMathSymbol{\delta}{\mathord}{letters}{0E}
272 \DeclareMathSymbol{\epsilon}{\mathord}{letters}{0F}
273 \DeclareMathSymbol{\zeta}{\mathord}{letters}{10}
274 \DeclareMathSymbol{\eta}{\mathord}{letters}{11}
275 \DeclareMathSymbol{\theta}{\mathord}{letters}{12}
276 \DeclareMathSymbol{\iota}{\mathord}{letters}{13}
277 \DeclareMathSymbol{\kappa}{\mathord}{letters}{14}
278 \DeclareMathSymbol{\lambda}{\mathord}{letters}{15}
279 \DeclareMathSymbol{\mu}{\mathord}{letters}{16}
280 \DeclareMathSymbol{\nu}{\mathord}{letters}{17}
281 \DeclareMathSymbol{\xi}{\mathord}{letters}{18}
282 \DeclareMathSymbol{\pi}{\mathord}{letters}{19}
283 \DeclareMathSymbol{\rho}{\mathord}{letters}{1A}
284 \DeclareMathSymbol{\sigma}{\mathord}{letters}{1B}
285 \DeclareMathSymbol{\tau}{\mathord}{letters}{1C}
286 \DeclareMathSymbol{\upsilon}{\mathord}{letters}{1D}
287 \DeclareMathSymbol{\phi}{\mathord}{letters}{1E}
288 \DeclareMathSymbol{\chi}{\mathord}{letters}{1F}
289 \DeclareMathSymbol{\psi}{\mathord}{letters}{20}
290 \DeclareMathSymbol{\omega}{\mathord}{letters}{21}
291 \DeclareMathSymbol{\varepsilon}{\mathord}{letters}{22}
292 \DeclareMathSymbol{\vartheta}{\mathord}{letters}{23}
293 \DeclareMathSymbol{\varpi}{\mathord}{letters}{24}

```

```

294 \DeclareMathSymbol{\varrho}{\mathord}{letters}{25}
295 \DeclareMathSymbol{\varsigma}{\mathord}{letters}{26}
296 \DeclareMathSymbol{\varphi}{\mathord}{letters}{27}
297 \DeclareMathSymbol{\Gamma}{\mathalpha}{operators}{00}
298 \DeclareMathSymbol{\Delta}{\mathalpha}{operators}{01}
299 \DeclareMathSymbol{\Theta}{\mathalpha}{operators}{02}
300 \DeclareMathSymbol{\Lambda}{\mathalpha}{operators}{03}
301 \DeclareMathSymbol{\Xi}{\mathalpha}{operators}{04}
302 \DeclareMathSymbol{\Pi}{\mathalpha}{operators}{05}
303 \DeclareMathSymbol{\Sigma}{\mathalpha}{operators}{06}
304 \DeclareMathSymbol{\Upsilon}{\mathalpha}{operators}{07}
305 \DeclareMathSymbol{\Phi}{\mathalpha}{operators}{08}
306 \DeclareMathSymbol{\Psi}{\mathalpha}{operators}{09}
307 \DeclareMathSymbol{\Omega}{\mathalpha}{operators}{0A}

```

6.4.2 Ordinary symbols

```

308 \DeclareMathSymbol{\aleph}{\mathord}{symbols}{40}
309 \DeclareMathSymbol{\imath}{\mathord}{letters}{7B}
310 \DeclareMathSymbol{\jmath}{\mathord}{letters}{7C}
311 \DeclareMathSymbol{\ell}{\mathord}{letters}{60}
312 \DeclareMathSymbol{\wp}{\mathord}{letters}{7D}
313 \DeclareMathSymbol{\Re}{\mathord}{symbols}{3C}
314 \DeclareMathSymbol{\Im}{\mathord}{symbols}{3D}
315 \DeclareMathSymbol{\partial}{\mathord}{letters}{40}
316 \DeclareMathSymbol{\infty}{\mathord}{symbols}{31}
317 \DeclareMathSymbol{\prime}{\mathord}{symbols}{30}
318 \DeclareMathSymbol{\emptyset}{\mathord}{symbols}{3B}
319 \DeclareMathSymbol{\nabla}{\mathord}{symbols}{72}
320 \DeclareMathSymbol{\top}{\mathord}{symbols}{3E}
321 \DeclareMathSymbol{\bot}{\mathord}{symbols}{3F}
322 \DeclareMathSymbol{\triangle}{\mathord}{symbols}{34}
323 \DeclareMathSymbol{\forall}{\mathord}{symbols}{38}
324 \DeclareMathSymbol{\exists}{\mathord}{symbols}{39}
325 \DeclareMathSymbol{\neg}{\mathord}{symbols}{3A}

```

Alias:

```

326 % \let\lnot=\neg
327 \DeclareMathSymbol{\lnot}{\mathord}{symbols}{3A}
328 \DeclareMathSymbol{\flat}{\mathord}{letters}{5B}
329 \DeclareMathSymbol{\natural}{\mathord}{letters}{5C}
330 \DeclareMathSymbol{\sharp}{\mathord}{letters}{5D}
331 \DeclareMathSymbol{\clubsuit}{\mathord}{symbols}{7C}
332 \DeclareMathSymbol{\diamondsuit}{\mathord}{symbols}{7D}
333 \DeclareMathSymbol{\heartsuit}{\mathord}{symbols}{7E}
334 \DeclareMathSymbol{\spadesuit}{\mathord}{symbols}{7F}

335 \DeclareRobustCommand\hbar{{\mathchar'26\mkern-9mu h}}
336 \DeclareRobustCommand\surd{{\mathchar"1270}}
337 \DeclareRobustCommand\angle{{\vbox{\ialign{$\m@th\scriptstyle##$\crrc
338 \not\mathrel{\mkern14mu}\crrc
339 \noalign{\nointerlineskip}
340 \mkern2.5mu\leaders\hrule \@height.34pt\hfill\mkern2.5mu\crrc}}}}

```

6.4.3 Large Operators

```

341 \DeclareMathSymbol{\coprod}{\mathop}{largesymbols}{60}

```

```

342 \DeclareMathSymbol{\bigvee}{\mathop}{largesymbols}{57}
343 \DeclareMathSymbol{\bigwedge}{\mathop}{largesymbols}{56}
344 \DeclareMathSymbol{\biguplus}{\mathop}{largesymbols}{55}
345 \DeclareMathSymbol{\bigcap}{\mathop}{largesymbols}{54}
346 \DeclareMathSymbol{\bigcup}{\mathop}{largesymbols}{53}
347 \DeclareMathSymbol{\intop}{\mathop}{largesymbols}{52}
348 \DeclareRobustCommand\int{\intop\nolimits}
349 \DeclareMathSymbol{\prod}{\mathop}{largesymbols}{51}
350 \DeclareMathSymbol{\sum}{\mathop}{largesymbols}{50}
351 \DeclareMathSymbol{\bigotimes}{\mathop}{largesymbols}{4E}
352 \DeclareMathSymbol{\bigoplus}{\mathop}{largesymbols}{4C}
353 \DeclareMathSymbol{\bigodot}{\mathop}{largesymbols}{4A}
354 \DeclareMathSymbol{\ointop}{\mathop}{largesymbols}{48}
355 \DeclareRobustCommand\oint{\ointop\nolimits}
356 \DeclareMathSymbol{\bigsqcup}{\mathop}{largesymbols}{46}
357 \DeclareMathSymbol{\smallint}{\mathop}{symbols}{73}

```

6.4.4 Binary symbols

```

358 \DeclareMathSymbol{\triangleleft}{\mathbin}{letters}{2F}
359 \DeclareMathSymbol{\triangleright}{\mathbin}{letters}{2E}
360 \DeclareMathSymbol{\bigtriangleup}{\mathbin}{symbols}{34}
361 \DeclareMathSymbol{\bigtriangledown}{\mathbin}{symbols}{35}

```

Alias:

```

362 % \let \varbigtriangledown \bigtriangledown
363 % \let \varbigtriangleup \bigtriangleup
364 \DeclareMathSymbol{\varbigtriangleup}{\mathbin}{symbols}{34}
365 \DeclareMathSymbol{\varbigtriangledown}{\mathbin}{symbols}{35}

```

These last two synonyms are needed because the `stmaryrd` package redefines them as Operators.

```

366 \DeclareMathSymbol{\wedge}{\mathbin}{symbols}{5E}
367 \DeclareMathSymbol{\vee}{\mathbin}{symbols}{5F}

```

Alias:

```

368 % \let\land=\wedge
369 % \let\lor=\vee
370 \DeclareMathSymbol{\land}{\mathbin}{symbols}{5E}
371 \DeclareMathSymbol{\lor}{\mathbin}{symbols}{5F}
372 \DeclareMathSymbol{\cap}{\mathbin}{symbols}{5C}
373 \DeclareMathSymbol{\cup}{\mathbin}{symbols}{5B}
374 \DeclareMathSymbol{\ddagger}{\mathbin}{symbols}{7A}
375 \DeclareMathSymbol{\dagger}{\mathbin}{symbols}{79}
376 \DeclareMathSymbol{\sqcap}{\mathbin}{symbols}{75}
377 \DeclareMathSymbol{\sqcup}{\mathbin}{symbols}{74}
378 \DeclareMathSymbol{\uplus}{\mathbin}{symbols}{5D}
379 \DeclareMathSymbol{\amalg}{\mathbin}{symbols}{71}
380 \DeclareMathSymbol{\diamond}{\mathbin}{symbols}{05}
381 \DeclareMathSymbol{\bullet}{\mathbin}{symbols}{0F}
382 \DeclareMathSymbol{\wr}{\mathbin}{symbols}{6F}
383 \DeclareMathSymbol{\div}{\mathbin}{symbols}{04}
384 \DeclareMathSymbol{\odot}{\mathbin}{symbols}{0C}
385 \DeclareMathSymbol{\oslash}{\mathbin}{symbols}{0B}
386 \DeclareMathSymbol{\otimes}{\mathbin}{symbols}{0A}
387 \DeclareMathSymbol{\ominus}{\mathbin}{symbols}{09}

```

```

388 \DeclareMathSymbol{\oplus}{\mathbin}{symbols}{"08}
389 \DeclareMathSymbol{\mp}{\mathbin}{symbols}{"07}
390 \DeclareMathSymbol{\pm}{\mathbin}{symbols}{"06}
391 \DeclareMathSymbol{\circ}{\mathbin}{symbols}{"0E}
392 \DeclareMathSymbol{\bigcirc}{\mathbin}{symbols}{"0D}
393 \DeclareMathSymbol{\setminus}{\mathbin}{symbols}{"6E}
394 \DeclareMathSymbol{\cdot}{\mathbin}{symbols}{"01}
395 \DeclareMathSymbol{\ast}{\mathbin}{symbols}{"03}
396 \DeclareMathSymbol{\times}{\mathbin}{symbols}{"02}
397 \DeclareMathSymbol{\star}{\mathbin}{letters}{"3F}

```

6.4.5 Relations

```

398 \DeclareMathSymbol{\propto}{\mathrel}{symbols}{"2F}
399 \DeclareMathSymbol{\sqsubseteq}{\mathrel}{symbols}{"76}
400 \DeclareMathSymbol{\sqsupseteq}{\mathrel}{symbols}{"77}
401 \DeclareMathSymbol{\parallel}{\mathrel}{symbols}{"6B}
402 \DeclareMathSymbol{\mid}{\mathrel}{symbols}{"6A}
403 \DeclareMathSymbol{\dashv}{\mathrel}{symbols}{"61}
404 \DeclareMathSymbol{\vdash}{\mathrel}{symbols}{"60}
405 \DeclareMathSymbol{\nearrow}{\mathrel}{symbols}{"25}
406 \DeclareMathSymbol{\searrow}{\mathrel}{symbols}{"26}
407 \DeclareMathSymbol{\nwarrow}{\mathrel}{symbols}{"2D}
408 \DeclareMathSymbol{\swarrow}{\mathrel}{symbols}{"2E}
409 \DeclareMathSymbol{\Leftrightarrow}{\mathrel}{symbols}{"2C}
410 \DeclareMathSymbol{\Leftarrow}{\mathrel}{symbols}{"28}
411 \DeclareMathSymbol{\Rightarrow}{\mathrel}{symbols}{"29}
412 \DeclareRobustCommand\neq{\not=}

```

As `\neq` is robust we should not use `\let` to define `\ne` as then it would change if `\neq` changes.

```

413 \DeclareRobustCommand\ne{\not=}

```

It would ok to use `\let` for those declared by `\DeclareMathSymbol` but for a cleaner interface we avoid it always (just in case the internals change).

```

414 \DeclareMathSymbol{\leq}{\mathrel}{symbols}{"14}
415 \DeclareMathSymbol{\geq}{\mathrel}{symbols}{"15}

```

Alias:

```

416 % \let\le=\leq
417 % \let\ge=\geq
418 \DeclareMathSymbol{\le}{\mathrel}{symbols}{"14}
419 \DeclareMathSymbol{\ge}{\mathrel}{symbols}{"15}
420 \DeclareMathSymbol{\succ}{\mathrel}{symbols}{"1F}
421 \DeclareMathSymbol{\prec}{\mathrel}{symbols}{"1E}
422 \DeclareMathSymbol{\approx}{\mathrel}{symbols}{"19}
423 \DeclareMathSymbol{\succeq}{\mathrel}{symbols}{"17}
424 \DeclareMathSymbol{\preceq}{\mathrel}{symbols}{"16}
425 \DeclareMathSymbol{\supset}{\mathrel}{symbols}{"1B}
426 \DeclareMathSymbol{\subset}{\mathrel}{symbols}{"1A}
427 \DeclareMathSymbol{\supseteq}{\mathrel}{symbols}{"13}
428 \DeclareMathSymbol{\subseteq}{\mathrel}{symbols}{"12}
429 \DeclareMathSymbol{\in}{\mathrel}{symbols}{"32}
430 \DeclareMathSymbol{\ni}{\mathrel}{symbols}{"33}

```

Alias:

```

431 % \let\owns=\ni

```

```

432 \DeclareMathSymbol{\owns}{\mathrel}{symbols}{"33}
433 \DeclareMathSymbol{\gg}{\mathrel}{symbols}{"1D}
434 \DeclareMathSymbol{\ll}{\mathrel}{symbols}{"1C}
435 \DeclareMathSymbol{\not}{\mathrel}{symbols}{"36}
436 \DeclareMathSymbol{\leftrightharpoonup}{\mathrel}{symbols}{"24}
437 \DeclareMathSymbol{\leftarrow}{\mathrel}{symbols}{"20}
438 \DeclareMathSymbol{\rightarrow}{\mathrel}{symbols}{"21}

```

Alias:

```

439 % \let\gets=\leftarrow
440 % \let\to=\rightarrow
441 \DeclareMathSymbol{\gets}{\mathrel}{symbols}{"20}
442 \DeclareMathSymbol{\to}{\mathrel}{symbols}{"21}
443 \DeclareMathSymbol{\mapstochar}{\mathrel}{symbols}{"37}
444 \DeclareRobustCommand\mapsto{\mapstochar\rightarrow}
445 \DeclareMathSymbol{\sim}{\mathrel}{symbols}{"18}
446 \DeclareMathSymbol{\simeq}{\mathrel}{symbols}{"27}
447 \DeclareMathSymbol{\perp}{\mathrel}{symbols}{"3F}
448 \DeclareMathSymbol{\equiv}{\mathrel}{symbols}{"11}
449 \DeclareMathSymbol{\asymp}{\mathrel}{symbols}{"10}
450 \DeclareMathSymbol{\smile}{\mathrel}{letters}{"5E}
451 \DeclareMathSymbol{\frown}{\mathrel}{letters}{"5F}
452 \DeclareMathSymbol{\leftharpoonup}{\mathrel}{letters}{"28}
453 \DeclareMathSymbol{\leftharpoondown}{\mathrel}{letters}{"29}
454 \DeclareMathSymbol{\rightharpoonup}{\mathrel}{letters}{"2A}
455 \DeclareMathSymbol{\rightharpoondown}{\mathrel}{letters}{"2B}

```

Here cometh much profligate robustification of math constructs. Warning: some of these commands may become non-robust if an AMS package is loaded.

Further potential problems: some math font packages may make unfortunate assumptions about some of these definitions that are not true of the robust versions we need.

```

456 \DeclareRobustCommand
457 \cong{\mathrel{\mathpalette@vereq\sim}} % congruence sign
458 \def@vereq#1#2{\lower.5p@\vbox{\lineskiplimit\maxdimen\lineskip-.5\p@
459 \ialign{${\m@th#1\hfil##\hfil$\crrc#2\crrc=\crrc}}}
460 \DeclareRobustCommand
461 \notin{\mathrel{\m@th\mathpalette@cncel\in}}
462 \def@cncel#1#2{\m@th\oalign{${\hfil#1\mkern1mu/\hfil$\crrc#1#2$}}
463 \DeclareRobustCommand
464 \rightleftharpoons{\mathrel{\mathpalette\rlh@{}}}
465 \def\rlh@#1{\vcenter{\m@th\hbox{\oalign{\raise2pt
466 \hbox{${\hfil#1\rightarpoonup$}\crrc
467 $#1\leftharpoondown$}}}}
468 \DeclareRobustCommand
469 \doteq{\buildrel\textstyle.\over=}

```

6.4.6 Arrows

```

470 \DeclareRobustCommand
471 \joinrel{\mathrel{\mkern-3mu}}
472 \DeclareRobustCommand
473 \relbar{\mathrel{\smash-}} % \smash, because -
474 % has the same height as +

```

In contrast to `plain.tex` `\Relbar` got braces around the equal sign to guard against it being “math active” expanding to `\futurelet....`. This might be the case when packages are implementing shorthands for math, e.g. `=>` meaning `\Rightarrow` etc. It would actually be better not to use `=` in such definitions but instead define something like `\mathequalsign` and use this. However we can’t do this now as it would break other math layouts where characters are in different places (since those wouldn’t know about the need for a new command name).

```

475 \DeclareRobustCommand
476   \Relbar{\mathrel{=}}
477 \DeclareMathSymbol{\lhook}{\mathrel}{letters}{"2C}
478   \DeclareRobustCommand\hookrightarrow{\lhook\joinrel\rightarrow}
479 \DeclareMathSymbol{\rhook}{\mathrel}{letters}{"2D}
480   \DeclareRobustCommand\hookleftarrow{\leftarrow\joinrel\rhook}
481 \DeclareRobustCommand
482   \bowtie{\mathrel{\triangleright\joinrel\mathrel{\triangleleft}}}
483 \DeclareRobustCommand
484   \models{\mathrel{|}\joinrel\Relbar}
485 \DeclareRobustCommand
486   \Longrightarrow{\Relbar\joinrel\Rightarrow}

```

LaTeX Change: `\longrightarrow` and `\longleftarrow` redefined to make them robust.

```

487 \DeclareRobustCommand\longrightarrow
488   {\relbar\joinrel\rightarrow}
489 \DeclareRobustCommand\longleftarrow
490   {\leftarrow\joinrel\relbar}
491 \DeclareRobustCommand
492   \Longleftarrow{\Leftarrow\joinrel\Relbar}
493 \DeclareRobustCommand
494   \longmapsto{\mapstochar\longrightarrow}
495 \DeclareRobustCommand
496   \longlefttrightarrow{\leftarrow\joinrel\rightarrow}
497 \DeclareRobustCommand
498   \Longlefttrightarrow{\Leftarrow\joinrel\Rightarrow}
499 \DeclareRobustCommand
500   \iff{\;\Longlefttrightarrow\;}

```

6.4.7 Punctuation symbols

```

501 \DeclareMathSymbol{\ldotp}{\mathpunct}{letters}{"3A}
502 \DeclareMathSymbol{\cdotp}{\mathpunct}{symbols}{"01}
503 \DeclareMathSymbol{\colon}{\mathpunct}{operators}{"3A}

```

This is commented out, since `\ldots` is now defined in `loutenc.dtx`.

```

504 %\def\ldots{\mathinner{\ldotp\ldotp\ldotp}}
505 %\DeclareRobustCommand\ldots
506 %   {\relax\ifmmode\ldots\else\mbox{$\m@th\ldots$}\fi}
507 \DeclareRobustCommand
508   \cdots{\mathinner{\cdotp\cdotp\cdotp}}
509 \DeclareRobustCommand
510   \vdots{\vbox{\baselineskip4p@ \lineskiplimit\z@
511     \kern6p@\hbox{.}\hbox{.}\hbox{.}}}
512 \DeclareRobustCommand
513   \ddots{\mathinner{\mkern1mu\raise7p@
514     \vbox{\kern7p@\hbox{.}}\mkern2mu}

```



```
515 \raise4\p@\hbox{.}\mkern2mu\raise\p@\hbox{.}\mkern1mu}}
```

6.4.8 Math accents

```
516 \DeclareMathAccent{\acute}{\mathalpha}{operators}{13}
517 \DeclareMathAccent{\grave}{\mathalpha}{operators}{12}
518 \DeclareMathAccent{\ddot}{\mathalpha}{operators}{7F}
519 \DeclareMathAccent{\tilde}{\mathalpha}{operators}{7E}
520 \DeclareMathAccent{\bar}{\mathalpha}{operators}{16}
521 \DeclareMathAccent{\breve}{\mathalpha}{operators}{15}
522 \DeclareMathAccent{\check}{\mathalpha}{operators}{14}
523 \DeclareMathAccent{\hat}{\mathalpha}{operators}{5E}
524 \DeclareMathAccent{\vec}{\mathord}{letters}{7E}
525 \DeclareMathAccent{\dot}{\mathalpha}{operators}{5F}
526 \DeclareMathAccent{\widetilde}{\mathord}{largesymbols}{65}
527 \DeclareMathAccent{\widehat}{\mathord}{largesymbols}{62}
```

For some reason plain T_EX never bothered to provide a ring accent in math (although it is available in the fonts), but since we got a request for it here we go:

```
528 \DeclareMathAccent{\mathring}{\mathalpha}{operators}{17}
```

6.4.9 Radicals

```
529 \DeclareMathRadical{\sqrtsign}{symbols}{70}{largesymbols}{70}
```

6.4.10 Over and under something, etc

```
530 \DeclareRobustCommand\overrightarrow[1]{\vbox{\m@th\ialign{##\crrc
531 \rightarrowfill\crrc\noalign{\kern-\p@\nointerlineskip}
532 $\hfil\displaystyle{#1}\hfil$\crrc}}}
533 \DeclareRobustCommand\overleftarrow[1]{\vbox{\m@th\ialign{##\crrc
534 \leftarrowfill\crrc\noalign{\kern-\p@\nointerlineskip}%
535 $\hfil\displaystyle{#1}\hfil$\crrc}}}
536 \DeclareRobustCommand\overbrace[1]
537 {\mathop{\vbox{\m@th\ialign{##\crrc\noalign{\kern3\p@}%
538 \downbracefill\crrc\noalign{\kern3\p@\nointerlineskip}%
539 $\hfil\displaystyle{#1}\hfil$\crrc}}}\limits}
540 \DeclareRobustCommand\underbrace[1]{\mathop{\vtop{\m@th\ialign{##\crrc
541 $\hfil\displaystyle{#1}\hfil$\crrc
542 \noalign{\kern3\p@\nointerlineskip}%
543 \upbracefill\crrc\noalign{\kern3\p@}}}\limits}
```

(quite a waste of tokens, IMHO — Frank)

```
544 \DeclareRobustCommand\skew[3]
545 {{\muskip\z@#1mu\divide\muskip\z@\tw@ \mkern\muskip\z@
546 #2{\mkern-\muskip\z@#3}\mkern\muskip\z@}\mkern-\muskip\z@}{}}
547 \DeclareRobustCommand\rightarrowfill{${\m@th\smash-\mkern-7mu%
548 \cleaders\hbox{${\mkern-2mu\smash-\mkern-2mu$}\hfill
549 \mkern-7mu\mathord\rightarrow$}
550 \DeclareRobustCommand\leftarrowfill{${\m@th\mathord\leftarrow\mkern-7mu%
551 \cleaders\hbox{${\mkern-2mu\smash-\mkern-2mu$}\hfill
552 \mkern-7mu\smash-$}
553 \DeclareMathSymbol{\braceld}{\mathord}{largesymbols}{7A}
554 \DeclareMathSymbol{\bracerd}{\mathord}{largesymbols}{7B}
555 \DeclareMathSymbol{\bracelu}{\mathord}{largesymbols}{7C}
556 \DeclareMathSymbol{\braceru}{\mathord}{largesymbols}{7D}
557 \DeclareRobustCommand\downbracefill{${\m@th \setbox\z@\hbox{${\braceld$}%
558 \braceld\leaders\vrule \@height\ht\z@ \@depth\z@\hfill\braceru
```

```

559 \bracelu\leaders\vrule \@height\ht\z@ \@depth\z@\hfill\bracerd$}
560 \DeclareRobustCommand\upbracefill{$\math \setbox\z@\hbox{$\braceld$}%
561 \bracelu\leaders\vrule \@height\ht\z@ \@depth\z@\hfill\bracerd
562 \braceld\leaders\vrule \@height\ht\z@ \@depth\z@\hfill\braceru$}

```

6.4.11 Delimiters

```

563 \DeclareMathDelimiter{\lmoustache} % top from (, bottom from )
564 {\mathopen}{largesymbols}{7A}{largesymbols}{40}
565 \DeclareMathDelimiter{\rmoustache} % top from ), bottom from (
566 {\mathclose}{largesymbols}{7B}{largesymbols}{41}
567 \DeclareMathDelimiter{\arrowvert} % arrow without arrowheads
568 {\mathord}{symbols}{6A}{largesymbols}{3C}
569 \DeclareMathDelimiter{\Arrowvert} % double arrow without arrowheads
570 {\mathord}{symbols}{6B}{largesymbols}{3D}
571 \DeclareMathDelimiter{\Vert}
572 {\mathord}{symbols}{6B}{largesymbols}{0D}

```

`\DeclareMathDelimiter` produces a command that is robust (with an internal macro containing the payload) so we should not use `\let` for making an alias

```

573 %\let\l=\Vert
574 \DeclareMathDelimiter{\l}
575 {\mathord}{symbols}{6B}{largesymbols}{0D}
576 \DeclareMathDelimiter{\vert}
577 {\mathord}{symbols}{6A}{largesymbols}{0C}
578 \DeclareMathDelimiter{\uparrow}
579 {\mathrel}{symbols}{22}{largesymbols}{78}
580 \DeclareMathDelimiter{\downarrow}
581 {\mathrel}{symbols}{23}{largesymbols}{79}
582 \DeclareMathDelimiter{\updownarrow}
583 {\mathrel}{symbols}{6C}{largesymbols}{3F}
584 \DeclareMathDelimiter{\Uparrow}
585 {\mathrel}{symbols}{2A}{largesymbols}{7E}
586 \DeclareMathDelimiter{\Downarrow}
587 {\mathrel}{symbols}{2B}{largesymbols}{7F}
588 \DeclareMathDelimiter{\Updownarrow}
589 {\mathrel}{symbols}{6D}{largesymbols}{77}
590 \DeclareMathDelimiter{\backslash} % for double coset G\backslash H
591 {\mathord}{symbols}{6E}{largesymbols}{0F}
592 \DeclareMathDelimiter{\rangle}
593 {\mathclose}{symbols}{69}{largesymbols}{0B}
594 \DeclareMathDelimiter{\langle}
595 {\mathopen}{symbols}{68}{largesymbols}{0A}
596 \DeclareMathDelimiter{\rbrace}
597 {\mathclose}{symbols}{67}{largesymbols}{09}
598 \DeclareMathDelimiter{\lbrace}
599 {\mathopen}{symbols}{66}{largesymbols}{08}
600 \DeclareMathDelimiter{\rceil}
601 {\mathclose}{symbols}{65}{largesymbols}{07}
602 \DeclareMathDelimiter{\lceil}
603 {\mathopen}{symbols}{64}{largesymbols}{06}
604 \DeclareMathDelimiter{\rfloor}
605 {\mathclose}{symbols}{63}{largesymbols}{05}
606 \DeclareMathDelimiter{\lfloor}
607 {\mathopen}{symbols}{62}{largesymbols}{04}

```

`\lgroup` There are three plain TeX delimiters which are not fully supported by NFSS, since they
`\rgroup` partly point into a bold cmr font. Allocating a full symbol font, just to have three
`\bracevert` delimiters seems a bit too much given the limited space available. For this reason only
the extensible sizes are supported. If this is not desired one can use, without losing
portability, define `\mathbf` and `\mathtt` as font symbol alphabet (setting up `cmr/bx/n`
and `cmtt/m/n` as symbol fonts first) and modify the delimiter declarations to point with
their small variant to those symbol fonts. (This is done in `oldlfont.dtx` so look there
for examples.)

```
608 \DeclareMathDelimiter{\lgroup} % extensible ( with sharper tips
609     {\mathopen}{largesymbols}{"3A}{largesymbols}{"3A}
610 \DeclareMathDelimiter{\rgroup} % extensible ) with sharper tips
611     {\mathclose}{largesymbols}{"3B}{largesymbols}{"3B}
612 \DeclareMathDelimiter{\bracevert} % the vertical bar that extends braces
613     {\mathord}{largesymbols}{"3E}{largesymbols}{"3E}
```

(End definition for \lgroup, \rgroup, and \bracevert.)

6.5 Math versions of text commands

The `\mathunderscore` here is really a text definition, so it has been put back into
`loutenc.dtx` (by Chris, 30/04/97) and should be removed from here.

These symbols are the math versions of text commands such as `\P`, `\$`, etc.

```
\mathparagraph These math symbols are not in plain TeX.
\mathsection 614 \DeclareMathSymbol{\mathparagraph}{\mathord}{symbols}{"7B}
\mathdollar 615 \DeclareMathSymbol{\mathsection}{\mathord}{symbols}{"78}
\mathsterling 616 \DeclareMathSymbol{\mathdollar}{\mathord}{operators}{"24}
\mathunderscore 617 \DeclareRobustCommand\mathsterling{\mathit{\mathchar"7024}}
618 \DeclareRobustCommand\mathunderscore{\kern.06em\vbox{\hrule\@width.3em}}
```

(End definition for \mathparagraph and others.)

`\mathellipsis` This is plain TeX's `\ldots`.

```
619 \DeclareRobustCommand\mathellipsis{\mathinner{\ldotp\ldotp\ldotp}}%
```

(End definition for \mathellipsis.)

6.6 Other special functions and parameters

6.6.1 Biggggg

```
620 </math>
621 <*math | latexrelease>
622 <latexrelease> \IncludeInRelease{2018/12/01}%
623 <latexrelease>         {\Big}{Start LR-mode}%
624 \DeclareRobustCommand\big[1]{\leavevmode@ifvmode
625     {\hbox{$\left#1\vbox to8.5\p@{\right.\n@space$}}}}
626 \DeclareRobustCommand\Big[1]{\leavevmode@ifvmode
627     {\hbox{$\left#1\vbox to11.5\p@{\right.\n@space$}}}}
628 \DeclareRobustCommand\bigg[1]{\leavevmode@ifvmode
629     {\hbox{$\left#1\vbox to14.5\p@{\right.\n@space$}}}}
630 \DeclareRobustCommand\Bigg[1]{\leavevmode@ifvmode
631     {\hbox{$\left#1\vbox to17.5\p@{\right.\n@space$}}}}
632 </math | latexrelease>
```

```

633 \latexrelease\EndIncludeInRelease
634 \latexrelease\IncludeInRelease{0000/00/00}%
635 \latexrelease{\Big}{Start LR-mode}%
636 \latexrelease\def\big#1{{\hbox{$\left#1\ vbox to8.5\p@{\}\right.\n@space$}}}}
637 \latexrelease\def\Big#1{{\hbox{$\left#1\ vbox to11.5\p@{\}\right.\n@space$}}}}
638 \latexrelease\def\bigg#1{{\hbox{$\left#1\ vbox to14.5\p@{\}\right.\n@space$}}}}
639 \latexrelease\def\Bigg#1{{\hbox{$\left#1\ vbox to17.5\p@{\}\right.\n@space$}}}}
640 \latexrelease\EndIncludeInRelease
641 \*math)

642 \def\n@space{\null\delimiterspace\z@ \m@th}

```

6.6.2 The log-like functions

`\operator@font` The `\operator@font` determines the symbol font used for log-like functions.

```

643 \def\operator@font{\mathgroup\symoperators}

(End definition for \operator@font.)

```

6.6.3 Parameters

```

644 \thinmuskip=3mu
645 \medmuskip=4mu plus 2mu minus 4mu
646 \thickmuskip=5mu plus 5mu

This finishes the low-level setup in fontmath.ltx.
647 \math)

```

7 Default cfg files

We provide default `cfg` files here to ensure that on installations that search large file trees we do not pick up some strange customisation files from somewhere.

```

648 \*cfgtext | cfgmath | cfgprel)
649 %%
650 %%
651 %%
652 %% Load the standard setup:
653 %%
654 \+cfgtext\input{fonttext.ltx}
655 \+cfgmath\input{fontmath.ltx}
656 \+cfgprel\input{preload.ltx}
657 %%
658 %% Small changes could go here; see documentation in cfgguide.tex for
659 %% allowed modifications.
660 %%
661 %% In particular it is not allowed to misuse this configuration file
662 %% to modify internal LaTeX commands!
663 %%
664 %% If you use this file as the basis for configuration please change
665 %% the \ProvidesFile lines to clearly identify your modification, e.g.,
666 %%
667 \+cfgtext%% \ProvidesFile{fonttext.cfg}[2001/06/01
668 \+cfgmath%% \ProvidesFile{fontmath.cfg}[2001/06/01
669 \+cfgprel%% \ProvidesFile{preload.cfg}[2001/06/01
670 %% Customised local font setup]
671 %%

```

```
672 %%  
673 </cfgtext | cfgmath | cfgprel>
```

File B

preload.dtx

1 Overview

This file contains an number of possible settings for preloading fonts during installation of NFSS2 (which is used by L^AT_EX 2_ε). It will be used to generate the following files:

preload.min	minimal subset of fonts necessary to run NFSS2
preload.ori	preload of CM fonts similar to the old <code>lfonts.tex</code>
preload.ltx	The standard selection of preloads
cmpreload.xpt	preload of CM fonts for 10pt document size
cmpreload.xip	preload of CM fonts for 11pt document size
cmpreload.xii	preload of CM fonts for 12pt document size
dcpreload.xpt	preload of DC fonts for 10pt size
dcpreload.xip	preload of DC fonts for 11pt size
dcpreload.xii	preload of DC fonts for 12pt size

These files are for installations that make use of Computer Modern fonts either old encoding (OT1) or Cork encoding (T1). The Computer Modern fonts with Cork encoding are known as DC-fonts.

Most important is `preload.ltx` which is used during format generation. You are *not* allowed to change this file.

2 Customization

You can customize the preloaded fonts in your L^AT_EX 2_ε system by installing a file with the name `preload.cfg`. If this file exists it will be used in place of the system file `preload.ltx`. You can, for example, copy one of the files mentioned above (that can be generated from this source) to `preload.cfg`.

Or you can define completely other preloads. In that case start from `preload.min` since that contains the fonts that have to be preloaded by **all** L^AT_EX 2_ε systems.

Avoid using `preload.ori`, it will load so many fonts that on most installations it is nearly impossible to load other font families afterwards. This file is only generated to show what fonts have been preloaded by L^AT_EX 2.09.

If you normally use other fonts than Computer Modern `preload.min` might be best.

Warning: If you preload fonts with encodings other than the normally supported encodings you have to declare that encoding in a `fontdef.cfg` configuration file (see the documentation in the file `fontdef.dtx`). Adding an extra encoding to the format might produce non-portable documents, thus this should be avoided if possible.

3 Module switches for the DOCSTRIP program

The DOCSTRIP will generate the above file from this source using the following module directives:

driver	produce a documentation driver file
preload	produce a preload... file
cm	for OT1 encoded Computer Modern
dc	for T1 encoded Computer Modern
min	produce minimal subset
xpt	produce 10pt preloads
xipt	produce 11pt preloads
xipt	produce 12pt preloads
ori	produce preloads similar to old <code>lfonts.tex</code>
tex	produce <code>preload.ltx</code>

A typical DOCSTRIP command file would then have entries like:

```
generateFile{preload.min}{t}{\from{preload.dtx}{preload,min}}
```

for generating preload files.

4 A driver for this document

The next bit of code contains the documentation driver file for T_EX, i.e., the file that will produce the documentation you are currently reading. It will be extracted from this file by the DOCSTRIP program.

```

1 <*driver>
2 \documentclass{ltxdoc}
3 %\OnlyDescription % comment out for implementation details
4 \begin{document}
5   \DocInput{preload.dtx}
6 \end{document}
7 </driver>
```

5 The code

We begin by loading the math extension font (cmex10) and the L^AT_EX line and circle fonts. It is necessary to do this explicitly since these are used by `lplain.tex` and `latex.tex`. Since the internal font name contains / characters and digits we construct the name via `\csname`. These are the only fonts (!) that must be loaded in this file.

All `\DeclarePreloadSizes` can be removed or others can be added, they only influence the processing speed.

```

8 \expandafter\font\csname OMX/cmex/m/n/10\endcsname=cmex10\relax
9 \font\tenln =line10 \font\tenlnw =linew10\relax
10 \font\tencirc=lcircle10 \font\tencircw=lcirclew10\relax
```

The above fonts should not be touched but anything below this point here in the preload suggestions can be modified without any problems.

```

11 <-tex>%*****
12 <-tex>% Start any modification below this point **
13 <-tex>%*****
14 <-tex>
15 %%
16 %% Computer Modern Roman:
17 %%-----
```

```

18 <*ori>
19 \DeclarePreloadSizes{OT1}{cmr}{m}{n}
20     {5,6,7,8,9,10,10.95,12,14.4,17.28,20.74,24.88}
21 \DeclarePreloadSizes{OT1}{cmr}{bx}{n}{9,10,10.95,12,14.4,17.28}
22 \DeclarePreloadSizes{OT1}{cmr}{m}{sl}{10,10.95,12}
23 \DeclarePreloadSizes{OT1}{cmr}{m}{it}{7,8,9,10,10.95,12}
24 </ori>
25 <+xpt & cm> \DeclarePreloadSizes{OT1}{cmr}{m}{n}{5,7,10}
26 <+xpt & dc> \DeclarePreloadSizes{T1}{cmr}{m}{n}{5,7,10}
27 <+xipt & cm> \DeclarePreloadSizes{OT1}{cmr}{m}{n}{6,8,10.95}
28 <+xipt & dc> \DeclarePreloadSizes{T1}{cmr}{m}{n}{6,8,10.95}
29 <+xipt & cm> \DeclarePreloadSizes{OT1}{cmr}{m}{n}{6,8,12}
30 <+xipt & dc> \DeclarePreloadSizes{T1}{cmr}{m}{n}{6,8,12}
31 %%
32 %% Computer Modern Sans:
33 %%-----
34 <+ori> \DeclarePreloadSizes{OT1}{cmss}{m}{n}{10,10.95,12}
35 %%
36 %% Computer Modern Typewriter:
37 %%-----
38 <+ori> \DeclarePreloadSizes{OT1}{cmtt}{m}{n}{9,10,10.95,12}
39 %%
40 %% Computer Modern Math:
41 %%-----
42 <*ori>
43 \DeclarePreloadSizes{OML}{cmm}{m}{it}
44     {5,6,7,8,9,10,10.95,12,14.4,17.28,20.74}
45 \DeclarePreloadSizes{OMS}{cmsy}{m}{n}
46     {5,6,7,8,9,10,10.95,12,14.4,17.28,20.74}
47 </ori>

```

The math fonts are the same for both DC and CM fonts. So far there isn't an agreed on standard.

```

48 <*xpt>
49 \DeclarePreloadSizes{OML}{cmm}{m}{it}{5,7,10}
50 \DeclarePreloadSizes{OMS}{cmsy}{m}{n}{5,7,10}
51 </xpt>
52 <*xipt>
53 \DeclarePreloadSizes{OML}{cmm}{m}{it}{6,8,10.95}
54 \DeclarePreloadSizes{OMS}{cmsy}{m}{n}{6,8,10.95}
55 </xipt>
56 <*xipt>
57 \DeclarePreloadSizes{OML}{cmm}{m}{it}{6,8,12}
58 \DeclarePreloadSizes{OMS}{cmsy}{m}{n}{6,8,12}
59 </xipt>
60 %%
61 %% LaTeX symbol fonts:
62 %%-----
63 <*ori>
64 \DeclarePreloadSizes{U}{lasy}{m}{n}
65     {5,6,7,8,9,10,10.95,12,14.4,17.28,20.74}
66 </ori>
67 </preload>

```


File C

lftntcmd.dtx

Abstract

The commands defined in this file `lftntcmd` are part of the kernel code for $\text{\LaTeX 2}_{\epsilon}/\text{NFSS2}$.

It is also meant to serve as documentation for package writers since it demonstrates how to define high-level font changing commands using a small number of creator functions.

1 Introduction

Font changes such as `\bfseries`, `\sffamily`, etc. are declarations; this means that their scope is delimited by the grouping structure, either by the next `\end` of some environment or by explicitly using a group, e.g., writing something like `{\bfseries...}` in the source. If you make the mistake of writing `\bfseries{...}` (thinking of `\bfseries` as a command with one argument) then the result is rather striking.

Font declarations are an artifact of the \TeX system and for several reasons it is better to avoid them on the user level whenever possible. In \LaTeX 3 they will probably all be replaced by environments and by font commands taking one argument.

This file defines a creator function for such declarative font switches. This function creates commands which can be used in both math and text.

This file also defines a number of high-level commands (all starting with `\text..`) that have one argument and typeset this argument in the requested way. Thus these commands are for typesetting short pieces of text in a specific family, series or shape. These are all produced as examples of the use of a creator function which is itself also defined in this file.

Table 1 shows all these high-level commands in action. A further advantage of using these commands is that they automatically take care of any necessary italic correction on either side of their argument.

Thus, when using such commands, one does not have to worry about forgetting the italic correction when changing fonts. Only in very few situations is this additional space wrong but, for example, most typographers recommend omitting the italic correction if a small punctuation character, like a comma, directly follows the font change. Since the amount of correction required is partly a matter of taste, you can define in what situations the italic correction should be suppressed. This is done by putting the characters that should cancel a preceding italic correction in the list `\nocorrlist`.²⁹ The default definition for this list is produced by the following.

```
\newcommand \nocorrlist {,.}
```

It is best to declare the most often used characters first, because this will make the processing slightly faster. For example,

```
\emph{When using the \NFSS{} high-level commands,  
the \emph{proper} use of italic corrections is  
automatically taken care of}. Only
```

²⁹Any package that changes the `\catcode` of a character inside `\nocorrlist` must then explicitly reset the list. Otherwise the changed character will no longer be recognized by the suppression algorithm.

<i>Command</i>	<i>Corresponds to</i>	<i>Action</i>
<code>\textnormal{..}</code>	<code>\normalfont</code>	Typeset argument in normal family
<code>\textrm{..}</code>	<code>\rmfamily</code>	Typeset argument in roman family
<code>\textsf{..}</code>	<code>\sffamily</code>	Typeset argument in sans serif family
<code>\texttt{..}</code>	<code>\ttfamily</code>	Typeset argument in typewriter family
<code>\textmd{..}</code>	<code>\mdseries</code>	Typeset argument in medium series
<code>\textbf{..}</code>	<code>\bfseries</code>	Typeset argument in bold series
<code>\textup{..}</code>	<code>\upshape</code>	Typeset argument in normal shape
<code>\textit{..}</code>	<code>\itshape</code>	Typeset argument in <i>italic</i> shape
<code>\textsl{..}</code>	<code>\slshape</code>	Typeset argument in <i>slanted</i> shape
<code>\textsc{..}</code>	<code>\scshape</code>	Typeset argument in SMALL CAPS shape
<code>\emph{..}</code>	<code>\em</code>	Typeset argument <i>emphasized</i>

Table 1: Font-change commands with arguments

The font change commands provided here all start with `\text..` to emphasize that they are for use in normal text and to be easily memorable. They automatically take care of any necessary italic correction on either side of the argument.

`\emph{sometimes}` one has to help `\LaTeX{}` by adding a `\verb=\nocorr=` command.

which results in:

When using the NFSS high-level commands, the proper use of italic corrections is automatically taken care of. Only sometimes one has to help L^AT_EX by adding a `\nocorr` command.

In contrast, the use of the declaration forms is often more appropriate when you define your own commands or environments.

```
\newenvironment{bfitemize}{\begin{itemize}\normalfont\bfseries}
                          {\end{itemize}}
\begin{bfitemize}
\item This environment produces boldface items.
\item It is defined in terms of \LaTeX's
      \texttt{itemize} environment and NFSS
      declarations.
\end{bfitemize}
```

This gives:

- **This environment produces boldface items.**
- **It is defined in terms of L^AT_EX's `itemize` environment and NFSS declarations.**

In addition to global customization of when to insert the italic correction, it is of course sometimes necessary to explicitly insert one with `\/`.

It is also possible to suppress the italic correction in individual instances. For this, the command `\nocorr` is provided.

The `\nocorr` must appear as the first or last token inside the braces of the argument of the `\text...` commands, at that end of the text where you wish to suppress the italic correction.

It is worth pointing out here that inserting a `\/` in places where it can have no function (i.e. anywhere except immediately after a slanted letter) is not an error—it will just be silently ignored. Unfortunately this is not true if the redefinition of `\/` in `amstex.sty` is used as this version can cause space to be removed immediately before the `\/`.

2 The implementation

`\DeclareTextFontCommand` This is the creator function for `\text..` commands. It gives a warning if `\foo` or `\fragfoo` is already defined.

In math mode it simply puts the font declaration and text into a box (possibly an automagically sized one).

Otherwise it first scans the text to see where `\nocorr` occurs within it. This sets the `\check@ic` commands to do what is necessary concerning the italic correction at both ends.

The algorithm for deciding whether to put in an italic correction is not very subtle: one is added whenever the newly current font is not itself positively sloped, unless the next token is a character in the ‘nocorr’ list. At the end of the text this is done after closing the group so as to check the ‘outer font’. Note that this will often result in adding an italic correction token after a character in an unsloped font; we believe (in early 2003) that this is perhaps inefficient but not dangerous.

It also now checks for empty contents of the text command and optimizes this case. Some care is also taken to check that doing dangerous things in vertical mode is avoided.

The italic correction token is added to the horizontal list before (in the list) an immediately preceding non-zero glob of glue (skip) and any non-zero penalty preceding that since, in the typical case, this puts it immediately after the last character in the preceding word.

Note that it is necessary to put in the `\aftergroup\maybe@ic` at the end of the group so that it comes after any other aftergroup tokens and immediately before the following tokens. It is also necessary to remove the `\fi` from the token list before the group ends; this is done by adding an `\expandafter` just before the closing brace.

```

1  \langle*2kernel\rangle
2  \def \DeclareTextFontCommand #1#2{%
3    \DeclareRobustCommand#1[1]{%
4      \ifmmode
5        \nfss@text{#2##1}%
6      \else
7        \hmode\bgroup
8        \text@command{##1}%
9        #2\check@icl ##1\check@icr
10       \expandafter
11       \egroup
12       \fi
13           }%
14  }
```

(End definition for `\DeclareTextFontCommand`.)

```
\textrm Now we define the \text<family> commands in terms of the above; \texttt does not
\textsf look very nice!
\texttt 15 \DeclareTextFontCommand{\textrm}{\rmfamily}
\textnormal 16 \DeclareTextFontCommand{\textsf}{\sffamily}
17 \DeclareTextFontCommand{\texttt}{\ttfamily}
18 \DeclareTextFontCommand{\textnormal}{\normalfont}
```

(End definition for `\textrm` and others.)

```
\textbf For the series attribute:
\textmd 19 \DeclareTextFontCommand{\textbf}{\bfseries}
20 \DeclareTextFontCommand{\textmd}{\mdseries}
```

(End definition for `\textbf` and `\textmd`.)

```
\textit And for the shapes:
\textsl 21 \DeclareTextFontCommand{\textit}{\itshape}
\textsc 22 \DeclareTextFontCommand{\textsl}{\slshape}
\textup 23 \DeclareTextFontCommand{\textsc}{\scshape}
24 \DeclareTextFontCommand{\textup}{\upshape}
```

(End definition for `\textit` and others.)

```
textulc
textsw 25 </2ekernel>
textssc 26 <*2ekernel | latexrelease>
27 <latexrelease>\IncludeInRelease{2020/02/02}%
28 <latexrelease> {\textulc}{Additional text commands}%
29 \DeclareTextFontCommand{\textulc}{\ulcshape}
30 \DeclareTextFontCommand{\textsw}{\swshape}
31 \DeclareTextFontCommand{\textssc}{\sscshape}
32 </2ekernel | latexrelease>
33 <latexrelease>\EndIncludeInRelease
34 <latexrelease>\IncludeInRelease{0000/00/00}%
35 <latexrelease> {\textulc}{Additional text commands}%
36 <latexrelease>
37 <latexrelease>\let\textulc\@undefined
38 <latexrelease>\let\textsw\@undefined
39 <latexrelease>\let\textssc\@undefined
40 <latexrelease>\EndIncludeInRelease
41 <*2ekernel>
```

(End definition for `textulc`, `textsw`, and `textssc`.)

`\emph` Finally we have the `\em` font change declaration of L^AT_EX. The corresponding definition with argument is

```
42 \DeclareTextFontCommand{\emph}{\em}
```

(End definition for `\emph`.)

`\nocorr` This is just a label, so it does nothing; it should also be unexpandable.

```
43 \let \nocorr \relax
```

(End definition for `\nocorr`.)

`\check@ic1` We define these defaults in case some error causes them to be expanded at the wrong
`\check@icr` time.

```
44 \let \check@ic1 \@empty
45 \let \check@icr \@empty
```

(End definition for `\check@ic1` and `\check@icr`.)

`\text@command` This checks for a `\nocorr` as the first token in its argument and also for one in any other
`\check@nocorr@` position not protected within braces (the latter is treated as if it were at the end of the argument).

Is this the correct action in the ‘empty’ case? It is efficient but typographically it is, strictly, incorrect!

```
46 \def \text@command #1{%
47   \def \reserved@a {#1}%
48   \ifx \reserved@a \@empty
49     \let \check@ic1 \@empty
50     \let \check@icr \@empty
51   \else
```

`\space` is a reserved word in L^AT_EX or actually already in plain T_EX. If somebody really redefines it so many things will break that I don’t see any reason to make this routine here slower than necessary.

```
52 %   \def \reserved@b { }%
53 %   \ifx \reserved@a \reserved@b
54   \ifx \reserved@a \space
55     \let \check@ic1 \@empty
56     \let \check@icr \@empty
57   \else
58     \check@nocorr@ #1\nocorr\@nil
59   \fi
60 \fi
61 }
62 \def \check@nocorr@ #1#2\nocorr#3\@nil {%
```

The two checks are initialised here to their values in the normal case.

```
63 \let \check@ic1 \maybe@ic
64 \def \check@icr {\ifvmode \else \aftergroup \maybe@ic \fi}%
65 \def \reserved@a {\nocorr}%
66 \def \reserved@b {#1}%
67 \def \reserved@c {#3}%
68 \ifx \reserved@a \reserved@b
69   \ifx \reserved@c \@empty
```

In this case there is a `\nocorr` at the start but not at the end, so `\check@ic1` should be empty.

```
70   \let \check@ic1 \@empty
71 \else
```

Otherwise there is a `\nocorr` both at the start and elsewhere, so no italic corrections should be added.

```
72   \let \check@ic1 \@empty
73   \let \check@icr \@empty
74 \fi
```

```

75  \else
76  \ifx \reserved@c \@empty

```

In this case there is no `\nocorr` anywhere, so we need to check for an italic correction at both the beginning and the end. This has been set up as the default so no code is needed here.

```

77  \else

```

In this case there is no `\nocorr` at the start but there is one elsewhere, so no `\aftergroup` is needed.

```

78      \let \check@icr \@empty
79  \fi
80  \fi
81 }

```

(End definition for \text@command and \check@nocorr@.)

`\ifmaybe@ic` Switch used solely within `\maybe@ic` not interfering with other switches.

```

82 \newif\ifmaybe@ic

```

(End definition for \ifmaybe@ic.)

`\maybe@ic` These macros implement the italic correction.

```

\maybe@ic@
83 \def \maybe@ic {\futurelet \@let@token \maybe@ic@}
84 \def \maybe@ic@ {%

```

We first check to see if the current font is positively sloped. (But do not forget the message Rainer sent about an upright font with non-zero slope! Or is this an urban myth?) It has been suggested that this should test against a small positive value, but what?

```

85  \ifdim \fontdimen\@ne\font>\z@
86  \else
87  \maybe@ictrue

```

It would be possible, but probably not worthwhile, to continue the forward scan beyond any closing braces.

```

88  \expandafter \@tfor \expandafter \reserved@a \expandafter : \expandafter = %
89  \nocorrlist

```

We have to hide the `\@let@token` in the macro `\t@st@ic` rather than testing it directly in the loop since it might be `\let` to a `\fi` or `\else`, which would result in chaos.

```

90  \do \t@st@ic

```

Frank thinks that the next bit is inefficient if done after the second change. Chris thinks that most all of this is inefficient for the commonest cases: but that is the price of a cleverer algorithm. It is certainly needed to deal with the use of `\nolinebreak`.

```

91  \ifmaybe@ic \sw@slant \fi
92  \fi
93 }

```

(End definition for \maybe@ic and \maybe@ic@.)

`\t@st@ic` The next token in the input stream is stored in `\@let@token` via a `\let`, the current token from `\nocorrlist` is stored via `\def` in `\reserved@a`. To compare them we have to fiddle around a bit.

If the only things to check were characters then this could be done via an `\if` thus their catcodes would not matter; but this will not work whilst `\futurelet` is used above.

```

94 \def \t@st@ic {%
95   \expandafter\let\expandafter\reserved@b\expandafter=\reserved@a\relax
96   \ifx\reserved@b\@let@token

```

If they are the same we record the fact and jump out of the loop.

```

97     \maybe@icfalse
98     \@break@tfor
99   \fi
100 }

```

(End definition for `\t@st@ic`.)

`\sw@slant` The definition of the mysterious `\sw@slant` command is as follows.
`\fix@penalty` `\def \sw@slant {%`

It is surely correct to put in an italic correction when there is no skip. If the last thing on the list is actually a zero skip (including things whose dimension part is zero, such as `\hfill`), or anything other than a character, then the italic correction will have no effect.

In order to work correctly with unbreakable spaces from `~` (and other common forms of line-breaking control) we also move back across a penalty before the glue.

```

102 \ifdim \lastskip=\z@
103   \fix@penalty
104 \else
105   \skip@ \lastskip
106   \unskip
107   \fix@penalty
108   \hskip \skip@
109 \fi
110 }

```

The above code means: “If there is a non-zero space just before the current position (`\ifdim...`) save the amount of that space (`\skip@\lastskip`), remove it (`\unskip`), then do a similar thing if there is a penalty just before the skip, and finally put the space back in.”

Since zero glue cannot be distinguished in this context from no glue, we dare not put in an `\hskip` in this case as this may produce an unwanted breakpoint. This is not satisfactory.

The penalty before the glue is handled similarly, with the same caveats concerning the zero case. Is this the first recorded use of `\unpenalty` in standard L^AT_EX code?

```

111 \def \fix@penalty {%
112   \ifnum \lastpenalty=\z@
113     \@@italiccorr
114   \else
115     \count@ \lastpenalty
116     \unpenalty
117     \@@italiccorr

```

```

118     \penalty \count@
119     \fi
120 }

```

(End definition for `\sw@slant` and `\fix@penalty`.)

\nocorrlist This holds the list of characters that should prevent italic correction. They should be ordered by decreasing frequency of use. If any such character is made active later on one needs to redefine the list so that the active character becomes part of it.

```

121 \def \nocorrlist {,.}

```

(End definition for `\nocorrlist`.)

\nfss@text This command will by default behave like a L^AT_EX `\mbox` but may be redefined by packages such as `amstext.sty` to be a bit cleverer.

```

122 \ifx \nfss@text\undefined
123   \def \nfss@text {\leavevmode\hbox}
124 \fi

```

(End definition for `\nfss@text`.)

\DeclareOldFontCommand This is the function used to create declarative font-changing commands that can also be used to change alphabets in math-mode.

Usage: `\DeclareOldFontCommand \fn{<font-change decls>} <math-alphabet>`

Here `\fn` is the font-declaration command being defined, `<font-change decls>` is the declaration it will expand to in text-mode, and `<math-alphabet>` is the (single) math alphabet specifier which is to be used in math-mode.

It does not care whether the command being defined already exists but it does give a warning if it redefines anything.

Here are some typical examples of its use in conjunction with more basic NFSS2 font commands.

```

\DeclareOldFontCommand{\rm}{\normalfont\rmfamily}{\mathrm}
\DeclareOldFontCommand{\sf}{\normalfont\sffamily}{\mathsf}
\DeclareOldFontCommand{\tt}{\normalfont\ttfamily}{\mathtt}

```

```

125 \def \DeclareOldFontCommand #1#2#3{%
126   \DeclareRobustCommand #1{\@fontswitch {#2}{#3}}%
127 }

```

(End definition for `\DeclareOldFontCommand`.)

\@fontswitch **\@math@egroup** These two commands actually do the necessary tests and declarative font- or alphabet-changing.

```

\@math@egroup
\@math@egroup
128 \def \@fontswitch #1#2{%
129   \ifmmode
130     \let \math@bgroup \relax
131     \def \math@egroup {\let \math@bgroup \@math@bgroup
132                       \let \math@egroup \@math@egroup}%

```


We need to have a `\relax` in the following line in case the `#2` is something like `\mathsf` grabbing the next token as an argument. For this reason the code also uses explicit arguments again (see pr/1275).

```

133     #2\relax
134   \else
135     #1%
136   \fi
137 }
138 \let \@@math@bgroup \math@bgroup
139 \let \@@math@egroup \math@egroup

(End definition for \@fontswitch, \@@math@egroup, and \@@math@egroup.)
These commands are available only in the preamble.
140 \@onlypreamble \DeclareTextFontCommand
141 \@onlypreamble \DeclareOldFontCommand

```

3 Initialization

```

\normalsize This is defined to produce an error.
142 \def\normalsize{%
143   \@latex@error {The font size command \protect\normalsize\space
144                 is not defined:\MessageBreak
145                 there is probably something wrong with
146                 the class file}\@eha
147 }
148 \</2ekernel>

(End definition for \normalsize.)

```

File D

ltxtextcomp.dtx

From File: ltxtextcomp.dtx

This file contains the implementation for accessing the glyphs provided by the TS1 encoding (Text Companion Encoding). This is now offered as part of the kernel and so the `textcomp` package which used to provide the definitions is now mainly needed for compatibility reasons (and doesn't do much any more).

```
1 \*2ekernel | latexrelease>
2 \latexrelease>\NewModuleRelease{2020/02/02}{ltxtextcomp}
3 \latexrelease>                                {Text Companion symbols}
```

`\oldstylenums` Preserve the old definition of `\oldstylenums` under a different name.

`\legacyoldstylenums` This macro implements old style numerals but only works if we assume that the standard math fonts are used. Thus it needs changing in case other math encodings are used.

```
4 \DeclareRobustCommand\legacyoldstylenums[1]{%
5   \begingroup
```

Provide spacing using the interword space of the current font.

```
6   \spaceskip\fontdimen\tw@\font
```

Then switch to the math italic font. We don't change the current value of `\f@series` which means that you can use bold numerals if `\bfseries` is in force. As family we use `\rmdefault` which means that this only works if there exist an OML encoded version of that font or rather a corresponding .fd file (which is the case for standard L^AT_EX fonts even though they only contain substitutions).

```
7   \usefont{OML}{\rmdefault}{\f@series}{it}%
8   \mathgroup\symletters #1%
9   \endgroup
10 }
```

And here is the improved one that adjusts depending on surroundings.

```
11 \DeclareRobustCommand\oldstylenums[1]{%
12   \begingroup
13   \ifmmode
14     \mathgroup\symletters #1%
15   \else
```

The `\CheckEncodingSubset` is discussed below.

```
16   \CheckEncodingSubset\@use@text@encoding{TS1}{\tc@oldstylesubst2{{#1}}}%
17   \fi
18   \endgroup
19 }
```

The helper to select the substitution if needed.

```
20 \def\tc@oldstylesubst#1{%
21   \tc@errorwarn
22     {Oldstyle digits unavailable for
23     family \f@family.\MessageBreak
24     Default oldstyle digits used instead}\@eha
25   \bgroup
26   \expand@font@defaults
```

The substitution defaults are provided in the file `fonttext.ltx`.

```

27     \ifx\f@family\rmdef@ult
28         \fontfamily\rmsubstdefault
29     \else\ifx\f@family\sfsdef@ult
30         \fontfamily\sfsbstdefault
31     \else\ifx\f@family\ttdef@ult
32         \fontfamily\ttsbstdefault
33     \else
34         \fontfamily\textcompsubstdefault
35     \fi\fi\fi
36     \fontencoding{TS1}\selectfont#1%
37 \egroup
38 }

```

(End definition for `\oldstylenums` and `\legacyoldstylenums`.)

`\textcompsubstdefault` Here is the default for the “unknown” case:

```

39 \def\textcompsubstdefault{\rmsubstdefault}

```

(End definition for `\textcompsubstdefault`.)

`\DeclareEncodingSubset` The declaration takes 3 mandatory arguments: an *encoding* for which a subsetting is wanted (currently always `TS1`, and most likely forever), the *font family* for which we declare the subset and finally the *subset* number (between 0 (all of the encoding is supported) and 9 many glyphs are missing).

For `TS1` the numbers have been chosen in a way that most fonts can be fairly correctly categorized, but the default settings are always conservative, that is they may claim that less glyphs are supported than there actually are.

As these days many font families are set up to end in `-LF` (lining figures), `-OsF` (oldstyle figures), etc. the declaration supports a shortcut: if the *font family* name ends in `-*` then the star gets replaced by these common ending, e.g.,

```
\DeclareEncodingSubset{TS1}{Alegreya-*}{2}
```

is the same as writing

```

\DeclareEncodingSubset{TS1}{Alegreya-LF}{2}
\DeclareEncodingSubset{TS1}{Alegreya-OsF}{2}
\DeclareEncodingSubset{TS1}{Alegreya-TLF}{2}
\DeclareEncodingSubset{TS1}{Alegreya-TOsF}{2}

```

If only some are needed then one can define them individually but in many cases all four are wanted, hence the shortcut.

The coding of the declaration has no error checking as it is mostly for internal use.

```

40 \def\DeclareEncodingSubset#1#2{%
41     \DeclareEncodingSubset@aux{#1}#2*\DeclareEncodingSubset@aux
42 }
43 \def\DeclareEncodingSubset@aux#1#2*#3\DeclareEncodingSubset@aux#4{%

```

if #3 is empty then there was no star, otherwise we define all four variants.

```

44 \expandafter\ifx\expandafter X\detokenize{#3}X%
45 \DeclareEncodingSubset{#1}{#2}{#4}%
46 \else
47 \DeclareEncodingSubset{#1}{#2LF}{#4}%
48 \DeclareEncodingSubset{#1}{#2TLF}{#4}%
49 \DeclareEncodingSubset{#1}{#2OsF}{#4}%
50 \DeclareEncodingSubset{#1}{#2T0sF}{#4}%
51 \fi
52 }

```

The subset info is stored in a command with the name `\family:subset` so if that already exists we change otherwise declare a subset.

```

53 \def\DeclareEncodingSubset#1#2#3{%
54 \ifundefined{#1:#2}%
55 {\font@info{Setting #2 sub-encoding to #1/#3}}%
56 {\font@info{Changing #2 sub-encoding to #1/#3}}%
57 \namedef{#1:#2}{#3}}

```

Any reason to allow those in the middle of documents?

```

58 \onlypreamble\DeclareEncodingSubset
59 \onlypreamble\DeclareEncodingSubset@aux
60 \onlypreamble\DeclareEncodingSubset

```

(End definition for `\DeclareEncodingSubset`.)

`\CheckEncodingSubset` The command `\CheckEncodingSubset` will check if the current font family has the right encoding subset to typeset a certain command. It takes five arguments as follows: first argument is either `\UseTextSymbol`, `\UseTextAccent` depending on whether or not the symbol is a text symbol or a text accent.

The second argument is the encoding from which this symbol should be fetched.

The third argument is either a fake accessor command or an error message. the code in that argument (if ever executed) receives two arguments: #2 and #5 of `\CheckEncodingSubset`.

Argument four is the subset encoding id to test against: if this value is higher than the subset id of the current font family then we typeset the symbol, i.e., execute `#1{#2}#5` otherwise it runs `#3#5`, e.g., to produce an error message or fake the glyph somehow.

Argument five is the symbol or accent command that is being checked.

For usage examples see definitions below.

```

61 \def\CheckEncodingSubset#1#2#3#4#5{%
62 \ifnum #4>%
63 \expandafter\ifx\csname #2:\f@family\endcsname\relax
64 0\csname #2:\f@family\endcsname
65 \else
66 \csname #2:\f@family\endcsname
67 \fi
68 \relax
69 \expandafter\@firstoftwo
70 \else
71 \expandafter\@secondoftwo
72 \fi
73 {#1{#2}}{#3}%
74 #5%
75 }

```

(End definition for `\CheckEncodingSubset`.)

To set up the glyphs for the subsets we need a number helpers.

`\tc@errorwarn` To we produce errors, warnings, or only info in the transcripts if glyphs require substitutions? By default it is “info” only. With the `textcomp` package that can be changed.

```
76 \def\tc@errorwarn#1#2{\@latex@info{#1}}
```

(End definition for `\tc@errorwarn`.)

`\tc@subst`

```
77 \def\tc@subst#1{%
78   \tc@errorwarn
79   {Symbol \string#1 not provided by\MessageBreak
80    font family \f@family\space
81    in TS1 encoding.\MessageBreak Default family used instead}\@eha
82   \bgroup
83     \expand@font@defaults
84     \ifx\f@family\rmdef@ult
85       \fontfamily\rmsubstdefault
86     \else\ifx\f@family\sfddef@ult
87       \fontfamily\sfsbstdefault
88     \else\ifx\f@family\ttdef@ult
89       \fontfamily\ttsbstdefault
90     \else
91       \fontfamily\textcompsubstdefault
92     \fi\fi\fi
```

Whatever default was chosen, we claim now (locally hopefully) that it can handle all slots (even if not true) to avoid looping in certain situations, e.g., when something was set up incorrectly.

```
93   \@namedef{TS1:\f@family}{0}%
94   \selectfont#1%
95   \egroup
96 }
```

(End definition for `\tc@subst`.)

`\tc@fake@euro` `\tc@fake@euro` is an example of a “fake” definition to use in arg #3 of the command `\CheckEncodingSubset` when a symbol is not available in a certain font family. Here we produce a poor man’s Euro symbol by combining a “C” with a “=”.

```
97 \def\tc@fake@euro#1{%
98   \leavevmode
99   \@font@info{Faking \noexpand#1for font family
100               \f@family\MessageBreak in TS1 encoding}%
101   \valign{##\cr
102     \vfil\hbox to 0.07em{\dimen@f@size\p@
103                           \math@fontsfalse
104                           \fontsize{.7\dimen@}\z@\selectfont=\hss}%
105     \vfil\cr%
106     \hbox{C}\crrcr
107   }%
108 }
```

(End definition for `\tc@fake@euro`.)

`\tc@check@symbol` These are two abbreviations that we use below to check symbols and accents in TS1.
`\tc@check@accent` Only there to save some space, e.g., we can then write

```
DeclareTextCommandDefault{\textcurrency}{\tc@check@symbol3\textcurrency}
```

to ensure that `\textcurrency` is only typeset if the current font has a TS1 subset id of less than 3. Otherwise `\tc@error` is called telling the user that for this font family `\textcurrency` is not available.

```
109 \def\tc@check@symbol{\CheckEncodingSubset\UseTextSymbol{TS1}\tc@subst}
```

Accents have been made an error in the `textcomp` package when not available. Now that we provide the functionality in the kernel we avoid the error by swapping in a T1 accent if the TS1 accent is not available.

```
110 %\def\tc@check@accent{\CheckEncodingSubset\UseTextAccent{TS1}\tc@error}
```

```
111 \def\tc@check@accent#1{\CheckEncodingSubset\UseTextAccent
112                               {TS1}{\tc@swap@accent#1}}
```

```
113 \def\tc@swap@accent#1#2{\UseTextAccent{T1}#1}
```

(End definition for `\tc@check@symbol` and `\tc@check@accent`.)

1 Sub-encodings

Here are the default definitions for the TS1 symbols. First those that we assume are always available if a font implements TS1.

```
114 \DeclareTextSymbolDefault{\textdollar}{TS1}
```

```
115 \UndeclareTextCommand{\textdollar}{OT1} % don't use the OT1 def any longer
```

```
116 \DeclareTextSymbolDefault{\textsterling}{TS1}
```

```
117 \UndeclareTextCommand{\textsterling}{OT1}% don't use the OT1 def any longer
```

```
118 \DeclareTextSymbolDefault{\textperthousand}{TS1}
```

```
119 \UndeclareTextCommand{\textperthousand}{T1} % don't use the T1 def
```

Using `\UndeclareTextCommand` above is enough only if the encoding definition files are not reloaded afterwards. In the past that happened if `fontenc` was used in the document preamble (not any longer). So in some sense it is better to fully remove them from the encoding files, but for rollbacks it is easier to keep them in for now.

These are the standard `itemize` and footnote symbols originally taken from OMS and now from TS1:

```
120 \DeclareTextSymbolDefault{\textasteriskcentered}{TS1}
```

```
121 \DeclareTextSymbolDefault{\textbullet}{TS1}
```

```
122 \DeclareTextSymbolDefault{\textdaggerdbl}{TS1}
```

```
123 \DeclareTextSymbolDefault{\textdagger}{TS1}
```

```
124 \DeclareTextSymbolDefault{\textparagraph}{TS1}
```

```
125 \DeclareTextSymbolDefault{\textperiodcentered}{TS1}
```

```
126 \DeclareTextSymbolDefault{\textsection}{TS1}
```

And here are the other TS1 glyphs that are implemented by every font (or nearly every—a few are commented out and moved to sub-encoding 9, because they aren't around in some fonts.

```
127 %%\DeclareTextSymbolDefault{\textbardbl}{TS1} % subst in sub-enc 9 above
```

```
128 \DeclareTextSymbolDefault{\textbrokenbar}{TS1}
```

```
129 %%\DeclareTextSymbolDefault{\textcelsius}{TS1} % subst in sub-enc 9 above
```

```
130 \DeclareTextSymbolDefault{\textcent}{TS1}
```

```

131 \DeclareTextSymbolDefault{\textcopyright}{TS1}
132 \DeclareTextSymbolDefault{\textdegree}{TS1}
133 \DeclareTextSymbolDefault{\textdiv}{TS1}
134 \DeclareTextSymbolDefault{\textlnot}{TS1}
135 \DeclareTextSymbolDefault{\textonehalf}{TS1}
136 \DeclareTextSymbolDefault{\textonequarter}{TS1}
137 %%\DeclareTextSymbolDefault{\textonesuperior}{TS1} % subst in sub-enc 9 above
138 \DeclareTextSymbolDefault{\textordfeminine}{TS1}
139 \DeclareTextSymbolDefault{\textordmasculine}{TS1}
140 \DeclareTextSymbolDefault{\textpm}{TS1}
141 \DeclareTextSymbolDefault{\textquotesingle}{TS1}
142 \DeclareTextSymbolDefault{\textquotestraightbase}{TS1}
143 \DeclareTextSymbolDefault{\textquotestraightdblbase}{TS1}
144 \DeclareTextSymbolDefault{\textregistered}{TS1}
145 %%\DeclareTextSymbolDefault{\textthrequartersemdash}{TS1} % subst in sub-enc 9 above
146 \DeclareTextSymbolDefault{\textthreequarters}{TS1}
147 %%\DeclareTextSymbolDefault{\textthreesuperior}{TS1} % subst in sub-enc 9 above
148 \DeclareTextSymbolDefault{\texttimes}{TS1}
149 \DeclareTextSymbolDefault{\texttrademark}{TS1}
150 %%\DeclareTextSymbolDefault{\texttwelveudash}{TS1} % subst in sub-enc 9 above
151 %%\DeclareTextSymbolDefault{\texttwosuperior}{TS1} % subst in sub-enc 9 above
152 \DeclareTextSymbolDefault{\textyen}{TS1}

153 \DeclareTextSymbolDefault{\textcapitalcompwordmark}{TS1}
154 \DeclareTextSymbolDefault{\textascendercompwordmark}{TS1}

```

In the following sections the remaining default definitions are ordered by sub-encoding in which they are become unavailable (i.e., they are not provided in the sub-encoding with that number and all sub-encodings with higher numbers).

Thus the symbols that are available in sub-encoding x are the symbols above (always available) and the symbols list in the sections for sub-encodings $x + 1$ and higher.

1.1 Sub-encoding 1 (drop symbols not working in Latin Modern)

The `\textcircled` is available but the glyph is simply too small so we keep using the OMS glyph.

```

155 \DeclareTextCommandDefault{\textcircled}
156   {\CheckEncodingSubset\UseTextAccent{TS1}{\UseTextAccent{OMS}}1\textcircled}

```

1.2 Sub-encoding 2 (majority of new OTF fonts via autointst)

```

157 \DeclareTextCommandDefault{\t}
158   {\CheckEncodingSubset\UseTextAccent{TS1}{\UseTextAccent{OML}}2\t}

```

Capital accents are really only very seldom implemented, so from sub-encoding 2 onwards we use the normal T1 accents if they are asked for in the document.

In Unicode engines we don't implement them at all but always use the basic accents instead. whether that works or not really depends on the font, something like `\"X` usually comes out wrong in Unicode engines.

```

159 \ifx\Umathcode\@undefined
160   \DeclareTextCommandDefault{\capitalacute}
161     {\tc@check@accent{\'}2\capitalacute}
162   \DeclareTextCommandDefault{\capitalbreve}
163     {\tc@check@accent{\u}2\capitalbreve}

```

```

164 \DeclareTextCommandDefault{\capitalcaron}
165         {\tc@check@accent{\v}2\capitalcaron}
166 \DeclareTextCommandDefault{\capitalcedilla}
167         {\tc@check@accent{\c}2\capitalcedilla}
168 \DeclareTextCommandDefault{\capitalcircumflex}
169         {\tc@check@accent{\^}2\capitalcircumflex}
170 \DeclareTextCommandDefault{\capitaldieresis}
171         {\tc@check@accent{\"}2\capitaldieresis}
172 \DeclareTextCommandDefault{\capitaldotaccent}
173         {\tc@check@accent{\.}2\capitaldotaccent}
174 \DeclareTextCommandDefault{\capitalgrave}
175         {\tc@check@accent{\`}2\capitalgrave}
176 \DeclareTextCommandDefault{\capitalhungarumlaut}
177         {\tc@check@accent{\H}2\capitalhungarumlaut}
178 \DeclareTextCommandDefault{\capitalmacron}
179         {\tc@check@accent{\=}2\capitalmacron}
180 \DeclareTextCommandDefault{\capitalogonek}
181         {\tc@check@accent{\k}2\capitalogonek}
182 \DeclareTextCommandDefault{\capitalring}
183         {\tc@check@accent{\r}2\capitalring}
184 \DeclareTextCommandDefault{\capitaltie}
185         {\tc@check@accent{\t}2\capitaltie}
186 \DeclareTextCommandDefault{\capitaltilde}
187         {\tc@check@accent{\~}2\capitaltilde}

```

For `\newtie` and `\capitalnewtie` this is actually wrong, they should pick up the accent from the substitution font (not done yet).

```

188 \DeclareTextCommandDefault{\newtie}
189         {\tc@check@accent{\t}2\newtie}
190 \DeclareTextCommandDefault{\capitalnewtie}
191         {\tc@check@accent{\t}2\capitalnewtie}

```

In Unicode engines we just execute the simple accents:

```

192 \else
193 \DeclareTextCommandDefault\capitalacute{\@tabacckludge'}
194 \DeclareTextCommandDefault\capitalbreve{\u}
195 \DeclareTextCommandDefault\capitalcaron{\v}
196 \DeclareTextCommandDefault\capitalcedilla{\c}
197 \DeclareTextCommandDefault\capitalcircumflex{\^}
198 \DeclareTextCommandDefault\capitaldieresis{\"}
199 \DeclareTextCommandDefault\capitaldotaccent{\.}
200 \DeclareTextCommandDefault\capitalgrave{\@tabacckludge`}
201 \DeclareTextCommandDefault\capitalhungarumlaut{\H}
202 \DeclareTextCommandDefault\capitalmacron{\@tabacckludge=}
203 \DeclareTextCommandDefault\capitalnewtie{\t}
204 \DeclareTextCommandDefault\capitalogonek{\k}
205 \DeclareTextCommandDefault\capitalring{\r}
206 \DeclareTextCommandDefault\capitaltie{\t}
207 \DeclareTextCommandDefault\capitaltilde{\~}
208 \DeclareTextCommandDefault\newtie{\t}
209 \fi

```

The next two symbols exist in some fonts (faked?), but we ignore that to keep the subsets reasonable compact and most important linear.

```

210 \DeclareTextCommandDefault{\textlbrackdbl}
211         {\tc@check@symbol2\textlbrackdbl}

```



```

212 \DeclareTextCommandDefault{\textrbrackdbl}
213      {\tc@check@symbol2\textrbrackdbl}

    Old style numerals are again in some fonts but using -OsF, etc. is the better approach
    to get them, so we claim they aren't in sub-encoding 2 as that's true for most fonts.

214 \DeclareTextCommandDefault{\texteightoldstyle}
215      {\tc@check@symbol2\texteightoldstyle}
216 \DeclareTextCommandDefault{\textfiveoldstyle}
217      {\tc@check@symbol2\textfiveoldstyle}
218 \DeclareTextCommandDefault{\textfouroldstyle}
219      {\tc@check@symbol2\textfouroldstyle}
220 \DeclareTextCommandDefault{\textnineoldstyle}
221      {\tc@check@symbol2\textnineoldstyle}
222 \DeclareTextCommandDefault{\textoneoldstyle}
223      {\tc@check@symbol2\textoneoldstyle}
224 \DeclareTextCommandDefault{\textsevenoldstyle}
225      {\tc@check@symbol2\textsevenoldstyle}
226 \DeclareTextCommandDefault{\textsixoldstyle}
227      {\tc@check@symbol2\textsixoldstyle}
228 \DeclareTextCommandDefault{\textthreeoldstyle}
229      {\tc@check@symbol2\textthreeoldstyle}
230 \DeclareTextCommandDefault{\texttwooldstyle}
231      {\tc@check@symbol2\texttwooldstyle}
232 \DeclareTextCommandDefault{\textzerooldstyle}
233      {\tc@check@symbol2\textzerooldstyle}

```

The next set of glyphs is special to T_EX fonts (and available with a few older PS fonts supported through virtual fonts), but not any longer in the majority of fonts provided through autost, so we pretend there aren't available in sub-encoding 2 and below.

```

234 \DeclareTextCommandDefault{\textacutedbl}
235      {\tc@check@symbol2\textacutedbl}
236 \DeclareTextCommandDefault{\textasciicute}
237      {\tc@check@symbol2\textasciicute}
238 \DeclareTextCommandDefault{\textasciibreve}
239      {\tc@check@symbol2\textasciibreve}
240 \DeclareTextCommandDefault{\textasciicaron}
241      {\tc@check@symbol2\textasciicaron}
242 \DeclareTextCommandDefault{\textasciidieresis}
243      {\tc@check@symbol2\textasciidieresis}
244 \DeclareTextCommandDefault{\textasciigrave}
245      {\tc@check@symbol2\textasciigrave}
246 \DeclareTextCommandDefault{\textasciimacron}
247      {\tc@check@symbol2\textasciimacron}
248 \DeclareTextCommandDefault{\textgravedbl}
249      {\tc@check@symbol2\textgravedbl}
250 \DeclareTextCommandDefault{\texttildelow}
251      {\tc@check@symbol2\texttildelow}

```

Finally those below are only available in CM-based fonts but in no font that has its origin outside of the T_EX world.

```

252 \DeclareTextCommandDefault{\textbaht}
253      {\tc@check@symbol2\textbaht}
254 \DeclareTextCommandDefault{\textbigcircle}
255      {\tc@check@symbol2\textbigcircle}
256 \DeclareTextCommandDefault{\textborn}
257      {\tc@check@symbol2\textborn}

```

```

258 \DeclareTextCommandDefault{\textcentoldstyle}
259         {\tc@check@symbol2\textcentoldstyle}
260 \DeclareTextCommandDefault{\textcircledP}
261         {\tc@check@symbol2\textcircledP}
262 \DeclareTextCommandDefault{\textcopyleft}
263         {\tc@check@symbol2\textcopyleft}
264 \DeclareTextCommandDefault{\textdblhyphenchar}
265         {\tc@check@symbol2\textdblhyphenchar}
266 \DeclareTextCommandDefault{\textdblhyphen}
267         {\tc@check@symbol2\textdblhyphen}
268 \DeclareTextCommandDefault{\textdied}
269         {\tc@check@symbol2\textdied}
270 \DeclareTextCommandDefault{\textdiscount}
271         {\tc@check@symbol2\textdiscount}
272 \DeclareTextCommandDefault{\textdivorced}
273         {\tc@check@symbol2\textdivorced}
274 \DeclareTextCommandDefault{\textdollaroldstyle}
275         {\tc@check@symbol2\textdollaroldstyle}
276 \DeclareTextCommandDefault{\textguarani}
277         {\tc@check@symbol2\textguarani}
278 \DeclareTextCommandDefault{\textleaf}
279         {\tc@check@symbol2\textleaf}
280 \DeclareTextCommandDefault{\textlquill}
281         {\tc@check@symbol2\textlquill}
282 \DeclareTextCommandDefault{\textmarried}
283         {\tc@check@symbol2\textmarried}
284 \DeclareTextCommandDefault{\textmho}
285         {\tc@check@symbol2\textmho}
286 \DeclareTextCommandDefault{\textmusicalnote}
287         {\tc@check@symbol2\textmusicalnote}
288 \DeclareTextCommandDefault{\textnaira}
289         {\tc@check@symbol2\textnaira}
290 \DeclareTextCommandDefault{\textopenbullet}
291         {\tc@check@symbol2\textopenbullet}
292 \DeclareTextCommandDefault{\textpeso}
293         {\tc@check@symbol2\textpeso}
294 \DeclareTextCommandDefault{\textpilcrow}
295         {\tc@check@symbol2\textpilcrow}
296 \DeclareTextCommandDefault{\textrecipe}
297         {\tc@check@symbol2\textrecipe}
298 \DeclareTextCommandDefault{\textreferencemark}
299         {\tc@check@symbol2\textreferencemark}
300 \DeclareTextCommandDefault{\textrquill}
301         {\tc@check@symbol2\textrquill}
302 \DeclareTextCommandDefault{\textservicemark}
303         {\tc@check@symbol2\textservicemark}
304 \DeclareTextCommandDefault{\textsurd}
305         {\tc@check@symbol2\textsurd}

```

The `\textpertenthousand` also belongs in this group but here we have a choice: in T1 there is a definition for `\textpertenthousand` making the symbol up from % and `\char 24` (twice) but in many fonts that char doesn't exist and the slot is reused for random ligatures. So better not use it because often it is wrong. But pointing to TS1 is also not great as only a few fonts have it as a real symbol, so we get a substitution to

CM or LM.

Alternatively we could just state that the symbol is unavailable in those fonts. For now I substitute.

```
306 \DeclareTextCommandDefault{\textpertenthousand}  
307     {\tc@check@symbol2\textpertenthousand}  
308 \UndeclareTextCommand{\textpertenthousand}{T1}
```

1.3 Sub-encoding 3

Sub-encoding 2 is the one where we loose many symbols. In the higher-numbered sub-encodings we see only a few dropped additionally.

```
309 \DeclareTextCommandDefault{\textlangle}  
310     {\tc@check@symbol3\textlangle}  
311 \DeclareTextCommandDefault{\textrangle}  
312     {\tc@check@symbol3\textrangle}
```

1.4 Sub-encoding 4

```
313 \DeclareTextCommandDefault{\textcolonmonetary}  
314     {\tc@check@symbol4\textcolonmonetary}  
315 \DeclareTextCommandDefault{\textdong}  
316     {\tc@check@symbol4\textdong}  
317 \DeclareTextCommandDefault{\textdownarrow}  
318     {\tc@check@symbol4\textdownarrow}  
319 \DeclareTextCommandDefault{\textleftarrow}  
320     {\tc@check@symbol4\textleftarrow}  
321 \DeclareTextCommandDefault{\textlira}  
322     {\tc@check@symbol4\textlira}  
323 \DeclareTextCommandDefault{\textrightarrow}  
324     {\tc@check@symbol4\textrightarrow}  
325 \DeclareTextCommandDefault{\textuparrow}  
326     {\tc@check@symbol4\textuparrow}  
327 \DeclareTextCommandDefault{\textwon}  
328     {\tc@check@symbol4\textwon}
```

1.5 Sub-encoding 5 (most older PS fonts)

Most older PS fonts (supported in T_EX since the early nineties when virtual fonts became available) are sorted under this sub-encoding. But in reality, many of them don't have all glyphs that should be available in sub-encoding 5. Instead they show little squares, i.e., they produce “tofu” if you are unlucky.

But the coverage is so random that it is impossible to sort them properly and if we tried to ensure that they only typeset those glyphs that are really always available, we would have to put them all into sub-encoding 9; so putting them into 5 is really a compromise.

Modern fonts usually don't typeset a tofu character if a glyph is missing. They are therefore only classified as sub-encoding 5 if they really support its glyph set completely.

```
329 \DeclareTextCommandDefault{\textestimated}  
330     {\tc@check@symbol5\textestimated}  
331 \DeclareTextCommandDefault{\textnumero}  
332     {\tc@check@symbol5\textnumero}
```

1.6 Sub-encoding 6

```
333 \DeclareTextCommandDefault{\textflorin}
334         {\tc@check@symbol6\textflorin}
335 \DeclareTextCommandDefault{\textcurrency}
336         {\tc@check@symbol6\textcurrency}
```

1.7 Sub-encoding 7

```
337 \DeclareTextCommandDefault{\textfractionsolidus}
338         {\tc@check@symbol7\textfractionsolidus}
339 \DeclareTextCommandDefault{\textohm}
340         {\tc@check@symbol7\textohm}
341 \DeclareTextCommandDefault{\textmu}
342         {\tc@check@symbol7\textmu}
343 \DeclareTextCommandDefault{\textminus}
344         {\tc@check@symbol7\textminus}
```

1.8 Sub-encoding 8

```
345 \DeclareTextCommandDefault{\textblank}
346         {\tc@check@symbol{8}\textblank}
347 \DeclareTextCommandDefault{\textinterrobangdown}
348         {\tc@check@symbol{8}\textinterrobangdown}
349 \DeclareTextCommandDefault{\textinterrobang}
350         {\tc@check@symbol{8}\textinterrobang}
```

Fonts with this sub-encoding don't have a Euro symbol, but instead of substituting we fake it.

```
351 \DeclareTextCommandDefault{\texteuro}
352     {\CheckEncodingSubset\UseTextSymbol{TS1}\tc@fake@euro{8}\texteuro}
```

1.9 Sub-encoding 9 (most missing)

```
353 \DeclareTextCommandDefault{\textcelsius}
354         {\tc@check@symbol{9}\textcelsius}
355 \DeclareTextCommandDefault{\textonesuperior}
356         {\tc@check@symbol{9}\textonesuperior}
357 \DeclareTextCommandDefault{\textthreequartersemdash}
358         {\tc@check@symbol{9}\textthreequartersemdash}
359 \DeclareTextCommandDefault{\textthreesuperior}
360         {\tc@check@symbol{9}\textthreesuperior}
361 \DeclareTextCommandDefault{\texttwelveudash}
362         {\tc@check@symbol{9}\texttwelveudash}
363 \DeclareTextCommandDefault{\texttwosuperior}
364         {\tc@check@symbol{9}\texttwosuperior}
365 \DeclareTextCommandDefault{\textbardbl}
366         {\tc@check@symbol{9}\textbardbl}
```

2 Unicode engine specials

If we are using a unicode engine we handle some glyphs differently, so this here are the definitions for the Unicode encoding (overwriting the defaults above).

```
367 \ifx \Umathcode\@undefined \else
```

This set should be taken from TS1 encoding even if it means you get it from the default font for that encoding.

```

368 %\DeclareTextSymbol{\textcopyleft}{TS1}{171}
369 %\DeclareTextSymbol{\textdblhyphen}{TS1}{45}
370 %\DeclareTextSymbol{\textdblhyphenchar}{TS1}{127}
371 %\DeclareTextSymbol{\textquotestraightbase}{TS1}{13}
372 %\DeclareTextSymbol{\textquotestraightdblbase}{TS1}{18}
373 %\DeclareTextSymbol{\textleaf}{TS1}{108}
374 %\DeclareTextSymbol{\texttwelvewardash}{TS1}{21}
375 %\DeclareTextSymbol{\textthreequartersemdash}{TS1}{22}

```

If oldstyle numerals are asked for we just use `\oldstylenums`.

```

376 \DeclareTextCommand{\textzerooldstyle} \UnicodeEncodingName{\oldstylenums{0}}
377 \DeclareTextCommand{\textoneoldstyle} \UnicodeEncodingName{\oldstylenums{1}}
378 \DeclareTextCommand{\texttwooldstyle} \UnicodeEncodingName{\oldstylenums{2}}
379 \DeclareTextCommand{\textthreeoldstyle} \UnicodeEncodingName{\oldstylenums{3}}
380 \DeclareTextCommand{\textfouroldstyle} \UnicodeEncodingName{\oldstylenums{4}}
381 \DeclareTextCommand{\textfiveoldstyle} \UnicodeEncodingName{\oldstylenums{5}}
382 \DeclareTextCommand{\textsixoldstyle} \UnicodeEncodingName{\oldstylenums{6}}
383 \DeclareTextCommand{\textsevenoldstyle} \UnicodeEncodingName{\oldstylenums{7}}
384 \DeclareTextCommand{\texteightoldstyle} \UnicodeEncodingName{\oldstylenums{8}}
385 \DeclareTextCommand{\textnineoldstyle} \UnicodeEncodingName{\oldstylenums{9}}

```

These have Unicode slots so this should be integrated into TU explicitly

```

386 \DeclareTextSymbol{\textpilcrow} \UnicodeEncodingName{"00B6}
387 \DeclareTextSymbol{\textborn} \UnicodeEncodingName{"002A}
388 \DeclareTextSymbol{\textdied} \UnicodeEncodingName{"2020}
389 \DeclareTextSymbol{\textlbrackdbl} \UnicodeEncodingName{"27E6}
390 \DeclareTextSymbol{\textrbrackdbl} \UnicodeEncodingName{"27E7}
391 \DeclareTextSymbol{\textguarani} \UnicodeEncodingName{"20B2}

```

We could make `\textcentoldstyle` and `\textdollaroldstyle` point to dollar and cent in the Unicode encoding

```

392 %\DeclareTextSymbol{\textcentoldstyle} \UnicodeEncodingName{"00A2}
393 %\DeclareTextSymbol{\textdollaroldstyle} \UnicodeEncodingName{"0024}

```

but I think it is better to pick them up from TS1 even if that usually means LMR fonts

```

394 \DeclareTextSymbol{\textdollaroldstyle}{TS1}{138}
395 \DeclareTextSymbol{\textcentoldstyle} {TS1}{139}

```

```

396 \fi % --- END of Unicode engines specials

```

3 Font family sub-encodings setup

We declare the subsets for a good number of fonts in the kernel ...

But first the default for anything that is not declared. We use 9 which is most likely much too conservative, but with the advantage that we aren't getting missing glyphs (or at least that this is very unlikely). For nearly all font in the T_EX Live distribution of 2019 "correct" classifications are given below, so that this default is only used for new font families, and over time the right classifications can be added here too.

```

397 \DeclareEncodingSubset{TS1}{?}{9}

```

This first block contains the fonts that have been already supported by the `textcomp` package way back, i.e., the font families that have \TeX support since the mid-nineties.

```

398 \DeclareEncodingSubset{TS1}{ccr}      {0}
399 \DeclareEncodingSubset{TS1}{cmbr}     {0}
400 \DeclareEncodingSubset{TS1}{cmr}      {0}
401 \DeclareEncodingSubset{TS1}{cmss}     {0}
402 \DeclareEncodingSubset{TS1}{cmtl}     {0}
403 \DeclareEncodingSubset{TS1}{cmtt}     {0}
404 \DeclareEncodingSubset{TS1}{cmvtt}    {0}
405 \DeclareEncodingSubset{TS1}{pxr}      {0}
406 \DeclareEncodingSubset{TS1}{pxss}     {0}
407 \DeclareEncodingSubset{TS1}{pxtt}     {0}
408 \DeclareEncodingSubset{TS1}{qag}      {0}
409 \DeclareEncodingSubset{TS1}{qbk}      {0}
410 \DeclareEncodingSubset{TS1}{qcr}      {0}
411 \DeclareEncodingSubset{TS1}{qcs}      {0}
412 \DeclareEncodingSubset{TS1}{qhvc}     {0}
413 \DeclareEncodingSubset{TS1}{qhv}      {0}
414 \DeclareEncodingSubset{TS1}{qpl}      {0}
415 \DeclareEncodingSubset{TS1}{qtm}      {0}
416 \DeclareEncodingSubset{TS1}{qzc}      {0}
417 \DeclareEncodingSubset{TS1}{txr}      {0}
418 \DeclareEncodingSubset{TS1}{txss}     {0}
419 \DeclareEncodingSubset{TS1}{txtt}     {0}

420 \DeclareEncodingSubset{TS1}{lmr}      {1}
421 \DeclareEncodingSubset{TS1}{lmdh}     {1}
422 \DeclareEncodingSubset{TS1}{lmss}     {1}
423 \DeclareEncodingSubset{TS1}{lmssq}    {1}
424 \DeclareEncodingSubset{TS1}{lmvtt}    {1}
425 \DeclareEncodingSubset{TS1}{lmtt}     {1} % missing TM, SM and
426                                     % pertenthousand for some reason

427 \DeclareEncodingSubset{TS1}{ptmx}     {2}
428 \DeclareEncodingSubset{TS1}{ptmj}     {2}
429 \DeclareEncodingSubset{TS1}{ul8}      {2}

430 \DeclareEncodingSubset{TS1}{bch} {5} % tofu for blank, ohm
431 \DeclareEncodingSubset{TS1}{futj} {5} % tofu for blank, interrobang/down, ohm
432 \DeclareEncodingSubset{TS1}{futs} {5} % tofu for blank, ohm
433 \DeclareEncodingSubset{TS1}{futsx} {5} % probably (currently broken distrib)
434 \DeclareEncodingSubset{TS1}{pag} {5} % tofu for blank, interrobang/down, ohm
435 \DeclareEncodingSubset{TS1}{pbk} {5} % tofu for blank, interrobang/down, ohm
436 \DeclareEncodingSubset{TS1}{pcr} {5} % tofu for blank, interrobang/down, ohm
437 \DeclareEncodingSubset{TS1}{phv} {5} % tofu for blank, interrobang/down, ohm
438 \DeclareEncodingSubset{TS1}{pnc} {5} % tofu for blank, interrobang/down, ohm
439 \DeclareEncodingSubset{TS1}{pplj} {5} % tofu for blank
440 \DeclareEncodingSubset{TS1}{pplx} {5} % tofu for blank
441 \DeclareEncodingSubset{TS1}{ppl} {5} % tofu for blank interrobang/down
442 \DeclareEncodingSubset{TS1}{ptm} {5} % tofu for blank, interrobang/down, ohm
443 \DeclareEncodingSubset{TS1}{pzc} {5} % tofu for blank, interrobang/down, ohm
444 \DeclareEncodingSubset{TS1}{ul9} {5} % tofu for blank, interrobang/down, ohm

445 \DeclareEncodingSubset{TS1}{dayroms} {6} % tofu for blank, interrobang/down, ohm
446 \DeclareEncodingSubset{TS1}{dayrom} {6} % tofu for blank, interrobang/down, ohm
447 \DeclareEncodingSubset{TS1}{augie} {8} % really only missing euro

```

```

448 \DeclareEncodingSubset{TS1}{put} {8}
449 \DeclareEncodingSubset{TS1}{uag} {8} % probably (currently broken distrib)
450 \DeclareEncodingSubset{TS1}{ugq} {8}
451 \DeclareEncodingSubset{TS1}{zi4} {9}

```

LucidaBright (sold through TUG) probably not quite correct, I guess as I have the older fonts ...

```

452 \DeclareEncodingSubset{TS1}{hls} {5}
453 \DeclareEncodingSubset{TS1}{hlst} {5}
454 \DeclareEncodingSubset{TS1}{hlct} {5}
455 \DeclareEncodingSubset{TS1}{hlh} {5}
456 \DeclareEncodingSubset{TS1}{hlx} {8}
457 \DeclareEncodingSubset{TS1}{hlce} {8}
458 \DeclareEncodingSubset{TS1}{hlcn} {8}
459 \DeclareEncodingSubset{TS1}{hlcw} {8}
460 \DeclareEncodingSubset{TS1}{hlcf} {8}

```

Below are the newer fonts that have support files for L^AT_EX. With very few exceptions the classifications are done so that all characters are correctly produced (either being available in the font or substituted).

There are a few fonts that contain “tofu” squares in places (instead of a real glyph) and in a few cases some really seldom needed chars are unavailable, i.e., produce missing glyphs (to avoid that a large number of available chars are unnecessarily substituted).

```

461 \DeclareEncodingSubset{TS1}{lato-*} {0} % with a bunch of tofu inside
462 \DeclareEncodingSubset{TS1}{opensans-*} {0} % with a bunch of tofu inside
463 \DeclareEncodingSubset{TS1}{cantarell-*} {0} % with a bunch of tofu inside
464 \DeclareEncodingSubset{TS1}{fbb-*} {0} % missing centoldstyle
465 \DeclareEncodingSubset{TS1}{tli} {1} % with lots of tofu inside

466 \DeclareEncodingSubset{TS1}{Alegreya-*} {2}
467 \DeclareEncodingSubset{TS1}{AlegreyaSans-*} {2}
468 \DeclareEncodingSubset{TS1}{DejaVuSans-TLF} {2}
469 \DeclareEncodingSubset{TS1}{DejaVuSansCondensed-TLF} {2}
470 \DeclareEncodingSubset{TS1}{DejaVuSansMono-TLF} {2}
471 \DeclareEncodingSubset{TS1}{EBGaramond-*} {2}
472 \DeclareEncodingSubset{TS1}{Tempora-TLF} {2}
473 \DeclareEncodingSubset{TS1}{Tempora-T0sF} {2}

474 \DeclareEncodingSubset{TS1}{Arimo-TLF} {3}
475 \DeclareEncodingSubset{TS1}{Carlito-*} {3}
476 \DeclareEncodingSubset{TS1}{FiraSans-*} {3}
477 \DeclareEncodingSubset{TS1}{IBMPlexSans-TLF} {3}
478 \DeclareEncodingSubset{TS1}{Merriweather-0sF} {3}
479 \DeclareEncodingSubset{TS1}{Montserrat-*} {3}
480 \DeclareEncodingSubset{TS1}{MontserratAlternates-*} {3}
481 \DeclareEncodingSubset{TS1}{SourceCodePro-TLF} {3}
482 \DeclareEncodingSubset{TS1}{SourceCodePro-T0sF} {3}
483 \DeclareEncodingSubset{TS1}{SourceSansPro-*} {3}
484 \DeclareEncodingSubset{TS1}{SourceSerifPro-*} {3}
485 \DeclareEncodingSubset{TS1}{Tinos-TLF} {3}

486 \DeclareEncodingSubset{TS1}{AccanthisADFStdNoThree-LF} {4}
487 \DeclareEncodingSubset{TS1}{Cabin-TLF} {4}
488 \DeclareEncodingSubset{TS1}{Caladea-TLF} {4}
489 \DeclareEncodingSubset{TS1}{Chivo-*} {4}

```

```

490 \DeclareEncodingSubset{TS1}{ClearSans-TLF} {4}
491 \DeclareEncodingSubset{TS1}{Coelacanth-LF} {4}
492 \DeclareEncodingSubset{TS1}{CrimsonPro-*} {4}
493 \DeclareEncodingSubset{TS1}{FiraMono-TLF} {4}
494 \DeclareEncodingSubset{TS1}{FiraMono-T0sF} {4}
495 \DeclareEncodingSubset{TS1}{Go-TLF} {4}
496 \DeclareEncodingSubset{TS1}{GoMono-TLF} {4}
497 \DeclareEncodingSubset{TS1}{InriaSans-*} {4}
498 \DeclareEncodingSubset{TS1}{InriaSerif-*} {4}
499 \DeclareEncodingSubset{TS1}{LibertinusSans-*} {4}
500 \DeclareEncodingSubset{TS1}{LibertinusSerif-*} {4}
501 \DeclareEncodingSubset{TS1}{LibreBodoni-TLF} {4}
502 \DeclareEncodingSubset{TS1}{LibreFranklin-TLF} {4}
503 \DeclareEncodingSubset{TS1}{LinguisticsPro-LF} {4}
504 \DeclareEncodingSubset{TS1}{LinguisticsPro-0sF} {4}
505 \DeclareEncodingSubset{TS1}{LinuxBiolumT-*} {4}
506 \DeclareEncodingSubset{TS1}{LinuxLibertineT-*} {4}
507 \DeclareEncodingSubset{TS1}{MerriweatherSans-0sF} {4}
508 \DeclareEncodingSubset{TS1}{MintSpirit-*} {4}
509 \DeclareEncodingSubset{TS1}{MintSpiritNoTwo-*} {4}
510 \DeclareEncodingSubset{TS1}{PTMono-TLF} {4}
511 \DeclareEncodingSubset{TS1}{PTSans-TLF} {4}
512 \DeclareEncodingSubset{TS1}{PTSansCaption-TLF} {4}
513 \DeclareEncodingSubset{TS1}{PTSansNarrow-TLF} {4}
514 \DeclareEncodingSubset{TS1}{PTSerif-TLF} {4}
515 \DeclareEncodingSubset{TS1}{PTSerifCaption-TLF} {4}
516 \DeclareEncodingSubset{TS1}{Raleway-TLF} {4}
517 \DeclareEncodingSubset{TS1}{Raleway-T0sF} {4}
518 \DeclareEncodingSubset{TS1}{Roboto-*} {4}
519 \DeclareEncodingSubset{TS1}{RobotoMono-TLF} {4}
520 \DeclareEncodingSubset{TS1}{RobotoSlab-TLF} {4}
521 \DeclareEncodingSubset{TS1}{Rosario-*} {4}
522 \DeclareEncodingSubset{TS1}{SticksTooText-*} {4}
523 \DeclareEncodingSubset{TS1}{UniversalisADFSd-LF} {4}

524 \DeclareEncodingSubset{TS1}{Almendra-0sF} {5}
525 \DeclareEncodingSubset{TS1}{Baskervaldx-*} {5}
526 \DeclareEncodingSubset{TS1}{BaskervilleF-*} {5}
527 \DeclareEncodingSubset{TS1}{Bitter-TLF} {5}
528 \DeclareEncodingSubset{TS1}{Cinzel-LF} {5}
529 \DeclareEncodingSubset{TS1}{CinzelDecorative-LF} {5}
530 \DeclareEncodingSubset{TS1}{DejaVuSerif-TLF} {5}
531 \DeclareEncodingSubset{TS1}{DejaVuSerifCondensed-TLF} {5}
532 \DeclareEncodingSubset{TS1}{GilliusADF-LF} {5}
533 \DeclareEncodingSubset{TS1}{GilliusADFCd-LF} {5}
534 \DeclareEncodingSubset{TS1}{GilliusADFNoTwo-LF} {5}
535 \DeclareEncodingSubset{TS1}{GilliusADFNoTwoCond-LF} {5}
536 \DeclareEncodingSubset{TS1}{LobsterTwo-LF} {5}
537 \DeclareEncodingSubset{TS1}{OldStandard-TLF} {5}
538 \DeclareEncodingSubset{TS1}{PlayfairDisplay-TLF} {5}
539 \DeclareEncodingSubset{TS1}{PlayfairDisplay-T0sF} {5}
540 \DeclareEncodingSubset{TS1}{TheanoDidot-TLF} {5}
541 \DeclareEncodingSubset{TS1}{TheanoDidot-T0sF} {5}
542 \DeclareEncodingSubset{TS1}{TheanoModern-TLF} {5}
543 \DeclareEncodingSubset{TS1}{TheanoModern-T0sF} {5}

```



```

544 \DeclareEncodingSubset{TS1}{TheanoOldStyle-TLF}      {5}
545 \DeclareEncodingSubset{TS1}{TheanoOldStyle-T0sF}     {5}

546 \DeclareEncodingSubset{TS1}{Crimson-TLF}             {6}
547 \DeclareEncodingSubset{TS1}{IBMPlexMono-TLF}         {6}
548 \DeclareEncodingSubset{TS1}{IBMPlexSerif-TLF}        {6}
549 \DeclareEncodingSubset{TS1}{LibertinusMono-TLF}      {6}
550 \DeclareEncodingSubset{TS1}{LibertinusSerifDisplay-LF}{6}
551 \DeclareEncodingSubset{TS1}{LinuxLibertineDisplayT-*} {6}
552 \DeclareEncodingSubset{TS1}{LinuxLibertineMonoT-LF}  {6}
553 \DeclareEncodingSubset{TS1}{LinuxLibertineMonoT-TLF} {6}
554 \DeclareEncodingSubset{TS1}{Overlock-LF}             {6}

555 \DeclareEncodingSubset{TS1}{CormorantGaramond-*}     {7}
556 \DeclareEncodingSubset{TS1}{Heuristica-TLF}         {7}
557 \DeclareEncodingSubset{TS1}{Heuristica-T0sF}        {7}
558 \DeclareEncodingSubset{TS1}{IMFELLEnglish-TLF}      {7}
559 \DeclareEncodingSubset{TS1}{LibreBaskerville-TLF}    {7}
560 \DeclareEncodingSubset{TS1}{LibreCaslon-*}           {7}
561 \DeclareEncodingSubset{TS1}{Marcellus-LF}            {7}
562 \DeclareEncodingSubset{TS1}{NotoSans-*}              {7}
563 \DeclareEncodingSubset{TS1}{NotoSansMono-TLF}        {7}
564 \DeclareEncodingSubset{TS1}{NotoSansMono-T0sF}      {7}
565 \DeclareEncodingSubset{TS1}{NotoSerif-*}             {7}
566 \DeclareEncodingSubset{TS1}{Quattrocento-TLF}       {7}
567 \DeclareEncodingSubset{TS1}{QuattrocentoSans-TLF}   {7}
568 \DeclareEncodingSubset{TS1}{XCharter-TLF}           {7}
569 \DeclareEncodingSubset{TS1}{XCharter-T0sF}          {7}
570 \DeclareEncodingSubset{TS1}{erewhon-*}              {7}
571 \DeclareEncodingSubset{TS1}{ComicNeue-TLF}          {7}
572 \DeclareEncodingSubset{TS1}{ComicNeueAngular-TLF}   {7}
573 \DeclareEncodingSubset{TS1}{Forum-LF}               {7} % the superiors are missing
574 \DeclareEncodingSubset{TS1}{Cochineal-*}            {8}
575 \DeclareEncodingSubset{TS1}{AlgolRevived-TLF}       {9}

```

4 Legacy symbol support for lists and footnote symbols

`\UseLegacyTextSymbols`

```

576 \def\UseLegacyTextSymbols{%
577   \DeclareTextSymbolDefault{\textasteriskcentered}{OMS}%
578   \DeclareTextSymbolDefault{\textbardbl}{OMS}%
579   \DeclareTextSymbolDefault{\textbullet}{OMS}%
580   \DeclareTextSymbolDefault{\textdaggerdbl}{OMS}%
581   \DeclareTextSymbolDefault{\textdagger}{OMS}%
582   \DeclareTextSymbolDefault{\textparagraph}{OMS}%
583   \DeclareTextSymbolDefault{\textperiodcentered}{OMS}%
584   \DeclareTextSymbolDefault{\textsection}{OMS}%
585   \UndeclareTextCommand{\textsection}{T1}%
586   \expandafter\let\csname oldstylenums \expandafter\endcsname
587     \csname legacyoldstylenums \endcsname
588 }

```

(End definition for `\UseLegacyTextSymbols`.)

`\textlegacyasteriskcentered` Here are new names for the legacy symbols that L^AT_EX used to pick up from the OMS encoded fonts (and used for itemize lists or footnote symbols).

`\textlegacybardbl` We go the roundabout way via separate OMS declarations so that

`\textlegacybullet` `\renewcommand\textbullet{\textlegacybullet}`

`\textlegacydaggerdbl` doesn't produce an endless loop.

`\textlegacydagger`

`\textlegacyparagraph`

`\textlegacyperiodcentered`

`\textlegacysection`

```
589 \DeclareTextSymbol{\textlegacyasteriskcentered}{OMS}{3} % "03
590 \DeclareTextSymbol{\textlegacybardbl}{OMS}{107} % "6B
591 \DeclareTextSymbol{\textlegacybullet}{OMS}{15} % "0F
592 \DeclareTextSymbol{\textlegacydaggerdbl}{OMS}{122} % "7A
593 \DeclareTextSymbol{\textlegacydagger}{OMS}{121} % "79
594 \DeclareTextSymbol{\textlegacyparagraph}{OMS}{123} % "7B
595 \DeclareTextSymbol{\textlegacyperiodcentered}{OMS}{1} % "01
596 \DeclareTextSymbol{\textlegacysection}{OMS}{120} % "78
```

```
597 \DeclareTextSymbolDefault{\textlegacyasteriskcentered}{OMS}
598 \DeclareTextSymbolDefault{\textlegacybardbl}{OMS}
599 \DeclareTextSymbolDefault{\textlegacybullet}{OMS}
600 \DeclareTextSymbolDefault{\textlegacydaggerdbl}{OMS}
601 \DeclareTextSymbolDefault{\textlegacydagger}{OMS}
602 \DeclareTextSymbolDefault{\textlegacyparagraph}{OMS}
603 \DeclareTextSymbolDefault{\textlegacyperiodcentered}{OMS}
604 \DeclareTextSymbolDefault{\textlegacysection}{OMS}
```

(End definition for `\textlegacyasteriskcentered` and others.)

Supporting rollback ...

```
605 </2ekernel | latexrelease>
606 <latexrelease>
607 <latexrelease>\IncludeInRelease{0000/00/00}%
608 <latexrelease> {ltxtextcomp}{\Undefined text companion symbols}%
609 <latexrelease>
610 <latexrelease>\DeclareRobustCommand\oldstylenums[1]{%
611 <latexrelease> \begingroup
612 <latexrelease> \spaceskip\fontdimen\tw@font
613 <latexrelease> \usefont{OML}{\rmdefault}{\f@series}{it}%
614 <latexrelease> \mathgroup\symletters #1%
615 <latexrelease> \endgroup
616 <latexrelease>}
617 <latexrelease>\let\legacyoldstylenums\@undefined
618 <latexrelease>\def\textcompsubstdefault{cmr}
619 <latexrelease>
620 <latexrelease>\let\DeclareEncodingSubset\@undefined
621 <latexrelease>\let\CheckEncodingSubset\@undefined
622 <latexrelease>
623 <latexrelease>\DeclareTextSymbolDefault{\textdollar}{OT1}
624 <latexrelease>\DeclareTextSymbolDefault{\textsterling}{OT1}
625 <latexrelease>\DeclareTextCommand{\textdollar}{OT1}{\hmode\bgroup
626 <latexrelease> \ifdim \fontdimen\@ne\font >\z@
627 <latexrelease> \slshape
628 <latexrelease> \else
629 <latexrelease> \upshape
630 <latexrelease> \fi
```

```

631 <latexrelease> \char'{$\egroup}
632 <latexrelease>\DeclareTextCommand{\textsterling}{OT1}{\hmode@bgroup
633 <latexrelease> \ifdim \fontdimen\@ne\font >\z@
634 <latexrelease> \itshape
635 <latexrelease> \else
636 <latexrelease> \fontshape{ui}\selectfont
637 <latexrelease> \fi
638 <latexrelease> \char'{$\egroup}
639 <latexrelease>\DeclareTextCommand{\textperthousand}{T1}
640 <latexrelease> {\%\char 24 }
641 <latexrelease>
642 <latexrelease>\DeclareTextSymbolDefault{\textasteriskcentered}{OMS}
643 <latexrelease>\DeclareTextSymbolDefault{\textbullet}{OMS}
644 <latexrelease>\DeclareTextSymbolDefault{\textdaggerdbl}{OMS}
645 <latexrelease>\DeclareTextSymbolDefault{\textdagger}{OMS}
646 <latexrelease>\DeclareTextSymbolDefault{\textparagraph}{OMS}
647 <latexrelease>\DeclareTextSymbolDefault{\textperiodcentered}{OMS}
648 <latexrelease>\DeclareTextSymbolDefault{\textsection}{OMS}
649 <latexrelease>
650 <latexrelease>\DeclareTextSymbolDefault{\textbardbl}{OMS}
651 <latexrelease>\let\textbrokenbar\@undefined
652 <latexrelease>\let\textcelsius\@undefined
653 <latexrelease>\let\textcent\@undefined
654 <latexrelease>\DeclareTextCommandDefault{\textcopyright}
655 <latexrelease> {\textcircled{c}}
656 <latexrelease>\let\textdegree\@undefined
657 <latexrelease>\let\textdiv\@undefined
658 <latexrelease>\let\textlnot\@undefined
659 <latexrelease>\let\textonehalf\@undefined
660 <latexrelease>\let\textonequarter\@undefined
661 <latexrelease>\let\textonesuperior\@undefined
662 <latexrelease>\DeclareTextCommandDefault{\textordfeminine}
663 <latexrelease> {\textsuperscript{a}}
664 <latexrelease>\DeclareTextCommandDefault{\textordmasculine}
665 <latexrelease> {\textsuperscript{o}}
666 <latexrelease>\let\textpm\@undefined
667 <latexrelease>\let\textquotesingle\@undefined
668 <latexrelease>\let\textquotestraightbase\@undefined
669 <latexrelease>\let\textquotestraightdblbase\@undefined
670 <latexrelease>\DeclareTextCommandDefault{\textregistered}
671 <latexrelease> {\textcircled{%
672 <latexrelease> \check@mathfonts\fontsize\sf@size\z@
673 <latexrelease> \math@fontsfalse\selectfont R}}
674 <latexrelease>\let\textthreequartersemdash\@undefined
675 <latexrelease>\let\textthreequarters\@undefined
676 <latexrelease>\let\textthreesuperior\@undefined
677 <latexrelease>\let\texttimes\@undefined
678 <latexrelease>\DeclareTextCommandDefault{\texttrademark}
679 <latexrelease> {\textsuperscript{TM}}
680 <latexrelease>\let\texttwelveudash\@undefined
681 <latexrelease>\let\texttwosuperior\@undefined
682 <latexrelease>\let\textyen\@undefined
683 <latexrelease>
684 <latexrelease>\let\textcapitalcompwordmark\@undefined

```

```

685 <latexrelease>\let\textascendercompwordmark\@undefined
686 <latexrelease>
687 <latexrelease>\DeclareTextAccentDefault{\textcircled}{OMS}
688 <latexrelease>\DeclareTextAccentDefault{\t}{OML}
689 <latexrelease>
690 <latexrelease>\let\capitalacute\@undefined
691 <latexrelease>\let\capitalbreve\@undefined
692 <latexrelease>\let\capitalcaron\@undefined
693 <latexrelease>\let\capitalcedilla\@undefined
694 <latexrelease>\let\capitalcircumflex\@undefined
695 <latexrelease>\let\capitaldieresis\@undefined
696 <latexrelease>\let\capitaldotaccent\@undefined
697 <latexrelease>\let\capitalgrave\@undefined
698 <latexrelease>\let\capitalhungarumlaut\@undefined
699 <latexrelease>\let\capitalmacron\@undefined
700 <latexrelease>\let\capitalnewtie\@undefined
701 <latexrelease>\let\capitalogonek\@undefined
702 <latexrelease>\let\capitalring\@undefined
703 <latexrelease>\let\capitaltie\@undefined
704 <latexrelease>\let\capitaltilde\@undefined
705 <latexrelease>\let\newtie\@undefined
706 <latexrelease>
707 <latexrelease>\let\textlbrackdbl\@undefined
708 <latexrelease>\let\textrrackdbl\@undefined
709 <latexrelease>
710 <latexrelease>\let\texteightoldstyle\@undefined
711 <latexrelease>\let\textfiveoldstyle\@undefined
712 <latexrelease>\let\textfouroldstyle\@undefined
713 <latexrelease>\let\textnineoldstyle\@undefined
714 <latexrelease>\let\textoneoldstyle\@undefined
715 <latexrelease>\let\textsevenoldstyle\@undefined
716 <latexrelease>\let\textsixoldstyle\@undefined
717 <latexrelease>\let\textthreeoldstyle\@undefined
718 <latexrelease>\let\texttwooldstyle\@undefined
719 <latexrelease>\let\textzerooldstyle\@undefined
720 <latexrelease>
721 <latexrelease>\let\textacutedbl\@undefined
722 <latexrelease>\let\textasciicute\@undefined
723 <latexrelease>\let\textasciibreve\@undefined
724 <latexrelease>\let\textasciicaron\@undefined
725 <latexrelease>\let\textasciidieresis\@undefined
726 <latexrelease>\let\textasciigrave\@undefined
727 <latexrelease>\let\textasciimacron\@undefined
728 <latexrelease>\let\textgravedbl\@undefined
729 <latexrelease>\let\texttildelow\@undefined
730 <latexrelease>
731 <latexrelease>\let\textbaht\@undefined
732 <latexrelease>\let\textbigcircle\@undefined
733 <latexrelease>\let\textborn\@undefined
734 <latexrelease>\let\textcentoldstyle\@undefined
735 <latexrelease>\let\textcircledP\@undefined
736 <latexrelease>\let\textcopyleft\@undefined
737 <latexrelease>\let\textdblhyphenchar\@undefined
738 <latexrelease>\let\textdblhyphen\@undefined

```

```

739 <latexrelease>\let\textdied\@undefined
740 <latexrelease>\let\textdiscount\@undefined
741 <latexrelease>\let\textdivorced\@undefined
742 <latexrelease>\let\textdollaroldstyle\@undefined
743 <latexrelease>\let\textguarani\@undefined
744 <latexrelease>\let\textleaf\@undefined
745 <latexrelease>\let\textlquill\@undefined
746 <latexrelease>\let\textmarried\@undefined
747 <latexrelease>\let\textmho\@undefined
748 <latexrelease>\let\textmusicalnote\@undefined
749 <latexrelease>\let\textnaira\@undefined
750 <latexrelease>\let\textopenbullet\@undefined
751 <latexrelease>\let\textpeso\@undefined
752 <latexrelease>\let\textpilcrow\@undefined
753 <latexrelease>\let\textrecipe\@undefined
754 <latexrelease>\let\textreferencemark\@undefined
755 <latexrelease>\let\texttrquill\@undefined
756 <latexrelease>\let\textservicemark\@undefined
757 <latexrelease>\let\textsurd\@undefined
758 <latexrelease>
759 <latexrelease>\DeclareTextCommand{\textpertenthousand}{T1}
760 <latexrelease>                {\%\char 24\char 24 }
761 <latexrelease>
762 <latexrelease>\let\textlangle\@undefined
763 <latexrelease>\let\textrangle\@undefined
764 <latexrelease>
765 <latexrelease>\let\textcolonmonetary\@undefined
766 <latexrelease>\let\textdong\@undefined
767 <latexrelease>\let\textdownarrow\@undefined
768 <latexrelease>\let\textleftarrow\@undefined
769 <latexrelease>\let\textlira\@undefined
770 <latexrelease>\let\textrightarrow\@undefined
771 <latexrelease>\let\textuparrow\@undefined
772 <latexrelease>\let\textwon\@undefined
773 <latexrelease>
774 <latexrelease>\let\textestimated\@undefined
775 <latexrelease>\let\textnumero\@undefined
776 <latexrelease>
777 <latexrelease>\let\textflorin\@undefined
778 <latexrelease>\let\textcurrency\@undefined
779 <latexrelease>
780 <latexrelease>\let\textfractionsolidus\@undefined
781 <latexrelease>\let\textohm\@undefined
782 <latexrelease>\let\textmu\@undefined
783 <latexrelease>\let\textminus\@undefined
784 <latexrelease>
785 <latexrelease>\let\textblank\@undefined
786 <latexrelease>\let\textinterrobangdown\@undefined
787 <latexrelease>\let\textinterrobang\@undefined
788 <latexrelease>
789 <latexrelease>\let\texteuro\@undefined
790 <latexrelease>
791 <latexrelease>\let\textcelsius\@undefined
792 <latexrelease>\let\tonesuperior\@undefined

```

```

793 <latexrelease>\let\textthreequartersemdash\@undefined
794 <latexrelease>\let\textthreesuperior\@undefined
795 <latexrelease>\let\texttwelveudash\@undefined
796 <latexrelease>\let\texttwosuperior\@undefined
797 <latexrelease>\let\textbardbl\@undefined
798 <latexrelease>
799 <latexrelease>\let\UseLegacyTextSymbols\@undefined
800 <latexrelease>\let\textlegacyasteriskcentered\@undefined
801 <latexrelease>\let\textlegacybardbl\@undefined
802 <latexrelease>\let\textlegacybullet\@undefined
803 <latexrelease>\let\textlegacydaggerdbl\@undefined
804 <latexrelease>\let\textlegacydagger\@undefined
805 <latexrelease>\let\textlegacyparagraph\@undefined
806 <latexrelease>\let\textlegacyperiodcentered\@undefined
807 <latexrelease>\let\textlegacysection\@undefined
808 <latexrelease>
809 <latexrelease>\EndModuleRelease

```

5 The textcomp package

```

810 <*TS1sty>
811 \providecommand\DeclareRelease[3]{}
812 \providecommand\DeclareCurrentRelease[2]{}
813
814 \DeclareRelease{}{2018-08-11}{textcomp-2018-08-11.sty}
815 \DeclareCurrentRelease{}{2020-02-02}
816
817 \ProvidesPackage{textcomp}
818 [2020/02/02 v2.0n Standard LaTeX package]

```

A precaution in case this is used without rebuilding the format.

```

819 \NeedsTeXFormat{LaTeX2e}[2020/02/02]

```

This is implemented by defining the default subset:

```

820 \DeclareOption{full}{\DeclareEncodingSubset{TS1}{?}{0}}
821 \DeclareOption{almostfull}{\DeclareEncodingSubset{TS1}{?}{1}}
822 \DeclareOption{euro}{\DeclareEncodingSubset{TS1}{?}{8}}
823 \DeclareOption{safe}{\DeclareEncodingSubset{TS1}{?}{9}}

```

The default is set up in the kernel is “safe” these days for unknown fonts but LaTeX has definitions for most families so it seldom applies.

If a different default is used then one needs to check the results to ensure that there aren’t “missing glyphs”.

The next set of options define the warning level (default in the kernel is info only). Using the package options you can change this behavior.

```

824 \DeclareOption{error}
825     {\gdef\tc@errorwarn{\PackageError{textcomp}}}
826 \DeclareOption{warn}
827     {\gdef\tc@errorwarn#1#2{\PackageWarning{textcomp}{#1}}}
828 \DeclareOption{info}
829     {\gdef\tc@errorwarn#1#2{\PackageInfo{textcomp}{#1}}}
830 \DeclareOption{quiet}{\gdef\tc@errorwarn#1#2{}}

```

The “force” option basically changes the sub-encoding to that of the default (which, unless changes, is 9 these days), i.e., it no longer depends on the font in use. This is

mainly there because it might have been used in older documents, but not something that is recommended.

```

831 \DeclareOption{force}{%
832   \def\CheckEncodingSubset#1#2#3#4#5{%
833     \ifnum #4>%
834       0\csname #2:?\endcsname
835       \relax
836     \expandafter\@firstoftwo
837   \else
838     \expandafter\@secondoftwo
839   \fi
840   {#1{#2}}{#3}%
841   #5}%
842 }
843 \ExecuteOptions{info}
844 \ProcessOptions\relax

```

There is not much else to do nowadays, because everything is already set up in the L^AT_EX kernel.

```

845 \InputIfFileExists{textcomp.cfg}
846 {\PackageInfo{textcomp}{Local configuration file used}}{}
847 </TS1sty>

```

5.1 The old textcomp package code

This section contains the old code for the textcomp package and its documentation. It is only used if we roll back prior to 2020. Thus all the rest is mainly for historians. Note that the old code categorized in the sub-encodings only into 6 classes not 10.

```

848 <*TS1oldsty>
849 \ProvidesPackage{textcomp}
850 [2018/08/11 v2.0j Standard LaTeX package]

```

This one is for the TS1 encoding which contains text symbols for use with the T1-encoded text fonts. It therefore first inputs the file `TS1enc.def` and then sets (or resets) the defaults for the symbols it contains. The result of this is that when one of these symbols is accessed and the current encoding does not provide it, the symbol will be supplied by a silent, local change to this encoding.

Since many PostScript fonts only implement a subset of TS1 many commands only produce black blobs of ink. To resolve the resulting problems a number of options have been introduced and some code has been developed to distinguish sub-encodings.

The sub-encodings have a numerical id and are defined as follows for TS1:

#5 those TS1 symbols that are also in the ISO-Adobe character set; without `textcurrency`, which is often misused for the Euro. Older Type1 fonts from the non-T_EX world provide only this subset.

#4 = **#5** + `\texteuro`. Most newer fonts provide this.

#3 = **#4** + `\textomega`. Can also be described as $TS1 \cap (ISO-Adobe \cup MacRoman)$. (Except for the missing "currency".)

#2 = **#3** + `\textestimated` + `\textcurrency`. Can also be described as $TS1 \cap Adobe-Western-2$. This may be relevant for OpenType fonts, which usually show the Adobe-Western-2 character set.

#1 = TS1 without `\textcircled` and `\t`. These two glyphs are often not implemented and if their kernel defaults are changed commands like `\copyright` unnecessarily fail.

#0 = full TS1

And here a summary to go in the transcript file:

```

851 \PackageInfo{textcomp}{Sub-encoding information:\MessageBreak
852   \space\space 5 = only ISO-Adobe without
853   \string\textcurrency\MessageBreak
854   \space\space 4 = 5 + \string\texteuro\MessageBreak
855   \space\space 3 = 4 + \string\textohm\MessageBreak
856   \space\space 2 = 3 + \noexpand\textestimated+
857   \string\textcurrency\MessageBreak
858   \space\space 1 = TS1 - \noexpand\textcircled-
859   \string\t\MessageBreak
860   \space\space 0 = TS1 (full)\MessageBreak
861   Font families with sub-encoding setting implement\MessageBreak
862   only a restricted character set as indicated.\MessageBreak
863   Family '?' is the default used for unknown fonts.\MessageBreak
864   See the documentation for details\@gobble}

```

\DeclareEncodingSubset An encoding subset to which a font family belongs is declared by the command `\DeclareEncodingSubset` that takes the major encoding as the first argument (e.g., TS1), the family name as the second argument (e.g., `cmr`), and the subset encoding id as a third, (e.g., 0 for `cmr`).

The default encoding subset to use when nothing is known about the current font family is named `?`.

```

865 \def\DeclareEncodingSubset#1#2#3{%
866   \@ifundefined{#1:#2}%
867   {\PackageInfo{textcomp}{Setting #2 sub-encoding to #1/#3}}%
868   {\PackageInfo{textcomp}{Changing #2 sub-encoding to #1/#3}}%
869   \@namedef{#1:#2}{#3}}
870 \@onlypreamble\DeclareEncodingSubset

```

(End definition for `\DeclareEncodingSubset`.)

The options for the package are the following:

safe for unknown font families enables only symbols that are also in the ISO-Adobe character set; without "currency", which is often misused for the Euro. Older Type1 fonts from the non-TeX world provide only this subset.

euro enables the “safe” symbols plus the `\texteuro` command. Most newer fonts provide this.

full enables all TS1 commands; useful only with fonts like EC or CM bright.

almostfull same as “full”, except that `\textcircled` and `\t` are *not* redefined from their defaults to avoid that commands like `\copyright` suddenly no longer work.

force ignore all subset encoding definitions stored in the package itself or in the configuration file and always use the default subset as specified by one of the other options (seldom useful, only dangerous).

`\iftc@forced` Switch used to implement the force option

```
871 \newif\iftc@forced \tc@forcedfalse
```

(End definition for `\iftc@forced`.)

This is implemented by defining the default subset:

```
872 \DeclareOption{full}{\DeclareEncodingSubset{TS1}{?}{0}}
873 \DeclareOption{almostfull}{\DeclareEncodingSubset{TS1}{?}{1}}
874 \DeclareOption{euro}{\DeclareEncodingSubset{TS1}{?}{4}}
875 \DeclareOption{safe}{\DeclareEncodingSubset{TS1}{?}{5}}
```

The default is “almostfull” which means that old documents will work except that `\textcircled` and `\t` will use the kernel defaults (with the advantage that this also works if the current font (as often the case) doesn’t implement these glyphs.

The “force” option simply sets the switch to true.

```
876 \DeclareOption{force}{\tc@forcedtrue}
```

The suggestions to user is to use the “safe” option always unless that balks in which case they could switch to “almostfull” but then better check their output manually.

```
877 \def\tc@errorwarn{\PackageError}
878 \DeclareOption{warn}{\gdef\tc@errorwarn#1#2#3{\PackageWarning{#1}{#2}}}
879 \DeclareOption{quiet}{\gdef\tc@errorwarn#1#2#3{}}
880 \ExecuteOptions{almostfull}
881 \ProcessOptions\relax
```

`\CheckEncodingSubset` The command `\CheckEncodingSubset` will check if the current font family has the right encoding subset to typeset a certain command. It takes five arguments as follows: first argument is either `\UseTextSymbol`, `\UseTextAccent` depending on whether or not the symbol is a text symbol or a text accent.

The second argument is the encoding from which this symbol should be fetched.

The third argument is either a fake accessor command or an error message. the code in that argument (if ever executed) receives two arguments: `#2` and `#5` of `\CheckEncodingSubset`.

Argument four is the subset encoding id to test against: if this value is higher than the subset id of the current font family then we typeset the symbol, i.e., execute `#1{#2}#5` otherwise it runs `#3#5`, e.g., to produce an error message or fake the glyph somehow.

Argument five is the symbol or accent command that is being checked.

For usage examples see definitions below.

```
882 \iftc@forced
```

If the “force” option was given we always use the default for testing against.

```
883 \def\CheckEncodingSubset#1#2#3#4#5{%
884   \ifnum #4>%
885     0\csname #2:\endcsname
886     \relax
887   \expandafter\@firstoftwo
888   \else
889   \expandafter\@secondoftwo
890   \fi
891   {#1{#2}}{#3}%
892   #5%
893 }
```

In normal circumstances the test is a bit more complicated: first check if there exists a macro `\langle arg2\rangle:\langle current-family\rangle` and if so use that value to test against, otherwise use the default to test against.

```

894 \else
895 \def\CheckEncodingSubset#1#2#3#4#5{%
896   \ifnum #4>%
897     \expandafter\ifx\csname #2:\f@family\endcsname\relax
898       0\csname #2:?\endcsname
899     \else
900       \csname #2:\f@family\endcsname
901     \fi
902   \relax
903   \expandafter\@firstoftwo
904 \else
905   \expandafter\@secondoftwo
906 \fi
907   {#1{#2}}{#3}%
908   #5%
909 }
910 \fi

```

(End definition for `\CheckEncodingSubset`.)

`\tc@subst`

```

911 \def\tc@subst#1{%
912   \tc@errorwarn{textcomp}%
913   {Symbol \string#1 not provided by\MessageBreak
914     font family \f@family\space
915     in TS1 encoding.\MessageBreak Default family used instead}\@eha
916   \bgroup\fontfamily\textcompstubdefault\selectfont#1\egroup
917 }

```

(End definition for `\tc@subst`.)

`\tc@error` `\tc@error` is going to be used in arg #3 of `\CheckEncodingSubset` when a symbol is not available in a certain font family. It gets pass the encoding it normally lives in (arg one) and the name of the symbol or accent that has a problem.

```

918 % error commands take argument:
919 % #1 symbol to be used
920 \def\tc@error#1{%
921   \PackageError{textcomp}% % should be latex error if general
922   {Accent \string#1 not provided by\MessageBreak
923     font family \f@family\space
924     in TS1 encoding}\@eha
925 }

```

(End definition for `\tc@error`.)

`\tc@fake@euro` `\tc@fake@euro` is an example of a “fake” definition to use in arg #3 of `\CheckEncodingSubset` when a symbol is not available in a certain font family. Here we produce an Euro symbol by combining a “C” with a “=”.

```

926 \def\tc@fake@euro#1{%
927   \leavevmode
928   \PackageInfo{textcomp}{Faking \noexpand#1for font family

```

```

929                                     \f@family\MessageBreak in TS1 encoding}%
930 \valign{##\cr
931     \vfil\hbox to 0.07em{\dimen@f@size\p@
932                                     \math@fontsfalse
933                                     \fontsize{.7\dimen@}\z@\selectfont=\hss}%
934     \vfil\cr%
935     \hbox{C}\crrcr
936 }%
937 }

```

(End definition for `\tc@fake@euro`.)

`\tc@check@symbol` These are two abbreviations that we use below to check symbols and accents in TS1.
`\tc@check@accent` Only there to save some space, e.g., we can then write

```
DeclareTextCommandDefault{\textcurrency}{\tc@check@symbol3\textcurrency}
```

to ensure that `\textcurrency` is only typeset if the current font has a TS1 subset id of less than 3. Otherwise `\tc@error` is called telling the user that for this font family `\textcurrency` is not available.

```

938 \def\tc@check@symbol{\CheckEncodingSubset\UseTextSymbol{TS1}\tc@subst}
939 \def\tc@check@accent{\CheckEncodingSubset\UseTextAccent{TS1}\tc@error}

```

(End definition for `\tc@check@symbol` and `\tc@check@accent`.)

We start with the commands that are “safe” and which can be unconditionally set up, first the accents...

```

940 \DeclareTextAccentDefault{\capitalcedilla}{TS1}
941 \DeclareTextAccentDefault{\capitalogonek}{TS1}
942 \DeclareTextAccentDefault{\capitalgrave}{TS1}
943 \DeclareTextAccentDefault{\capitalacute}{TS1}
944 \DeclareTextAccentDefault{\capitalcircumflex}{TS1}
945 \DeclareTextAccentDefault{\capitaltilde}{TS1}
946 \DeclareTextAccentDefault{\capitaldieresis}{TS1}
947 \DeclareTextAccentDefault{\capitalhungarumlaut}{TS1}
948 \DeclareTextAccentDefault{\capitalring}{TS1}
949 \DeclareTextAccentDefault{\capitalcaron}{TS1}
950 \DeclareTextAccentDefault{\capitalbreve}{TS1}
951 \DeclareTextAccentDefault{\capitalmacron}{TS1}
952 \DeclareTextAccentDefault{\capitaldotaccent}{TS1}

```

...and then the other glyphs.

```

953 \DeclareTextSymbolDefault{\textcapitalcompwordmark}{TS1}
954 \DeclareTextSymbolDefault{\textascendercompwordmark}{TS1}
955 \DeclareTextSymbolDefault{\textquotestraightbase}{TS1}
956 \DeclareTextSymbolDefault{\textquotestraightdblbase}{TS1}
957 \DeclareTextSymbolDefault{\texttwelveudash}{TS1}
958 \DeclareTextSymbolDefault{\textthreequartersemdash}{TS1}
959 \DeclareTextSymbolDefault{\textdollar}{TS1}
960 \DeclareTextSymbolDefault{\textquotesingle}{TS1}
961 \DeclareTextSymbolDefault{\textasteriskcentered}{TS1}
962 \DeclareTextSymbolDefault{\textfractionsolidus}{TS1}
963 \DeclareTextSymbolDefault{\textminus}{TS1}
964 \DeclareTextSymbolDefault{\textlbrackdbl}{TS1}
965 \DeclareTextSymbolDefault{\textrbrackdbl}{TS1}
966 \DeclareTextSymbolDefault{\textasciigrave}{TS1}

```

```

967 \DeclareTextSymbolDefault{\texttildelow}{TS1}
968 \DeclareTextSymbolDefault{\textasciibreve}{TS1}
969 \DeclareTextSymbolDefault{\textasciicaron}{TS1}
970 \DeclareTextSymbolDefault{\textgravedbl}{TS1}
971 \DeclareTextSymbolDefault{\textacutedbl}{TS1}
972 \DeclareTextSymbolDefault{\textdagger}{TS1}
973 \DeclareTextSymbolDefault{\textdaggerdbl}{TS1}
974 \DeclareTextSymbolDefault{\textbardbl}{TS1}
975 \DeclareTextSymbolDefault{\textperthousand}{TS1}
976 \DeclareTextSymbolDefault{\textbullet}{TS1}
977 \DeclareTextSymbolDefault{\textcelsius}{TS1}
978 \DeclareTextSymbolDefault{\textflorin}{TS1}
979 \DeclareTextSymbolDefault{\texttrademark}{TS1}
980 \DeclareTextSymbolDefault{\textcent}{TS1}
981 \DeclareTextSymbolDefault{\textsterling}{TS1}
982 \DeclareTextSymbolDefault{\textyen}{TS1}
983 \DeclareTextSymbolDefault{\textbrokenbar}{TS1}
984 \DeclareTextSymbolDefault{\textsection}{TS1}
985 \DeclareTextSymbolDefault{\textasciidieresis}{TS1}
986 \DeclareTextSymbolDefault{\textcopyright}{TS1}
987 \DeclareTextSymbolDefault{\textordfeminine}{TS1}
988 \DeclareTextSymbolDefault{\textlnot}{TS1}
989 \DeclareTextSymbolDefault{\textregistered}{TS1}
990 \DeclareTextSymbolDefault{\textasciimacron}{TS1}
991 \DeclareTextSymbolDefault{\textdegree}{TS1}
992 \DeclareTextSymbolDefault{\textpm}{TS1}
993 \DeclareTextSymbolDefault{\texttwosuperior}{TS1}
994 \DeclareTextSymbolDefault{\textthreesuperior}{TS1}
995 \DeclareTextSymbolDefault{\textasciicute}{TS1}
996 \DeclareTextSymbolDefault{\textmu}{TS1}
997 \DeclareTextSymbolDefault{\textparagraph}{TS1}
998 \DeclareTextSymbolDefault{\textperiodcentered}{TS1}
999 \DeclareTextSymbolDefault{\textonesuperior}{TS1}
1000 \DeclareTextSymbolDefault{\textordmasculine}{TS1}
1001 \DeclareTextSymbolDefault{\textonequarter}{TS1}
1002 \DeclareTextSymbolDefault{\textonehalf}{TS1}
1003 \DeclareTextSymbolDefault{\textthreequarters}{TS1}
1004 \DeclareTextSymbolDefault{\texttimes}{TS1}
1005 \DeclareTextSymbolDefault{\textdiv}{TS1}

```

The `\texteuro` is only available for subsets with id 4 or less. Otherwise we fake the glyph using `\tc@fake@euro`

```

1006 \DeclareTextCommandDefault{\texteuro}
1007 {\CheckEncodingSubset\UseTextSymbol{TS1}\tc@fake@euro5\texteuro}

```

The `\textohm` is only available for subsets with id 3 or less. Otherwise we produce an error.

```

1008 \DeclareTextCommandDefault{\textohm}{\tc@check@symbol4\textohm}

```

The `\textestimated` and `\textcurrency` are only provided for fonts with subset encoding with id 2 or less.

```

1009 \DeclareTextCommandDefault{\textestimated}%
1010 {\tc@check@symbol3\textestimated}
1011 \DeclareTextCommandDefault{\textcurrency}%
1012 {\tc@check@symbol3\textcurrency}

```

Nearly all of the remaining glyphs are provided only with fonts with id 1 or 0, i.e., are essentially complete.

```

1013 \DeclareTextCommandDefault{\capitaltie}%
1014     {\tc@check@accent2\capitaltie}
1015 \DeclareTextCommandDefault{\newtie}%
1016     {\tc@check@accent2\newtie}
1017 \DeclareTextCommandDefault{\capitalnewtie}%
1018     {\tc@check@accent2\capitalnewtie}
1019 \DeclareTextCommandDefault{\textleftarrow}%
1020     {\tc@check@symbol2\textleftarrow}
1021 \DeclareTextCommandDefault{\textrightarrow}%
1022     {\tc@check@symbol2\textrightarrow}
1023 \DeclareTextCommandDefault{\textblank}%
1024     {\tc@check@symbol2\textblank}
1025 \DeclareTextCommandDefault{\textdblhyphen}%
1026     {\tc@check@symbol2\textdblhyphen}
1027 \DeclareTextCommandDefault{\textzerooldstyle}%
1028     {\tc@check@symbol2\textzerooldstyle}
1029 \DeclareTextCommandDefault{\textoneoldstyle}%
1030     {\tc@check@symbol2\textoneoldstyle}
1031 \DeclareTextCommandDefault{\texttwooldstyle}%
1032     {\tc@check@symbol2\texttwooldstyle}
1033 \DeclareTextCommandDefault{\textthreeoldstyle}%
1034     {\tc@check@symbol2\textthreeoldstyle}
1035 \DeclareTextCommandDefault{\textfouroldstyle}%
1036     {\tc@check@symbol2\textfouroldstyle}
1037 \DeclareTextCommandDefault{\textfiveoldstyle}%
1038     {\tc@check@symbol2\textfiveoldstyle}
1039 \DeclareTextCommandDefault{\textsixoldstyle}%
1040     {\tc@check@symbol2\textsixoldstyle}
1041 \DeclareTextCommandDefault{\textsevenoldstyle}%
1042     {\tc@check@symbol2\textsevenoldstyle}
1043 \DeclareTextCommandDefault{\texteightoldstyle}%
1044     {\tc@check@symbol2\texteightoldstyle}
1045 \DeclareTextCommandDefault{\textnineoldstyle}%
1046     {\tc@check@symbol2\textnineoldstyle}
1047 \DeclareTextCommandDefault{\textlangle}%
1048     {\tc@check@symbol2\textlangle}
1049 \DeclareTextCommandDefault{\textrangle}%
1050     {\tc@check@symbol2\textrangle}
1051 \DeclareTextCommandDefault{\textmho}%
1052     {\tc@check@symbol2\textmho}
1053 \DeclareTextCommandDefault{\textbigcircle}%
1054     {\tc@check@symbol2\textbigcircle}
1055 \DeclareTextCommandDefault{\textuparrow}%
1056     {\tc@check@symbol2\textuparrow}
1057 \DeclareTextCommandDefault{\textdownarrow}%
1058     {\tc@check@symbol2\textdownarrow}
1059 \DeclareTextCommandDefault{\textborn}%
1060     {\tc@check@symbol2\textborn}
1061 \DeclareTextCommandDefault{\textdivorced}%
1062     {\tc@check@symbol2\textdivorced}
1063 \DeclareTextCommandDefault{\textdied}%
1064     {\tc@check@symbol2\textdied}

```

```

1065 \DeclareTextCommandDefault{\textleaf}%
1066     {\tc@check@symbol2\textleaf}
1067 \DeclareTextCommandDefault{\textmarried}%
1068     {\tc@check@symbol2\textmarried}
1069 \DeclareTextCommandDefault{\textmusicalnote}%
1070     {\tc@check@symbol2\textmusicalnote}
1071 \DeclareTextCommandDefault{\textdblhyphenchar}%
1072     {\tc@check@symbol2\textdblhyphenchar}
1073 \DeclareTextCommandDefault{\textdollaroldstyle}%
1074     {\tc@check@symbol2\textdollaroldstyle}
1075 \DeclareTextCommandDefault{\textcentoldstyle}%
1076     {\tc@check@symbol2\textcentoldstyle}
1077 \DeclareTextCommandDefault{\textcolonmonetary}%
1078     {\tc@check@symbol2\textcolonmonetary}
1079 \DeclareTextCommandDefault{\textwon}%
1080     {\tc@check@symbol2\textwon}
1081 \DeclareTextCommandDefault{\textnaira}%
1082     {\tc@check@symbol2\textnaira}
1083 \DeclareTextCommandDefault{\textguarani}%
1084     {\tc@check@symbol2\textguarani}
1085 \DeclareTextCommandDefault{\textpeso}%
1086     {\tc@check@symbol2\textpeso}
1087 \DeclareTextCommandDefault{\textlira}%
1088     {\tc@check@symbol2\textlira}
1089 \DeclareTextCommandDefault{\textrecipe}%
1090     {\tc@check@symbol2\textrecipe}
1091 \DeclareTextCommandDefault{\textinterrobang}%
1092     {\tc@check@symbol2\textinterrobang}
1093 \DeclareTextCommandDefault{\textinterrobangdown}%
1094     {\tc@check@symbol2\textinterrobangdown}
1095 \DeclareTextCommandDefault{\textdong}%
1096     {\tc@check@symbol2\textdong}
1097 \DeclareTextCommandDefault{\textpertenthousand}%
1098     {\tc@check@symbol2\textpertenthousand}
1099 \DeclareTextCommandDefault{\textpilcrow}%
1100     {\tc@check@symbol2\textpilcrow}
1101 \DeclareTextCommandDefault{\textbaht}%
1102     {\tc@check@symbol2\textbaht}
1103 \DeclareTextCommandDefault{\textnumero}%
1104     {\tc@check@symbol2\textnumero}
1105 \DeclareTextCommandDefault{\textdiscount}%
1106     {\tc@check@symbol2\textdiscount}
1107 \DeclareTextCommandDefault{\textopenbullet}%
1108     {\tc@check@symbol2\textopenbullet}
1109 \DeclareTextCommandDefault{\textservicemark}%
1110     {\tc@check@symbol2\textservicemark}
1111 \DeclareTextCommandDefault{\textlquill}%
1112     {\tc@check@symbol2\textlquill}
1113 \DeclareTextCommandDefault{\textrquill}%
1114     {\tc@check@symbol2\textrquill}
1115 \DeclareTextCommandDefault{\textcopyleft}%
1116     {\tc@check@symbol2\textcopyleft}
1117 \DeclareTextCommandDefault{\textcircledP}%
1118     {\tc@check@symbol2\textcircledP}

```

```

1119 \DeclareTextCommandDefault{\textreferencemark}%
1120   {\tc@check@symbol2\textreferencemark}
1121 \DeclareTextCommandDefault{\textsurd}%
1122   {\tc@check@symbol2\textsurd}

```

The `\textcircled` and `\t` are handled specially, unless the current font has a subset id of 0 (i.e. full TS1) we pick the symbols up from the math font encodings, i.e., the third argument to `\CheckEncodingSubset` uses `\UseTextAccent` to get them from there.

```

1123 \DeclareTextCommandDefault{\textcircled}
1124   {\CheckEncodingSubset\UseTextAccent{TS1}%
1125    {\UseTextAccent{OMS}}1\textcircled}
1126 \DeclareTextCommandDefault{\t}
1127   {\CheckEncodingSubset\UseTextAccent{TS1}%
1128    {\UseTextAccent{OML}}1\t}

```

Finally input the encoding-specific definitions for TS1 thus making the top-level definitions optimized for this encoding (and not for the default encoding).

```

1129 \input{ts1enc.def}

```

Now having the new glyphs available we also want to make sure that they are used. For most cases this will automatically happen but for some glyphs there are inferior definitions already known to L^AT_EX which will prevent the usage of the TS1 versions. So we better get rid of them:

```

1130 \UndeclareTextCommand{\textsterling}{OT1}
1131 \UndeclareTextCommand{\textdollar} {OT1}

```

Similar declarations should probably be made for other encodings like OT4 if they are in use.

```

1132 %\UndeclareTextCommand{\textsterling}{OT4}
1133 %\UndeclareTextCommand{\textdollar} {OT4}

```

From the T1 encoding there are two candidates for removal: `%0` and `%00` since these are both constructed from `%` followed by a tiny ‘`o`’ rather than being a single glyph. The problem with this approach is that in PostScript fonts this small zero is usually not available resulting in `%■` rather than `%0` while the real glyph (at least for `\textperthousand`) is available in the PostScript version of TS1. So for the moment we compromise by removing the T1 declaration for `\textperthousand` but keeping the one for `\textpertenthousand`. This will have the effect that with Computer Modern fonts everything will come out (although `%0` and `%00` are not taken from the same physical font) and with PostScript fonts `%0` will come out correctly while `%00` will most likely look like `%■` — which is probably an improvement over just getting a single ‘`■`’ to indicate a completely missing glyph, which would happen if we also ‘undeclared’ `\textpertenthousand`.

```

1134 \UndeclareTextCommand{\textperthousand}{T1}
1135 %\UndeclareTextCommand{\textpertenthousand}{T1}

```

5.1.1 Supporting oldstyle digits

```

1136 \DeclareRobustCommand\oldstylenums[1]{%
1137   \begingroup
1138   \ifmmode
1139     \mathgroup\symletters #1%
1140   \else
1141     \CheckEncodingSubset\@use@text@encoding{TS1}%
1142     {\PackageWarning{textcomp}%

```

```

1143         {Oldstyle digits unavailable for
1144         family \f@family.\MessageBreak
1145         Lining digits used instead}}}%
1146     \tw@{#1}%
1147 \fi
1148 \endgroup
1149 }

```

5.1.2 Subset encoding defaults

For many font families commonly used in the T_EX world we provide the subset encoding data here. Users can add additional font families in the file `textcomp.cfg` if they own other fonts.

However, if the option “forced” was given then all subset encoding specifications are ignored, so there is no point in setting any of them up:

```

1150 \iftc@forced \else

```

Computer modern based fonts (e.g., CM, CM-Bright, Concrete):

```

1151 \DeclareEncodingSubset{TS1}{cmr}      {0}
1152 \DeclareEncodingSubset{TS1}{cmss}     {0}
1153 \DeclareEncodingSubset{TS1}{cmtt}     {0}
1154 \DeclareEncodingSubset{TS1}{cmvtt}    {0}
1155 \DeclareEncodingSubset{TS1}{cmbr}     {0}
1156 \DeclareEncodingSubset{TS1}{cmtl}     {0}
1157 \DeclareEncodingSubset{TS1}{ccr}      {0}

```

PSNFSS fonts:

```

1158 \DeclareEncodingSubset{TS1}{ptm}      {4}
1159 \DeclareEncodingSubset{TS1}{pcr}      {4}
1160 \DeclareEncodingSubset{TS1}{phv}      {4}
1161 \DeclareEncodingSubset{TS1}{ppl}      {3}
1162 \DeclareEncodingSubset{TS1}{pag}      {4}
1163 \DeclareEncodingSubset{TS1}{pbk}      {4}
1164 \DeclareEncodingSubset{TS1}{pnc}      {4}
1165 \DeclareEncodingSubset{TS1}{pzc}      {4}
1166 \DeclareEncodingSubset{TS1}{bch}      {4}
1167 \DeclareEncodingSubset{TS1}{put}      {5}

```

Other CTAN fonts (probably not complete):

```

1168 \DeclareEncodingSubset{TS1}{uag}      {5}
1169 \DeclareEncodingSubset{TS1}{ugq}      {5}
1170 \DeclareEncodingSubset{TS1}{ul8}      {4}
1171 \DeclareEncodingSubset{TS1}{ul9}      {4}    % (LuxiSans, one day)
1172 \DeclareEncodingSubset{TS1}{augie}    {5}
1173 \DeclareEncodingSubset{TS1}{dayrom}    {3}
1174 \DeclareEncodingSubset{TS1}{dayroms}  {3}
1175 \DeclareEncodingSubset{TS1}{pxr}      {0}
1176 \DeclareEncodingSubset{TS1}{pxss}     {0}
1177 \DeclareEncodingSubset{TS1}{pxtt}     {0}
1178 \DeclareEncodingSubset{TS1}{txr}      {0}
1179 \DeclareEncodingSubset{TS1}{txss}     {0}
1180 \DeclareEncodingSubset{TS1}{txtt}     {0}

```

Latin Modern and TeX Gyre:

```

1181 \DeclareEncodingSubset{TS1}{lmr}      {0}
1182 \DeclareEncodingSubset{TS1}{lmdh}     {0}

```



```

1183 \DeclareEncodingSubset{TS1}{lmss}    {0}
1184 \DeclareEncodingSubset{TS1}{lmssq}   {0}
1185 \DeclareEncodingSubset{TS1}{lmvtt}   {0}
1186 \DeclareEncodingSubset{TS1}{lmtt}    {0}

1187 \DeclareEncodingSubset{TS1}{qhv}     {0}
1188 \DeclareEncodingSubset{TS1}{qag}     {0}
1189 \DeclareEncodingSubset{TS1}{qbk}     {0}
1190 \DeclareEncodingSubset{TS1}{qcr}     {0}
1191 \DeclareEncodingSubset{TS1}{qcs}     {0}
1192 \DeclareEncodingSubset{TS1}{qpl}     {0}
1193 \DeclareEncodingSubset{TS1}{qtm}     {0}
1194 \DeclareEncodingSubset{TS1}{qzc}     {0}
1195 \DeclareEncodingSubset{TS1}{qhvc}    {0}

```

Fourier-GUTenberg:

```

1196 \DeclareEncodingSubset{TS1}{futs}    {4}
1197 \DeclareEncodingSubset{TS1}{futex}   {4}
1198 \DeclareEncodingSubset{TS1}{futj}    {4}

```

Y&Y's Lucida Bright

```

1199 \DeclareEncodingSubset{TS1}{hlh}     {3}
1200 \DeclareEncodingSubset{TS1}{hls}     {3}
1201 \DeclareEncodingSubset{TS1}{hlst}    {3}

```

The remaining settings for Lucida are conservative: the following fonts contain the `\textohm` character but not the `\texteuro`, i.e., belong to neither subset 4 nor subset 3. If you want to use the `\textohm` with these fonts copy these definition to `textcomp.cfg` and change the subset to 3. However in that case make sure that you do not use the `\texteuro`.

```

1202 \DeclareEncodingSubset{TS1}{hlct}    {5}
1203 \DeclareEncodingSubset{TS1}{hlx}     {5}
1204 \DeclareEncodingSubset{TS1}{hlce}    {5}
1205 \DeclareEncodingSubset{TS1}{hlcn}    {5}
1206 \DeclareEncodingSubset{TS1}{hlcw}    {5}
1207 \DeclareEncodingSubset{TS1}{hlcf}    {5}

```

Other commercial families...

```

1208 \DeclareEncodingSubset{TS1}{pplx}    {3}
1209 \DeclareEncodingSubset{TS1}{pplj}    {3}
1210 \DeclareEncodingSubset{TS1}{ptmx}    {4}
1211 \DeclareEncodingSubset{TS1}{ptmj}    {4}

```

If the file `textcomp.cfg` exists it will be loaded at this point. This allows to define further subset encodings for font families not covered by default.

```

1212 \InputIfFileExists{textcomp.cfg}
1213   {\PackageInfo{textcomp}{Local configuration file used}}{}
1214 \fi
1215 </TS1oldsty>

```

File E

ltpageno.dtx

1 Page Numbering

Page numbers are produced by a page counter, used just like any other counter. The only difference is that `\c@page` contains the number of the next page to be output (the one currently being produced), rather than one minus it. Thus, it is normally initialized to 1 rather than 0. `\c@page` is defined to be `\count0`, rather than a count assigned by `\newcount`.

`\pagenumbering` The user sets the page number style with the `\pagenumbering{<foo>}` command, which sets the page counter to 1 and defines `\thepage` to be `\foo`. For example, `\pagenumbering{roman}` causes pages to be numbered i, ii, etc.

```
1 <*2kernel>
2 \message{page nos.,}

3 \countdef\c@page=0 \c@page=1
4 \def\c1@page{}
5 \def\pagenumbering#1{%
6   \global\c@page \c@one \gdef\thepage{\csname @#1\endcsname
7     \c@page}}
8 </2kernel>
```

File F

ltxref.dtx

1 Cross Referencing

The user writes `\label{foo}` to define the following cross-references:

`\ref{foo}`: value of most recently incremented referenceable counter. in the current environment. (Chapter, section, theorem and enumeration counters are referenceable, footnote counters are not.)

`\pageref{foo}`: page number at which `\label{foo}` command appeared. where foo can be any string of characters not containing `\`, `{` or `}`.

Note: The scope of the `\label` command is delimited by environments, so `\begin{theorem} \label{foo} ... \end{theorem} \label{bar}` defines `\ref{foo}` to be the theorem number and `\ref{bar}` to be the current section number.

Note: `\label` does the right thing in terms of spacing – i.e., leaving a space on both sides of it is equivalent to leaving a space on either side.

1.1 Cross Referencing

Historical L^AT_EX 2.09 comments (not necessarily accurate any more):

```
1 \<*2ekernel)
2 \message{x-ref,}
```

This is implemented as follows. A referenceable counter CNT is incremented by the command `\refstepcounter{CNT}`, which sets `\@currentlabel == {CNT}{eval(\p@cnt\theCNT)}`. The command `\label{FOO}` then writes the following on file `\@auxout`:

```
\newlabel{FOO}{{eval(\@currentlabel)}{eval(\thepage)}}
```

```
\ref{FOO} ==
BEGIN
  if \r@foo undefined
  then  @refundefined := G T
        ??
        Warning: 'reference foo on page ... undefined'
  else  \@car \eval(\r@FOO)\@nil
  fi
END
```

```
\pageref{foo} =
BEGIN
  if \r@foo undefined
  then  @refundefined := G T
        ??
        Warning: 'reference foo on page ... undefined'
  else  \@cdr \eval(\r@FOO)\@nil
  fi
END
```

End of historical L^AT_EX 2.09 comments.

`\labelformat`

A reference via `\ref` produces by default the data associated with the corresponding `\label` command (typically a number); any additional formatting has to be provided by the user. If, for example, references to equations are always to be typeset as “equation (*number*)”, one has to code “`equation (\ref{key})`”. With `\labelformat` there is a possibility to generate such frills automatically without resorting to low-level coding. The command takes two arguments: the first is the name of a counter and the second is its representation when referenced. This means that for a successful usage, one has to know the counter name being used for generating the label, though in practice this should not pose a problem. The current counter number is picked up as an argument. Here are two examples:

```
\labelformat{section}{section-#1}
\labelformat{equation}{equation~( #1 )}
```

`\Ref`

A side effect of using `\labelformat` is that, depending on the defined formatting, it becomes impossible to use `\ref` at the beginning of a sentence (if its replacement text starts with a lowercase letter). To overcome this problem we introduce the command `\Ref` that behave like `\ref` except that it uppercases the first token of the generated string.

To make `\Ref` work properly the very first token in the second argument of `\labelformat` has to be a simple ASCII or UTF-8 letter, otherwise the capitalization will fail or worse, you will end up with some error messages. If you actually need something more complicated in this place (e.g., an accented letter not written as a UTF-8 character) you have to explicitly surround it with braces, to identify the part that needs to be capitalized. For example, for figure references in the Hungarian language you might want to write `\labelformat{figure}{\ 'a}bra~\thefigure}` or use `\labelformat{figure}{ábra~\thefigure}` which avoids the brace problem.

`\G@refundefinedtrue`
`\@refundefined`

This does not save on name-space (since `\G@refundefinedfalse` was never needed) but it does make the implementation of such one-way switches more consistent. The extra macro to make the change is used since this change appears several times.

Note despite its name, `\G@refundefinedtrue` does *not* correspond to an `\if` command, and there is no matching `...false`. It would be more natural to call the command `\G@refundefined` (as inspection of the change log will reveal) but unfortunately such a change would break any package that had defined a `\ref`-like command that mimicked the definition of `\ref`, calling `\G@refundefinedtrue`. Inspection of the T_EX archives revealed several such packages, and so this command has been named `...true` so that the definition of `\ref` need not be changed, and the packages will work without change.

```
3 % \newif\ifG@refundefined
4 % \def\G@refundefinedtrue{\global\let\ifG@refundefined\iftrue}
5 % \def\G@refundefinedfalse{\global\let\ifG@refundefined\iffalse}
6 \def\G@refundefinedtrue{%
7   \gdef\@refundefined{%
8     \@latex@warning@no@line{There were undefined references}}
9 \let\@refundefined\relax
```

(End definition for \G@refundefinedtrue and \@refundefined.)

`\ref` Referencing a `\label`. RmS 91/10/25: added a few extra `\reset@font`, as suggested by
`\pageref` Bernd Raichle
`\setref` RmS 92/08/14: made `\ref` and `\pageref` robust
RmS 93/09/08: Added setting of redefined switch.

```

10 \def\@setref#1#2#3{%
11   \ifx#1\relax
12     \protect\G@refundefinedtrue
13     \nfss@text{\reset@font\bfseries ??}%
14     \@latex@warning{Reference ‘#3’ on page \thepage \space
15       undefined}%
16   \else
17     \expandafter#2#1\null
18   \fi}
19 \def\ref#1{\expandafter\@setref\csname r@#1\endcsname\@firstoftwo{#1}}
20 \def\pageref#1{\expandafter\@setref\csname r@#1\endcsname
21   \@secondoftwo{#1}}

```

(End definition for `\ref`, `\pageref`, and `\@setref`.)

`\newlabel` This command will be written to the `.aux` file to pass label information from one run to another.

The internal form of `\newlabel` and `\bibcite`. Note that this macro does its work inside a group. That way the local assignments it needs to do don't clutter the save stack. This prevents large documents with many labels to run out of save stack.

```

22 \def\@newl@bel#1#2#3{%
23   \@ifundefined{#1@#2}%
24     \relax
25     {\gdef \@multiplelabels {%
26       \@latex@warning@no@line{There were multiply-defined labels}}%
27       \@latex@warning@no@line{Label ‘#2’ multiply defined}}%
28   \global\@namedef{#1@#2}{#3}}
29 \def\newlabel{\@newl@bel r}
30 \@onlypreamble\@newl@bel

```

(End definition for `\newlabel` and `\@newl@bel`.)

`\if@multiplelabels` This is redefined to produce a warning if at least one label is defined more than once. It
`\@multiplelabels` is executed by the `\enddocument` command.

```

31 \let \@multiplelabels \relax

```

(End definition for `\if@multiplelabels` and `\@multiplelabels`.)

`\label` The commands `\label` and `\refstepcounter` have been changed to allow `\protect`'ed commands to work properly. For example,

```

\def\thechapter{\protect\foo{\arabic{chapter}.\roman{section}}}

```

will cause a `\label{bar}` command to define `\ref{bar}` to expand to something like `\foo{4.d}`. Change made 20 Jul 88.

```

32 \def\label#1{\@bsphack
33   \protected@write\@auxout{%
34     {\string\newlabel{#1}{\@currentlabel}{\thepage}}}%
35   \@esphack}

```

(End definition for \label.)

```
36 </2ekernel>
37 <*2ekernel | latexrelease>
38 <latexrelease>\IncludeInRelease{2020/10/01}%
39 <latexrelease>          {\refstepcounter}{Add \@currentcounter}%
```

\refstepcounter Step the counter and allow for labels to point to its current value.

```
40 \def\@currentcounter{}
41 \def\refstepcounter#1{\stepcounter{#1}%
42   \edef\@currentcounter{#1}%
43   \protected@edef\@currentlabel
```

By generating the second csname first the `\p@...` command can grab it as an argument which can be helpful for more complicated typesetting arrangements.

The trick is to ensure that `\csname the#1\endcsname` is turned into a single token before `\p@...` is expanded further. This way, if the `\p@...` command is a macro with one argument it will receive `\the...`. With the original kernel code (i.e., without the `\expandafter`) it will instead pick up `\csname` which would be disastrous.

Using `\expandafter` instead of braces delimiting the argument is better because, assuming that the `\p@...` command is not defined as a macro with one argument, the braces will stay and prohibit kerning that might otherwise happen between the glyphs generated by `\the...` and surrounding glyphs.

```
44   {\csname p@#1\expandafter\endcsname\csname the#1\endcsname}%
45 }
```

(End definition for \refstepcounter.)

\labelformat A shortcut to set the `\p@...` macro for a counter. It will pick up the counter representation as an argument so that it can be specially formatted.

```
46 \def\labelformat#1{\expandafter\def\csname p@#1\endcsname##1}
```

(End definition for \labelformat.)

\Ref This macro expands the result of `\ref` and then uppercases the first token. Only useful if the label was generated via `\labelformat` and contains some lower case letter at its start. If the label starts with a complicated construct (e.g., an accented letter that is provided via a command, e.g., `\"a` instead of a UTF-8 character like `ä`) one has to surround everything that needs uppercasing in a brace group in the definition of `\labelformat`.³⁰

```
47 \DeclareRobustCommand\Ref[1]{\protected@edef\@tempa{\ref{#1}}%
48   \expandafter\MakeUppercase\@tempa}
```

(End definition for \Ref.)

```
49 </2ekernel | latexrelease>
50 <latexrelease>\EndIncludeInRelease
51 <latexrelease>\IncludeInRelease{2019/10/01}%
52 <latexrelease>          {\refstepcounter}{Add \labelformat and \Ref}%
53 <latexrelease>\let\@currentcounter\@undefined
54 <latexrelease>\def\refstepcounter#1{\stepcounter{#1}%
55 <latexrelease>   \protected@edef\@currentlabel
```

³⁰There is one problem with this approach: the braces are kept in a normal `\ref` which might spoil kerning. Perhaps one day this needs redoing.

```

56 <latexrelease>      {\csname p@#1\expandafter\endcsname\csname the#1\endcsname}%
57 <latexrelease>}
58 <latexrelease>\def\labelformat#1{\expandafter\def\csname p@#1\endcsname##1}
59 <latexrelease>\DeclareRobustCommand\Ref[1]{\protected@edef\@tempa{\ref{#1}}}%
60 <latexrelease>      \expandafter\MakeUppercase\@tempa}
61 <latexrelease>\EndIncludeInRelease
62 <latexrelease>\IncludeInRelease{0000/00/00}%
63 <latexrelease>      {\refstepcounter}{Add \labelformat and \Ref}%
64 <latexrelease>
65 <latexrelease>\def\refstepcounter#1{\stepcounter{#1}%
66 <latexrelease>      \protected@edef\@currentlabel
67 <latexrelease>      {\csname p@#1\endcsname\csname the#1\endcsname}%
68 <latexrelease>}
69 <latexrelease>\let\labelformat\@undefined
70 <latexrelease>\let\Ref\@undefined
71 <latexrelease>
72 <latexrelease>\EndIncludeInRelease
73 <*2ekernel>

```

`\@currentlabel` Default for `\label` commands that come before any environment.

```

74 \def\@currentlabel{}

```

(End definition for \@currentlabel.)

```

75 </2ekernel>

```

File G

ltmiscen.dtx

1 Miscellaneous Environments

This section implements the basic environment mechanism, and also a few specific environments including `document`, The math environments and related commands, the ‘flushing’ environments, (`center`, `flushleft`, `flushright`), and `verbatim`.

```
1 \*2ekernel)
2 \message{environments,}
```

1.1 Environments

`\begin{foo}` and `\end{foo}` are used to delimit environment `foo`.

`\begin{foo}` starts a group and calls `\foo` if it is defined, otherwise it does nothing.

`\end{foo}` checks to see that it matches the corresponding `\begin` and if so, it calls `\endfoo` and does an `\endgroup`. Otherwise, `\end{foo}` does nothing.

If `\end{foo}` needs to ignore blanks after it, then `\endfoo` should globally set the `@ignore` switch true with `\@ignoretrue` (this will automatically be global).

NOTE: `\@@end` is defined to be the `\end` command of $\mathrm{T}_{\mathrm{E}}\mathrm{X}82$.

`\enddocument` is the user’s command for ending the manuscript file.

`\stop` is a panic button — to end $\mathrm{T}_{\mathrm{E}}\mathrm{X}$ in the middle.

Historical $\mathrm{L}^{\mathrm{A}}\mathrm{T}_{\mathrm{E}}\mathrm{X} 2.09$ comments (not necessarily accurate any more):

```
\enddocument ==
BEGIN
  \@checkend{document}    %% checks for unmatched \begin
  \clearpage
  \begingroup
    if @filesw = true
    then close file @mainaux
    if G@refundefined = true
    then LaTeX Warning: 'There are undefined references.' fi
    if @multiplelabels = true
    then LaTeX Warning:
      'One or more label(s) multiply defined.'
    else
      \@setckpt {ARG1}{ARG2} == null
      \newlabel{LABEL}{VAL} ==
      BEGIN
        \reserved@a == VAL
        if def(\reserved@a) = def(\r@LABEL)
        else @tempswa := true          fi
      END
      \bibcite{LABEL}{VAL} == null
      BEGIN
        \reserved@a == VAL
        if def(\reserved@a) = def(\g@LABEL)
        else @tempswa := true          fi
```



```

                                END
                                @tempswa := false
                                make @ a letter
                                \input \jobname.AUX
                                if @tempswa = true
                                    then LaTeX Warning: 'Label may have changed.
                                                Rerun to get cross-references right.'
                                fi
                                fi
                                fi
                                \endgroup
                                finish up
                                END

                                \@writefile{EXT}{ENTRY} ==
                                    if tf@EXT undefined
                                        else \write\tf@EXT{ENTRY}
                                    fi
                                End of historical LATEX 2.09 comments.

\currentenvir The name of the current environment. Initialized to document to so that \end{document}
works correctly.
3 \def\currentenvir{document}

(End definition for \currentenvir.)

\if@ignore
\@ignoretrue 4 \def\ignorefalse{\global\let\if@ignore\iffalse}
\ignorefalse 5 \def\ignoretrue {\global\let\if@ignore\iftrue}
6 \ignorefalse

(End definition for \if@ignore, \@ignoretrue, and \ignorefalse.)

\ignorespacesafterend
7 \let\ignorespacesafterend\@ignoretrue

(End definition for \ignorespacesafterend.)

\enddocument
8 \</2kernel>
9 \<*2kernel | latexrelease>
10 \<latexrelease>\IncludeInRelease{2020/10/01}%
11 \<latexrelease> \enddocument}{Use Hooks}%
12 \def\enddocument{%

The \end{document} hook is executed first. If necessary it can contain a \clearpage to
output dangling floats first. In this position it can also contain something like \end{foo}
so that the whole document effectively starts and ends with some special environment.
However, this must be used with care, eg if two applications would use this without knowl-
edge of each other the order of the environments will be wrong after all. \AtEndDocument
is redefined at this point so that and such commands that get into the hook do not chase
their tail...

13 \@kernel@before@enddocument
14 \UseOneTimeHook{enddocument}%
15 \@kernel@after@enddocument

```

```

16 \checkend{document}%
17 \clearpage
18 \UseOneTimeHook{enddocument/afterlastpage}%
19 \@kernel@after@enddocument@afterlastpage
20 \begin{group}
21   \if@filesw
22     \immediate\closeout\@mainaux
23     \let\@setckpt\@gobbletwo
24     \let\@newl@bel\@testdef

```

The previous line is equiv to setting

```

\def\newlabel{\@testdef r}%
\def\bibcite{\@testdef b}%

```

We use `\@input` to load the `.aux` file, so that it doesn't show up in the list of files produced by `\listfiles`.

```

25 \tempwafalse
26 \makeatletter \@input\jobname.aux
27 \fi
28 \UseOneTimeHook{enddocument/afteraux}%

```

Next hook is expect to contain only code for writing info messages on the terminal.

```

29 \UseOneTimeHook{enddocument/info}%
30 \endgroup
31 \UseOneTimeHook{enddocument/end}%
32 \deadcycles\z@%

```

The public hooks used in `\enddocument`:

```

33 \NewHook{enddocument}
34 \NewHook{enddocument/afterlastpage}
35 \NewHook{enddocument/afteraux}
36 \NewHook{enddocument/info}
37 \NewHook{enddocument/end}

```

This is one of the few places where we already add data and rules to a hook already in the kernel.

```

38 \AddToHook{enddocument/info}[kernel/filelist]{\@dofilelist}
39 \AddToHook{enddocument/info}[kernel/warnings]{\@enddocument@kernel@warnings}
40 \DeclareHookRule{enddocument/info}{kernel/filelist}{before}{kernel/warnings}

```

(End definition for \enddocument.)

`\@enddocument@kernel@warnings`

```

41 \def\@enddocument@kernel@warnings{%

```

First we check for font size substitution bigger than `\fontsubfuzz`. The `\relax` is necessary because this is a macro not a register.

```

42 \ifdim \font@submax >\fontsubfuzz\relax

```

In case you wonder about the `\@gobbletwo` inside the message below, this is a horrible hack to remove the tokens `\on@line`. that are added by `\@font@warning` at the end.

```

43 \font@warning{Size substitutions with differences\MessageBreak
44             up to \font@submax\space have occurred.\@gobbletwo}%
45 \fi

```

The macro `\@defaultsubs` is initially `\relax` but gets redefined to produce a warning if there have been some default font substitutions.

```
46 \@@defaultsubs
```

The macro `\@refundefined` is initially `\relax` but gets redefined to produce a warning if there are undefined refs.

```
47 \@@refundefined
```

If a label is defined more than once, `\@tempswa` will always be true and thus produce a “Label(s) may ...” warning. But since a rerun will not solve that problem (unless one uses a package like `varioref` that generates labels on the fly), we suppress this message.

```
48 \if@filesw
49   \ifx \@multiplelabels \relax
50     \if@tempswa
51       \@latex@warning@no@line{Label(s) may have changed.
52         Rerun to get cross-references right}%
53     \fi
54   \else
55     \@multiplelabels
56   \fi
57   \ifx \@extra@page@added \relax
58     \@latex@warning@no@line{Temporary extra page added at the end.
59       Rerun to get it removed}%
60   \fi
```

We could think of adding a warning that nothing can be corrected while `\nofiles` is in force. In the past the warnings related to the aux file are simply suppressed in this case.

```
61 \fi
62 }
```

(End definition for `\@enddocument@kernel@warnings.`)

```
63 </2ekernel | latexrelease>
64 <latexrelease>\EndIncludeInRelease
65 <latexrelease>\IncludeInRelease{0000/00/00}%
66 <latexrelease>          {\enddocument}{Use Hooks}%
67 <latexrelease>
68 <latexrelease>\def\enddocument{%
69 <latexrelease>  \let\AtEndDocument\@firstofone
70 <latexrelease>  \@enddocumenthook
71 <latexrelease>  \@checkend{document}%
72 <latexrelease>  \clearpage
73 <latexrelease>  \begingroup
74 <latexrelease>    \if@filesw
75 <latexrelease>      \immediate\closeout\@mainaux
76 <latexrelease>      \let\@setckpt\@gobbletwo
77 <latexrelease>      \let\@newl@bel\@testdef
78 <latexrelease>      \@tempwafalse
79 <latexrelease>      \makeatletter \@@input\jobname.aux
80 <latexrelease>    \fi
81 <latexrelease>    \@dofilelist
82 <latexrelease>    \ifdim \font@submax >\fontsubfuzz\relax
83 <latexrelease>      \@font@warning{Size substitutions with differences\MessageBreak
84 <latexrelease>        up to \font@submax\space have occurred.\@gobbletwo}%
```

```

85 <latexrelease> \fi
86 <latexrelease> \@defaultsubs
87 <latexrelease> \@refundefined
88 <latexrelease> \if@filesw
89 <latexrelease> \ifx \@multiplelabels \relax
90 <latexrelease> \if@tempswa
91 <latexrelease> \@latex@warning@no@line{Label(s) may have changed.
92 <latexrelease> Rerun to get cross-references right}%
93 <latexrelease> \fi
94 <latexrelease> \else
95 <latexrelease> \@multiplelabels
96 <latexrelease> \fi
97 <latexrelease> \fi
98 <latexrelease> \endgroup
99 <latexrelease> \deadcycles\z@\@end}
100 <latexrelease>
101 <latexrelease>\let\@enddocument@kernel@warnings\@undefined
102 <latexrelease>
103 <latexrelease>\EndIncludeInRelease
104 <*2ekernel>

```

`\@kernel@before@enddocument` The `\@kernel@before@enddocument` hook is slightly different because we initialize it with `\par` so that `\enddocument` always returns to vertical mode as its first action.

```

105 </2ekernel>
106 <*2ekernel | latexrelease>
107 <latexrelease>\IncludeInRelease{2021/06/01}%
108 <latexrelease> \{\@kernel@before@enddocument\}{kernel before hook}%
109 <def>\@kernel@before@enddocument{\par}
110 </2ekernel | latexrelease>
111 <latexrelease>\EndIncludeInRelease

```

The rollback code renders it harmless.

```

112 <latexrelease>\IncludeInRelease{0000/00/00}%
113 <latexrelease> \{\@kernel@before@enddocument\}{kernel before hook}%
114 <latexrelease>
115 <latexrelease>\let\@kernel@before@enddocument\@empty
116 <latexrelease>
117 <latexrelease>\EndIncludeInRelease
118 <*2ekernel>

```

(End definition for \@kernel@before@enddocument.)

`\@testdef`

```

119 <def>\@testdef #1#2#3{%
120 <def>\reserved@a{#3}\expandafter \ifx \cename #1@#2\endcename
121 <reserved@a \else \@tempwattrue \fi}

```

(End definition for \@testdef.)

Reading data from auxiliary files (like `.toc` normally happens in vertical mode and it therefore doesn't matter if line endings are converted to spaces by \TeX during that process.

However, especially the `.toc` file might be read in L-R mode (in cases the `\tableofcontents` attempts to put, say a list of sub-sections as a paragraph. In that case the newlines after a line like

`\contentsline {subsubsection}{\numberline {1.1.1}A C-head}{2}`

might result in spurious spaces (e.g., when that level is not included).

That could be fixed by reading in the file using `\endlinechar=-1` but that has the danger that it drops some valid endlines that should be converted to spaces (for example when the user edited the TOC and then used `\nofiles` to preserve it).

So the approach taken instead is this:

- `\addcontentsline` adds the command `\protected@file@percent` to the end of the second argument of `\@writefile` that is written to the `.aux`. As the name indicates this is a protected macro so it doesn't change if it is written out.
- When the `.aux` is read back in at the end of the run, `\@writefile` is executed and writes its second argument unmodified to the file with the extension given by its first argument. Or rather that was how it was in the past.
- Instead we change `\@writefile` slightly: basically it looks at the second argument and if the last token in there is `\protected@file@percent` then it is replaced by a percent character and that is then written out. If not (for example, if the data came from a user issued `\addtocontents`, or from some package that uses `\@writefile` for writing its own files) then the command behaves exactly as before.

`\protected@file@percent` Dummy cs to be replaced by a percent sign inside `\@writefile`. If it survives (when used incorrectly) it will expand to nothing in a typsetting context.

```

122 </2ekernel>
123 <*2ekernel | latexrelease>
124 <latexrelease>\IncludeInRelease{2018/12/01}%
125 <latexrelease>          {\protected@file@percent}{Mask line endings}%
126 \protected\def\protected@file@percent{}

```

(End definition for \protected@file@percent.)

`\add@percent@to@temptokena` Helper function which is used to inspect a sequence of tokens (the second argument of `\@writefile` and if the last token is `\protected@file@percent` it will replace it by a harmless percent. The result is saved in `\@temptokena` for later use.

```

127 \catcode'\^^A=9
128 \long\gdef\add@percent@to@temptokena
129   #1\protected@file@percent#2\add@percent@to@temptokena

```

When we call this macro in `\@writefile` we stick in `\@empty` at the beginning, so that in case the tokenlist consists of a single brace group the braces aren't stripped. The `\expandafter` then expands this extra token away again.

```

130   {\expandafter\ifx\expandafter X\detokenize{#2}X\expandafter\dont@add@percent@to@temptokena
131     \expandafter\do@add@percent@to@temptokena\fi{#1}}
132 \long\def\dont@add@percent@to@temptokena#1{%
133   \@temptokena\expandafter{#1}}

```

`latexrelease` will read this code in high-speed mode in certain situations. During that it will only look for `\if` tests but not actually execute the `\catcode` change above. As a result it will drop anything after the `%` character in the definition. Therefore the `\fi` needs to be on the next line and we need locally another comment character to avoid getting spaces into the definition—a weird problem :-)

```

134 \begingroup

```

```

135 \catcode'\%=12
136 \catcode'\^^A=14
137 \long\gdef\do@add@percent@to@temptokena#1{\@temptokena\expandafter{#1%\^^A

```

Can't be on the same line as the % — see above.

```

138   }}
139 \endgroup

```

(End definition for \add@percent@to@temptokena.)

\@writefile

```

140 \long\def\@writefile#1#2{%
141   \@ifundefined{tf@#1}\relax
142   {%

```

If we write to the file we first prepare #2 using \add@percent@to@temptokena and then write the token register out.

```

143     \add@percent@to@temptokena
144     \@empty#2\protected@file@percent
145     \add@percent@to@temptokena
146     \immediate\write\csname tf@#1\endcsname{\the\@temptokena}%
147   }%
148 }

149 </2ekernel | latexrelease>
150 <latexrelease>\EndIncludeInRelease
151 <latexrelease>\IncludeInRelease{0000/00/00}%
152 <latexrelease>          {\protected@file@percent}{Mask line endings}%
153 <latexrelease>\let\protected@file@percent\@undefined
154 <latexrelease>\let\add@percent@to@temptokena\@undefined
155 <latexrelease>\let\do@add@percent@to@temptokena\@undefined
156 <latexrelease>\let\dont@add@percent@to@temptokena\@undefined
157 <latexrelease>\long\def\@writefile#1#2{%
158 <latexrelease>   \@ifundefined{tf@#1}\relax
159 <latexrelease>   {\@temptokena{#2}%
160 <latexrelease>     \immediate\write\csname tf@#1\endcsname{\the\@temptokena}%
161 <latexrelease>   }%
162 <latexrelease>}
163 <latexrelease>\EndIncludeInRelease
164 <*2ekernel>

```

(End definition for \@writefile.)

\stop

```

165 \def\stop{\clearpage\deadcycles\z@\let\par\@@par\@@end}

```

(End definition for \stop.)

Historical L^AT_EX 2.09 comments (not necessarily accurate any more):

```

166 \everypar{\@nodocument} %% To get an error if text appears before the
167 \nullfont                %% \begin{document}

```

`\begin`, `\end`, and `\@checkend` changed so `\end{document}` will catch an unmatched `\begin`. Changed 24 May 89 as suggested by Frank Mittelbach and Rainer Sch\"opf.

```

\begin{NAME} ==
BEGIN
  IF \NAME undefined THEN \reserved@a == BEGIN report error END
                        ELSE \reserved@a ==
                            (\@currentvir :=L NAME) \NAME
  FI
  @ignore :=G F      %% Added 30 Nov 88
  \begingroup
  \@endpe := F
  \@currentvir :=L NAME
  \NAME
END

\end{NAME} ==
BEGIN
  \endNAME
  \@checkend{NAME}
  \endgroup
  IF \@endpe = T      %% \@endpe set True by \@endparenv
    THEN \@doendpe    %% \@doendpe redefines \par and \everypar
                    %% to suppress paragraph indentation in
                    %% immediately following text
  FI
  IF @ignore = T
    THEN @ignore :=G F
        \ignorespaces
  FI
END

\@checkend{NAME} ==
BEGIN
  IF \@currentvir = NAME
    ELSE \@badend{NAME}
  FI
END

```

End of historical L^AT_EX 2.09 comments.

```

\begin
168 </2kernel>
169 <*2kernel | latexrelease>
170 <latexrelease>\IncludeInRelease{2020/10/01}%
171 <latexrelease>          {\begin}{Use hook system}%
172 \DeclareRobustCommand*\begin[1]{%
173   \UseHook{env/#1/before}%
174   \@ifundefined{#1}%
175     {\def\reserved@a{\@latex@error{Environment #1 undefined}\@eha}}%

```

```

176   {\def\reserved@a{\def\@currentenv{#1}%
177     \edef\@currentvline{\on@line}%
178     \@execute@begin@hook{#1}%
179     \csname #1\endcsname}}%
180   \ignorefalse
181   \begingroup\@endpfalse\reserved@a}

```

Before the `\document` code is executed we have to first undo the `\endgroup` as there should be none for this environment to avoid that changes on top-level unnecessarily go to TeX's savestack, and we have to initialize all hooks in the hook system. So we need to test for this environment name. But once it has been found all this testing is no longer needed and so we redefine `\@execute@begin@hook` to simply use the hook.

```

182 \def\@execute@begin@hook #1{%
183   \expandafter\ifx\csname #1\endcsname\document
184     \endgroup
185     \gdef\@execute@begin@hook##1{\UseHook{env/##1/begin}}%
186     \expl@@@initialize@all@@
187   \fi

```

If this is an environment before `\begin{document}` we just run the hook so this can be outside the test.

```

188   \UseHook{env/#1/begin}%
189 }

```

The top level definition for `\end.` for an explanation see below (this is the same as the 2019 version where it was introduced, but for rollback we have to repeat it).

```

190 \edef\end
191   {\unexpanded{%
192     \romannumeral
193     \ifx\protect\@typeset@protect
194       \expandafter      %1
195       \expandafter      %2
196       \expandafter      %1
197       \expandafter      %3 expands the \csname inside \end<space>
198       \expandafter      %1
199       \expandafter      %2 expands \end<space>
200       \expandafter      %1 expands the \else
201       \z@
202     \else
203       \expandafter\z@\expandafter\protect
204     \fi
205   }%
206   \expandafter\noexpand\csname end \endcsname
207 }

```

Version that adds hooks (so different from the 2019 version). It fixes tlb3722 but the change should perhaps be made in `tabularx` instead.

```

208 \@namedef{end }#1{%
209   \romannumeral
210   \IfHookEmptyTF{env/#1/end}%
211     {\expandafter\z@}%
212     {\z@\UseHook{env/#1/end}}%
213   \csname end#1\endcsname\@checkend{#1}%
214   \expandafter\endgroup\if@endpe\@doendpe\fi
215   \UseHook{env/#1/after}%

```



```

216 \if@ignore\@ignorefalse\ignorespaces\fi
217 }

```

Version without the fix for tlb3722 for the record:

```

218 %\@namedef{end }#1{%
219 % \UseHook{env/#1/end}%
220 % \csname end#1\endcsname\@checkend{#1}%
221 % \expandafter\endgroup\if@endpe\@doendpe\fi
222 % \UseHook{env/#1/after}%
223 % \if@ignore\@ignorefalse\ignorespaces\fi}%

224 </2ekernel | latexrelease>
225 <latexrelease>\EndIncludeInRelease

226 <latexrelease>\IncludeInRelease{2019/10/01}%
227 <latexrelease> \begin}{Making \begin/\end robust}%
228 <latexrelease>\DeclareRobustCommand\begin[1]{%
229 <latexrelease> \ifundefined{#1}%
230 <latexrelease> {\def\reserved@a{\@latex@error{Environment #1 undefined}\@eha}}%
231 <latexrelease> {\def\reserved@a{\def\@currentvir{#1}%
232 <latexrelease> \edef\@currentvline{\on@line}%
233 <latexrelease> \csname #1\endcsname}}%
234 <latexrelease> \@ignorefalse
235 <latexrelease> \begingroup\@endpefalse\reserved@a}

```

A version that doesn't start out with `\relax` when in typesetting mode would be the following, but since `\begin` issues a `\begingroup` it wouldn't help much with respect to allowing things like `\noalign` or `\multicolumn` inside.

```

236 %\edef\begin
237 % {\unexpanded{%
238 % \ifx\protect\@typeset@protect
239 % \expandafter\@gobble
240 % \fi
241 % \protect
242 % }%
243 % \expandafter\noexpand\csname begin \endcsname
244 % }
245 %\@namedef{begin }#1{%
246 % \ifundefined{#1}%
247 % {\def\reserved@a{\@latex@error{Environment #1 undefined}\@eha}}%
248 % {\def\reserved@a{\def\@currentvir{#1}%
249 % \edef\@currentvline{\on@line}%
250 % \csname #1\endcsname}}%
251 % \@ignorefalse
252 % \begingroup\@endpefalse\reserved@a}

```

While `\begin` was made robust simply by using `\DeclareRobustCommand` we need to be a bit more subtle with `\end` as there are packages out there that try to look into the top-level contents of `\end{foo}` (that is at the expansion of `\endfoo`) to see if it contains certain macros. This is done by hitting `\end{foo}` with three `\expandafters`, the first to get

```
\csname endfoo\endcsname \quad \@checkend{foo}% etc.
```

the second to expand the `\csname`, i.e., to get to

```
\endfoo                                \@checkend{foo}% etc.
```

and the third to finally get to the top-level content of `\endfoo`, i.e.

```
<top-level content of \endfoo> \@checkend{foo}% etc.
```

Therefore a robust replacement should produce the same results after three expansions (there first is obviously different).

Basically the definition of `\end` should either produce `\protect\end_` (when not doing typesetting) or it should produce `\end_` (without the `\protect`) when doing typesetting. Furthermore, it should (when in typesetting mode) show exactly the same result as `\end_` (which is the original fragile definition of `\end`) when you expand either of them twice, i.e.,

```
\endfoo                                \@checkend{foo}% etc.
```

That is achieved with the code below (which is worth studying carefully).

There is some trickery involved here: in particular we use `\romannumeral` to change a single expansion into three successive expansions in one go. That primitive expands until it has scanned a number (0 in this case, so it doesn't produce any output) and so it allows us to place arbitrary many `\expandafter` inside that are all going to be executed when `\romannumeral` is hit by a single `\expandafter`.

```
253 <latexrelease>\edef\end
254 <latexrelease>  {\unexpanded{%
255 <latexrelease>    \romannumeral
256 <latexrelease>    \ifx\protect\@typeset@protect
257 <latexrelease>    \expandafter      %1
258 <latexrelease>    \expandafter      %2
259 <latexrelease>    \expandafter      %1
260 <latexrelease>    \expandafter      %3 expands the \cename inside \end<space>
261 <latexrelease>    \expandafter      %1
262 <latexrelease>    \expandafter      %2 expands \end<space>
263 <latexrelease>    \expandafter      %1 expands the \else
264 <latexrelease>    \z@
265 <latexrelease>    \else
266 <latexrelease>    \expandafter\z@\expandafter\protect
267 <latexrelease>    \fi
268 <latexrelease>  }%
269 <latexrelease>  \expandafter\noexpand\cename end \endcename
270 <latexrelease>  }
```

And here is the original definition of `\end` the way it was in L^AT_EX for several decades now hidden in `\end_`.

```
271 <latexrelease>\@namedef{end }#1{%
272 <latexrelease>  \cename end#1\endcename\@checkend{#1}%
273 <latexrelease>  \expandafter\endgroup\if@endpe\@doendpe\fi
274 <latexrelease>  \if@ignore\@ignorefalse\ignorespaces\fi}
275 <latexrelease>\EndIncludeInRelease
```

An here the rollback in case that is ever needed.

```
276 <latexrelease>\IncludeInRelease{0000/00/00}%
277 <latexrelease>      {\begin}{Making \begin/\end robust}%
278 <latexrelease>\def\begin#1{%
279 <latexrelease>  \@ifundefined{#1}%
```

```

280 <latexrelease>      {\def\reserved@a{\@latex@error{Environment #1 undefined}\@eha}}%
281 <latexrelease>      {\def\reserved@a{\def\@currentvir{#1}%
282 <latexrelease>        \edef\@currentvline{\on@line}%
283 <latexrelease>        \csname #1\endcsname}}%
284 <latexrelease>      \ignorefalse
285 <latexrelease>      \begingroup\@endpfalse\reserved@a}
286 <latexrelease>\def\end#1{%
287 <latexrelease>      \csname end#1\endcsname\@checkend{#1}%
288 <latexrelease>      \expandafter\endgroup\if@endpe\@doendpe\fi
289 <latexrelease>      \if@ignore\ignorefalse\ignorespaces\fi}
290 <latexrelease>

```

Also undo the internal commands as some packages unfortunately test for their existence instead of using `\IfFormatAtLeastTF`.

```

291 <latexrelease>\expandafter\let\csname begin \endcsname\@undefined
292 <latexrelease>\expandafter\let\csname end \endcsname\@undefined
293 <latexrelease>
294 <latexrelease>\EndIncludeInRelease
295 <*2ekernel>

```

(End definition for `\begin` and `\end`.)

`\@checkend`

```

296 \def\@checkend#1{\def\reserved@a{#1}\ifx
297     \reserved@a\@currentvir \else\@badend{#1}\fi}

```

(End definition for `\@checkend`.)

`\@currentvline` We do need a default value for `\@currentvline` on top-level since the document environment cancels the brace group. This means that a mismatch with `\begin{document}` will not produce a line number. Thus the outer default must be `\@empty` or we will end up with two spaces.

```

298 \let\@currentvline\@empty

```

(End definition for `\@currentvline`.)

`\AtBeginEnvironment` We provide 4 high-level hook interfaces directly, the others only when etoolbox is loaded

`\AtEndEnvironment`

`\BeforeBeginEnvironment`

`\AfterEndEnvironment`

```

299 </2ekernel>
300 <*2ekernel | latexrelease>
301 <latexrelease>\IncludeInRelease{2020/10/01}%
302 <latexrelease>      {\AtBeginEnvironment}{Hooks for environments}%
303 \newcommand\AtBeginEnvironment[2][.]    {\AddToHook{env/#2/begin}[#1]}
304 \newcommand\AtEndEnvironment[2][.]      {\AddToHook{env/#2/end}[#1]}
305 \newcommand\BeforeBeginEnvironment[2][.]{\AddToHook{env/#2/before}[#1]}
306 \newcommand\AfterEndEnvironment[2][.]    {\AddToHook{env/#2/after}[#1]}
307 </2ekernel | latexrelease>
308 <latexrelease>\EndIncludeInRelease
309 <latexrelease>\IncludeInRelease{0000/00/00}%
310 <latexrelease>      {\AtBeginEnvironment}{Hooks for environments}%
311 <latexrelease>
312 <latexrelease>\let\AtBeginEnvironment\@undefined
313 <latexrelease>\let\AtEndEnvironment\@undefined
314 <latexrelease>\let\BeforeBeginEnvironment\@undefined

```

```

315 <latexrelease>\let\AfterEndEnvironment\@undefined
316 <latexrelease>
317 <latexrelease>\EndIncludeInRelease
318 <*2ekernel>

```

(End definition for `\AtBeginEnvironment` and others. These functions are documented on page 182.)

1.2 Center, Flushright, Flushleft

```

319 \message{center,}

```

Historical *L^AT_EX* 2.09 comments (not necessarily accurate any more):

```

\center, \flushright and \flushleft set
  \rightskip = 0pt or \@flushglue (as appropriate)
  \leftskip  = 0pt or \@flushglue (as appropriate)
  \parindent = 0pt
  \parfillskip = 0pt. (except \flushleft)
  \\\          == \par \vskip -\parskip
  \|[LENGTH] == \\\ \vskip LENGTH
  \\\*         == \par \penalty 10000 \vskip -\parskip
  \\\*[LEN]    == \\\* \vskip LENGTH

```

They invoke the `trivlist` environment to handle vertical spacing before and after them.

`\centering`, `\raggedright` and `\raggedleft` are the declaration analogs of the above.

`\raggedright` has a more universal effect, however. It sets `\@rightskip := flushglue`. Every environment, like the list environments, that set `\rightskip` to its 'normal' value set it to `\@rightskip`

End of historical *L^AT_EX* 2.09 comments.

`\@centercr`

```

320 </2ekernel>
321 <*2ekernel | latexrelease>
322 <latexrelease>\IncludeInRelease{2020/02/02}%
323 <latexrelease>          {\@centercr}{Make robust}%
324 \protected\def\@centercr{\ifhmode \unskip\else \@nolnerr\fi
325   \par\@ifstar{\nobreak\@xcentercr}\@xcentercr}
326 </2ekernel | latexrelease>

327 <latexrelease>\EndIncludeInRelease
328 <latexrelease>\IncludeInRelease{0000/00/00}%
329 <latexrelease>          {\@centercr}{Make robust}%
330 <latexrelease>
331 <latexrelease>\def\@centercr{\ifhmode \unskip\else \@nolnerr\fi
332 <latexrelease>          \par\@ifstar{\nobreak\@xcentercr}\@xcentercr}
333 <latexrelease>
334 <latexrelease>\EndIncludeInRelease
335 <*2ekernel>

```

```

(End definition for \@centercr.)

\@xcentercr
336 \def\@xcentercr{\addvspace{-\parskip}\@ifnextchar
337   [\@icentercr\ignorespaces}

(End definition for \@xcentercr.)

\@icentercr
338 \</2ekernel>
339 \<*2ekernel | latexrelease>
340 \<latexrelease>\IncludeInRelease{2020/10/01}%
341 \<latexrelease>           {\@icentercr}{centering, etc support calc}%
342 \def\@icentercr[#1]{\@vspace@calcify{#1}\ignorespaces}
343 \</2ekernel | latexrelease>
344 \<latexrelease>\EndIncludeInRelease
345 \<latexrelease>\IncludeInRelease{0000/00/00}%
346 \<latexrelease>           {\@icentercr}{centering, etc support calc}%
347 \<latexrelease>
348 \<latexrelease>\def\@icentercr[#1]{\vskip #1\ignorespaces}
349 \<latexrelease>\EndIncludeInRelease
350 \<*2ekernel>

(End definition for \@icentercr.)

center We use \relax to prevent \item scanning too far.
351 \def\center{\trivlist \centering\item\relax}
352 \def\endcenter{\endtrivlist}

353 \</2ekernel>
354 \<*2ekernel | latexrelease>
355 \<latexrelease>\IncludeInRelease{2020/10/01}%
356 \<latexrelease>           {\centering}{Set finaldhypendemerits}%

\centering
357 \DeclareRobustCommand\centering{%
358   \let\\ \@centercr
359   \rightskip\@flushglue\leftskip\@flushglue
360   \finalhyphendemerits=\z@
361   \parindent\z@\parfillskip\z@skip}

(End definition for \centering.)

\raggedright
362 \DeclareRobustCommand\raggedright{%
363   \let\\ \@centercr\@rightskip\@flushglue \rightskip\@rightskip
364   \finalhyphendemerits=\z@
365   \leftskip\z@skip
366   \parindent\z@}

(End definition for \raggedright.)

```

`\raggedleft`

```
367 \DeclareRobustCommand\raggedleft{%
368   \let\\\@centercr
369   \rightskip\z@skip\leftskip\@flushglue
370   \finalhyphendemerits=\z@
371   \parindent\z@\parfillskip\z@skip}

(End definition for \raggedleft.)

372 </2kernel | latexrelease>
373 <latexrelease>\EndIncludeInRelease
374 <latexrelease>\IncludeInRelease{2019/10/01}%
375 <latexrelease>           {\centering}{Make commands robust}%
376 <latexrelease>
377 <latexrelease>\DeclareRobustCommand\centering{%
378 <latexrelease>  \let\\\@centercr
379 <latexrelease>  \rightskip\@flushglue\leftskip\@flushglue
380 <latexrelease>  \parindent\z@\parfillskip\z@skip}
381 <latexrelease>\DeclareRobustCommand\raggedright{%
382 <latexrelease>  \let\\\@centercr\@rightskip\@flushglue \rightskip\@rightskip
383 <latexrelease>  \leftskip\z@skip
384 <latexrelease>  \parindent\z@}
385 <latexrelease>\DeclareRobustCommand\raggedleft{%
386 <latexrelease>  \let\\\@centercr
387 <latexrelease>  \rightskip\z@skip\leftskip\@flushglue
388 <latexrelease>  \parindent\z@\parfillskip\z@skip}
389 <latexrelease>\EndIncludeInRelease
390 <latexrelease>
391 <latexrelease>\IncludeInRelease{0000/00/00}%
392 <latexrelease>           {\centering}{Make commands robust}%
393 <latexrelease>
394 <latexrelease>\kernel@make@fragile\centering
395 <latexrelease>\kernel@make@fragile\raggedright
396 <latexrelease>\kernel@make@fragile\raggedleft
397 <latexrelease>
398 <latexrelease>\EndIncludeInRelease
399 <*2kernel>
```

`\@rightskip`

```
400 \newskip\@rightskip \@rightskip \z@skip
```

(End definition for \@rightskip.)

flushleft We use `\relax` to prevent `\item` scanning too far.

```
401 \def\flushleft{\trivlist \raggedright\item\relax}
402 \def\endflushleft{\endtrivlist}
```

flushright We use `\relax` to prevent `\item` scanning too far.

```
403 \def\flushright{\trivlist \raggedleft\item\relax}
404 \def\endflushright{\endtrivlist}
```

1.3 Verbatim

405 \message{verbatim,}

The verbatim environment uses the fixed-width `\ttfamily` font, turns blanks into spaces, starts a new line for each carriage return (or sequence of consecutive carriage returns), and interprets *every* character literally. I.e., all special characters `\`, `{`, `$`, etc. are `\catcode'd` to 'other'.

The command `\verb` produces in-line verbatim text, where the argument is delimited by any pair of characters. E.g., `\verb #...#` takes '...' as its argument, and sets it verbatim in `\ttfamily` font.

The *-variants of these commands are the same, except that spaces print as the TeXbook's space character instead of as blank spaces.

\@vobeyspaces

406 {\catcode'\ =\active%
407 \gdef\@vobeyspaces{\catcode'\ \active\let \@xobeysp}}

(End definition for \@vobeyspaces.)

\@xobeysp

(End definition for \@xobeysp.)

\@xverbatim

\@sxverbatim

408 \begingroup \catcode '| =0 \catcode '[= 1
409 \catcode'] =2 \catcode '\ =12 \catcode '\ =12
410 \catcode'\ =12 |gdef|@xverbatim#1\end{verbatim}[#1\end[verbatim]]
411 |gdef|@sxverbatim#1\end{verbatim*}[#1\end[verbatim*]]
412 |endgroup

(End definition for \@xverbatim and \@sxverbatim.)

\@verbatim Real start of verbatim environment We use `\relax` to prevent `\item` scanning too far.

413 </2kernel>
414 <*2kernel | latexrelease>
415 <latexrelease>\IncludeInRelease{2017-04-15}{\@verbatim}%
416 <latexrelease> {Disable hyphenation in verbatim}%
417 \def\@verbatim{\trivlist \item\relax
418 \if@minipage\else\vskip\parskip\fi
419 \leftskip\@totalleftmargin\rightskip\z@skip
420 \parindent\z@\parfillskip\@flushglue\parskip\z@skip

Added `\@@par` to clear possible `\parshape` definition from a surrounding list (the verbatim guru says). Switch language when in vertical mode.

421 \@@par

Set `\language` here to suppress hyphenation. Done this way rather than setting `\hyphenchar` as that is a global setting.

422 \language\l@nohyphenation
423 \@tempwafalse
424 \def\par{%
425 \if@tempswa

A `\leavevmode` added: needed if, for example, a blank verbatim line is the first thing in a list item (wow!).

```

426     \leavevmode \null \@@par\penalty\interlinepenalty
427   \else
428     \@tempswatrue
429     \ifhmode\@@par\penalty\interlinepenalty\fi
430   \fi}%

```

To allow customization we hide the font used in a separate macro.

```

431   \let\do\@makeoother \dospecials
432   \obeylines \verbatim@font \@noligs

```

To avoid a breakpoint after the labels box, we remove the penalty put there by the list macros: another use of `\unpenalty`!

```

433   \everypar \expandafter{\the\everypar \unpenalty}%
434 }
435 \</2ekernel | latexrelease>
436 \<latexrelease>\EndIncludeInRelease
437 \<latexrelease>\IncludeInRelease{0000-00-00}{\@verbatim}%
438 \<latexrelease>      {Disable hyphenation in verbatim}%
439 \<latexrelease>\def\@verbatim{\trivlist \item\relax
440 \<latexrelease>  \if@minipage\else\vskip\parskip\fi
441 \<latexrelease>  \leftskip\@totalleftmargin\rightskip\z@skip
442 \<latexrelease>  \parindent\z@\parfillskip\@flushglue\parskip\z@skip
443 \<latexrelease>  \@@par
444 \<latexrelease>  \@tempswafalse
445 \<latexrelease>  \def\par{%
446 \<latexrelease>    \if@tempswa
447 \<latexrelease>      \leavevmode \null \@@par\penalty\interlinepenalty
448 \<latexrelease>    \else
449 \<latexrelease>      \@tempswatrue
450 \<latexrelease>      \ifhmode\@@par\penalty\interlinepenalty\fi
451 \<latexrelease>    \fi}%
452 \<latexrelease>  \let\do\@makeoother \dospecials
453 \<latexrelease>  \obeylines \verbatim@font \@noligs
454 \<latexrelease>  \hyphenchar\font\m@ne
455 \<latexrelease>  \everypar \expandafter{\the\everypar \unpenalty}%
456 \<latexrelease>}
457 \<latexrelease>\EndIncludeInRelease
458 \<*2ekernel>

```

(End definition for \@verbatim.)

\verbatim (RmS 93/09/19) Protected against ‘missing item’ error message triggered by empty verbatim environment.

```

459 \def\verbatim{\@verbatim \frenchspacing\@vobeyspaces \@xverbatim}
460 \def\endverbatim{\if@newlist \leavevmode\fi\endtrivlist}

```

(End definition for \verbatim and \endverbatim.)

\verbatim@font Macro to select the font used for verbatim typesetting. It also does other work if necessary for the font used.

```

461 \def\verbatim@font{\normalfont\ttfamily}

```


(End definition for \verbatim@font.)

```
462 \endkernel
463 \ifx\kernel\latexrelease
464 \ifx\kernel\includeinrelease{2018/12/01}%
465 \ifx\kernel\includeinrelease{\verbvisiblespace}{Setup visible space for \verb}%
```

\asciispace The character in slot 32, in typewriter fonts (historically) a visible space but in other fonts a real space or something else

```
466 \DeclareRobustCommand\asciispace{\char 32 }
```

(End definition for \asciispace.)

\verbvisiblespace This defines how to get a visible space in **\verb*** and friends. In classic T_EX this is just the slot 32, but in T_U encoded fonts we switch fonts and take the character from cmtt.

```
467 \ifx\Umathcode\@undefined
468 \let\verbvisiblespace\asciispace % Pdftex version
469 \else
470 \DeclareRobustCommand\verbvisiblespace
471 {\leavevmode{\usefont{OT1}{cmtt}{m}{n}\asciispace}} % xetex/luatex version
472 \fi
```

(End definition for \verbvisiblespace.)

\@setupverbvisiblespace In pdfT_EX a catcode 12 space will produce the character in slot 32 which is assumed to be a visible space character (in a typewriter font in OT1 or T1 encoding). In XeT_EX or LuaT_EX a font in T_U encoding is normally used and that has a real space in this slot. So what we do in this case is this: we check the definition of **\verbvisiblespace** and if it is **\asciispace** we assume that the char32 can be used (e.g., in pdfT_EX). We then redefine **\@xobeysp** so that after running **\@xobeyspaces** we get characters from slot 32 for each active space.

```
473 \def\@setupverbvisiblespace{%
474 \ifx\verbvisiblespace\asciispace
475 \let\@xobeysp\asciispace
476 \else
```

Otherwise we measure the width of a character in the mon-spaced current font and place a **\verbvisiblespace** into a box of the right width which we are then using as the character for a space. By default this will be the space character from OT1 cmtt but by changing **\verbvisiblespace** one could use, for example, the **\textvisiblespace** of the current typewriter font.

```
477 \setbox\z@\hbox{x}%
478 \setbox\@verbvisiblespacebox\hbox to\wd\z@{\hss\verbvisiblespace\hss}%
479 \def\@xobeysp{\leavevmode\copy\@verbvisiblespacebox}%
480 \fi
481 }
```

(End definition for \@setupverbvisiblespace.)

\@verbvisiblespacebox The box to hold the visible space character if it isn't in slot 32 in the current typewriter font.

```
482 \newbox\@verbvisiblespacebox
```

(End definition for \@verbvisiblespacebox.)

`verbatim*` For `verbatim*` we also set up the correct visible space character definition and then run `\@vobeyspaces`. As this code is not called as part of the normal `verbatim` environment (the method is done the other way around this time) we don't have to check if space is already active—it shouldn't be.

```

483 \namedef{verbatim*}{\@verbatim
484   \@setupverbvisiblespace
485   \frenchspacing\@vobeyspaces\@sxverbatim}
486 \expandafter\let\csname endverbatim*\endcsname =\endverbatim

487 </2ekernel | latexrelease>
488 <latexrelease>\EndIncludeInRelease
489 <latexrelease>\IncludeInRelease{0000/00/00}%
490 <latexrelease>           {\verbvisiblespace}{Setup visible space for \verb}%
491 <latexrelease>
492 <latexrelease>\namedef{verbatim*}{\@verbatim\@sxverbatim}
493 <latexrelease>
494 <latexrelease>\let\asciispace           \@undefined
495 <latexrelease>\let\verbvisiblespace     \@undefined
496 <latexrelease>\let\@setupverbvisiblespace \@undefined
497 <latexrelease>\let\@verbvisiblespacebox \@undefined
498 <latexrelease>\EndIncludeInRelease
499 <*2ekernel>

```

`\@sverb` Definitions of `\@sverb` and `\@verb` changed so `\verb+ foo+` does not lose leading blanks
`\@@sverb` when it comes at the beginning of a line. Change made 24 May 89. Suggested by Frank Mittelbach and Rainer Schöpf.

```

500 </2ekernel>
501 <*2ekernel | latexrelease>
502 <latexrelease>\IncludeInRelease{2020/10/01}%
503 <latexrelease>           {\@sverb}{Drop spaces before \verb delimiter}%

```

If the users types `\verb !~! foo` then surprisingly we would get the space as the delimiter and thus “!~!foo” in the output. To avoid this scenario we check if `#1` has the character code of a space, if so we recurse otherwise we call `\@@sverb` (which is the original definition of `\@sverb`).

```

504 \def\@sverb#1{\if\noexpand#1 \expandafter\@sverb\else\@@sverb{#1}\fi}

505 \def\@@sverb#1{%
506   \catcode'#1\active
507   \lccode'\~'#1%
508   \gdef\verb@balance@group{\verb@egroup
509     \@latexerror{\noexpand\verb illegal in command argument}\@ehc}%
510   \aftergroup\verb@balance@group
511   \lowercase{\let~\verb@egroup}%

```

If `\@sverb` is called from `\@verb` then space is already active and supposed to produce a real space. In this case we do nothing. Otherwise we run `\@setupverbvisiblespace` to setup the right visible space char and afterwards `\@vobeyspaces` to make it the definition for the active space character.

```

512   \ifnum\catcode'\ =\active
513   \else \@setupverbvisiblespace \@vobeyspaces \fi
514 }

```

```

515 </2ekernel | latexrelease>
516 <latexrelease>\EndIncludeInRelease
517 <latexrelease>\IncludeInRelease{2018/12/01}%
518 <latexrelease>          {\@sverb}{Setup visible space for \verb}%
519 <latexrelease>
520 <latexrelease>\def\@sverb#1{%
521 <latexrelease>  \catcode'#1\active
522 <latexrelease>  \lccode'\~'#1%
523 <latexrelease>  \gdef\verb@balance@group{\verb@egroup
524 <latexrelease>    \@latex@error{\noexpand\verb illegal in command argument}\@ehc}%
525 <latexrelease>  \aftergroup\verb@balance@group
526 <latexrelease>  \lowercase{\let~\verb@egroup}%
527 <latexrelease>  \ifnum\catcode'\ =\active
528 <latexrelease>  \else  \@setupverbvisiblespace \@vobeyspaces \fi
529 <latexrelease>}
530 <latexrelease>\let\@@sverb\@undefined
531 <latexrelease>\EndIncludeInRelease
532 <latexrelease>
533 <latexrelease>\IncludeInRelease{0000/00/00}%
534 <latexrelease>          {\@sverb}{Setup visible space for \verb}%
535 <latexrelease>\def\@sverb#1{%
536 <latexrelease>  \catcode'#1\active
537 <latexrelease>  \lccode'\~'#1%
538 <latexrelease>  \gdef\verb@balance@group{\verb@egroup
539 <latexrelease>    \@latex@error{\noexpand\verb illegal in command argument}\@ehc}%
540 <latexrelease>  \aftergroup\verb@balance@group
541 <latexrelease>  \lowercase{\let~\verb@egroup}}%
542 <latexrelease>
543 <latexrelease>\EndIncludeInRelease
544 <*2ekernel>

```

(End definition for \@sverb and \@@sverb.)

\@makeoother

```

545 \def\@makeoother#1{\catcode'#112\relax}

```

(End definition for \@makeoother.)

\verb@balance@group

```

546 \let\verb@balance@group\@empty

```

(End definition for \verb@balance@group.)

\verb@egroup

```

547 \def\verb@egroup{\global\let\verb@balance@group\@empty\egroup}

```

(End definition for \verb@egroup.)

\verb@eol@error

```

548 \begingroup
549   \obeylines%
550   \gdef\verb@eol@error{\obeylines%
551     \def~M{\verb@egroup\@latex@error{%
552       \noexpand\verb ended by end of line}\@ehc}}%
553 \endgroup

```

(End definition for \verb@eol@error.)

\verb Typesetting a small piece verbatim.

```
554 </2ekernel>
555 <*2ekernel | latexrelease>
556 <latexrelease>\IncludeInRelease{2017-04-15}{\verb}%
557 <latexrelease> {Disable hyphenation in verb}%
558 \def\verb{\relax\ifmmode\hbox\else\leavevmode\null\fi
559   \bgroup
560   \verb@eol@error \let\do\@makeoother \dospecials
561   \verbatim@font\@noligs
```

Set \language here to suppress hyphenation. Done this way rather than setting \hyphenchar as that is a global setting.

```
562   \language\l@nohyphenation
563   \ifstar\@sverb\@verb}
564 </2ekernel | latexrelease>
565 <latexrelease>\EndIncludeInRelease
566 <latexrelease>\IncludeInRelease{0000-00-00}{\verb}%
567 <latexrelease> {Disable hyphenation in verb}%
568 <latexrelease>\def\verb{\relax\ifmmode\hbox\else\leavevmode\null\fi
569 <latexrelease> \bgroup
570 <latexrelease> \verb@eol@error \let\do\@makeoother \dospecials
571 <latexrelease> \verbatim@font\@noligs
572 <latexrelease> \ifstar\@sverb\@verb}
573 <latexrelease>\EndIncludeInRelease
574 <*2ekernel>
```

(End definition for \verb.)

\@verb

```
575 \def\@verb{\@vobeyspaces \frenchspacing \@sverb}
```

(End definition for \@verb.)

\verbatim@nolig@list

```
576 \def\verbatim@nolig@list{\do\‘\do\<\do\>\do\,\do\’\do\~}
```

(End definition for \verbatim@nolig@list.)

\do@noligs

```
577 \def\do@noligs#1{%
578   \catcode‘#1\active
579   \begingroup
580   \lccode‘~‘#1\relax
581   \lowercase{\endgroup\def~{\leavevmode\kern\z@\char‘#1}}}
```

(End definition for \do@noligs.)

\@noligs To stay compatible with packages that use \@noligs we keep it.

```
582 \def\@noligs{\let\do\do@noligs \verbatim@nolig@list}
```

(End definition for \@noligs.)

```
583 </2ekernel>
```

File H

ltmath.dtx

1 Math setup

This file contains a lot of the original plain T_EX code, as well as the L^AT_EX environments for math. It still needs sorting out.

```
1 \*2kernel)
2 \message{math definitions,}
```

1.1 Math commands based on plain T_EX

1.1.1 The log-like functions

`\log` The standard operators:

```
3 \DeclareRobustCommand\log{\mathop{\operator@font log}\nolimits}
4 \DeclareRobustCommand\lg{\mathop{\operator@font lg}\nolimits}
5 \DeclareRobustCommand\ln{\mathop{\operator@font ln}\nolimits}
6 \DeclareRobustCommand\lim{\mathop{\operator@font lim}}
7 \DeclareRobustCommand\limsup{\mathop{\operator@font lim}\,sup}}
8 \DeclareRobustCommand\liminf{\mathop{\operator@font lim}\,inf}}
9 \DeclareRobustCommand\sin{\mathop{\operator@font sin}\nolimits}
10 \DeclareRobustCommand\arcsin{\mathop{\operator@font arcsin}\nolimits}
11 \DeclareRobustCommand\sinh{\mathop{\operator@font sinh}\nolimits}
12 \DeclareRobustCommand\cos{\mathop{\operator@font cos}\nolimits}
13 \DeclareRobustCommand\arccos{\mathop{\operator@font arccos}\nolimits}
14 \DeclareRobustCommand\cosh{\mathop{\operator@font cosh}\nolimits}
15 \DeclareRobustCommand\tan{\mathop{\operator@font tan}\nolimits}
16 \DeclareRobustCommand\arctan{\mathop{\operator@font arctan}\nolimits}
17 \DeclareRobustCommand\tanh{\mathop{\operator@font tanh}\nolimits}
18 \DeclareRobustCommand\cot{\mathop{\operator@font cot}\nolimits}
19 \DeclareRobustCommand\coth{\mathop{\operator@font coth}\nolimits}
20 \DeclareRobustCommand\sec{\mathop{\operator@font sec}\nolimits}
21 \DeclareRobustCommand\csc{\mathop{\operator@font csc}\nolimits}
22 \DeclareRobustCommand\max{\mathop{\operator@font max}}
23 \DeclareRobustCommand\min{\mathop{\operator@font min}}
24 \DeclareRobustCommand\sup{\mathop{\operator@font sup}}
25 \DeclareRobustCommand\inf{\mathop{\operator@font inf}}
26 \DeclareRobustCommand\arg{\mathop{\operator@font arg}\nolimits}
27 \DeclareRobustCommand\ker{\mathop{\operator@font ker}\nolimits}
28 \DeclareRobustCommand\dim{\mathop{\operator@font dim}\nolimits}
29 \DeclareRobustCommand\hom{\mathop{\operator@font hom}\nolimits}
30 \DeclareRobustCommand\det{\mathop{\operator@font det}}
31 \DeclareRobustCommand\exp{\mathop{\operator@font exp}\nolimits}
32 \DeclareRobustCommand\Pr{\mathop{\operator@font Pr}}
33 \DeclareRobustCommand\gcd{\mathop{\operator@font gcd}}
34 \DeclareRobustCommand\deg{\mathop{\operator@font deg}\nolimits}
```

(End definition for \log.)

`\bmod` And some operators have to be done by hand:

```

35 \DeclareRobustCommand\bmod{%
36   \nonscript\mskip-\medmuskip\mkern5mu%
37   \mathbin{\operator@font mod}\penalty900\mkern5mu%
38   \nonscript\mskip-\medmuskip}

```

(End definition for \bmod.)

\pmod

```

39 \DeclareRobustCommand\pmod[1]{%
40   \allowbreak\mkern18mu({\operator@font mod}\,\,\,#1)}

```

(End definition for \pmod.)

1.1.2 Biggggg

\big Variants on \big and friends for use with delimiters:

```

41 \DeclareRobustCommand\bigl{\mathopen\big}
42 \DeclareRobustCommand\bigm{\mathrel\big}
43 \DeclareRobustCommand\bigl{\mathclose\big}
44 \DeclareRobustCommand\Bigl{\mathopen\Big}
45 \DeclareRobustCommand\Bigm{\mathrel\Big}
46 \DeclareRobustCommand\Bigl{\mathclose\Big}
47 \DeclareRobustCommand\biggl{\mathopen\bigg}
48 \DeclareRobustCommand\biggm{\mathrel\bigg}
49 \DeclareRobustCommand\biggr{\mathclose\bigg}
50 \DeclareRobustCommand\Biggl{\mathopen\Bigg}
51 \DeclareRobustCommand\Biggm{\mathrel\Bigg}
52 \DeclareRobustCommand\Biggr{\mathclose\Bigg}

```

(End definition for \big.)

1.1.3 The UNSORTED Rest

The other math commands are lifted from plain T_EX.

\jot

```

53 \newdimen\jot
54 \jot=3pt

```

(End definition for \jot.)

\interdisplaylinepenalty

```

55 \newcount\interdisplaylinepenalty
56 \interdisplaylinepenalty=100

```

(End definition for \interdisplaylinepenalty.)

\choose

```

57 \def\choose{\atopwithdelims()}

```

(End definition for \choose.)

\brack

```

58 \def\brack{\atopwithdelims[]}

```

(End definition for \brack.)

```

\brace
59 \def\brace{\atopwithdelims\{\}}
(End definition for \brace.)

\mathpalette
60 \def\mathpalette#1#2{%
61   \mathchoice
62     {#1\displaystyle{#2}}%
63     {#1\textstyle{#2}}%
64     {#1\scriptstyle{#2}}%
65     {#1\scriptscriptstyle{#2}}
(End definition for \mathpalette.)

\root
\rootbox
\root
66 \newbox\rootbox
67 \def\root#1\of{%
68   \setbox\rootbox\hbox{$\m@th\scriptscriptstyle{#1}$}%
69   \mathpalette\root
70 \def\root#1#2{%
71   \setbox\rootbox\hbox{$\m@th#1\sqrt{#2}$}%
72   \dimen@=\ht\rootbox \advance\dimen@-\dp\rootbox
73   \mkern5mu\raise.6\dimen@\copy\rootbox
74   \mkern-10mu\box\rootbox
(End definition for \root, \rootbox, and \root.)

\phantom
\hphantom
\phantom
75 \newif\ifv@
76 \newif\ifh@
77 \kernel
78 \kernel | latexrelease
79 \kernel | latexrelease\IncludeInRelease{2019/10/01}%
80 \kernel | latexrelease\phantom{Make commands robust}%
81 \DeclareRobustCommand\phantom{\v@true\h@false\ph@nt}
82 \DeclareRobustCommand\hphantom{\v@false\h@true\ph@nt}
83 \DeclareRobustCommand\phantom{\v@true\h@true\ph@nt}

84 \DeclareRobustCommand\mathstrut{\phantom{}}

\mathstrut
85 \kernel | latexrelease
86 \kernel | latexrelease\EndIncludeInRelease
87 \kernel | latexrelease\IncludeInRelease{0000/00/00}%
88 \kernel | latexrelease\phantom{Make commands robust}%
89 \kernel | latexrelease
90 \kernel | latexrelease\kernel@make@fragile\phantom
91 \kernel | latexrelease\kernel@make@fragile\hphantom
92 \kernel | latexrelease\kernel@make@fragile\phantom
93 \kernel | latexrelease\kernel@make@fragile\mathstrut
94 \kernel | latexrelease
95 \kernel | latexrelease\EndIncludeInRelease
96 \kernel

```

```

97 \def\ph@nt{%
98   \ifmmode
99     \expandafter\mathpalette\expandafter\mathph@nt
100   \else
101     \expandafter\makeph@nt
102   \fi}

103 \def\makeph@nt#1{%
104   \setbox\z@\hbox{\color@begingroup#1\color@endgroup}\finph@nt}

105 \def\mathph@nt#1#2{%
106   \setbox\z@\hbox{\$ \m@th#1{#2}$}\finph@nt}

107 \</2ekernel>
108 \<*2ekernel | latexrelease>
109 \<latexrelease>\IncludeInRelease{2018/12/01}%
110 \<latexrelease>          {\finph@nt}{Start LR-mode}%
111 \def\finph@nt{%
112   \setbox\tw@\null
113   \ifv@ \ht\tw@\ht\z@ \dp\tw@\dp\z@\fi
114   \ifh@ \wd\tw@\wd\z@\fi

115   \leavevmode@ifvmode\box\tw@}
116 \</2ekernel | latexrelease>
117 \<latexrelease>\EndIncludeInRelease
118 \<latexrelease>\IncludeInRelease{0000/00/00}%
119 \<latexrelease>          {\finph@nt}{Start LR-mode}%
120 \<latexrelease>\def\finph@nt{%
121 \<latexrelease>   \setbox\tw@\null
122 \<latexrelease>   \ifv@ \ht\tw@\ht\z@ \dp\tw@\dp\z@\fi
123 \<latexrelease>   \ifh@ \wd\tw@\wd\z@\fi \box\tw@}
124 \<latexrelease>\EndIncludeInRelease
125 \<*2ekernel>

```

(End definition for \phantom and others.)

\smash

```

126 \DeclareRobustCommand\smash{%
127   \relax % \relax, in case this comes first in \halign
128   \ifmmode
129     \expandafter\mathpalette\expandafter\mathsm@sh
130   \else
131     \expandafter\makesm@sh
132   \fi}

133 \def\makesm@sh#1{%
134   \setbox\z@\hbox{\color@begingroup#1\color@endgroup}\finsm@sh}
135 \def\mathsm@sh#1#2{%
136   \setbox\z@\hbox{\$ \m@th#1{#2}$}\finsm@sh}

137 \</2ekernel>
138 \<*2ekernel | latexrelease>
139 \<latexrelease>\IncludeInRelease{2018/12/01}%
140 \<latexrelease>          {\finsm@sh}{Start LR-mode}%
141 \def\finsm@sh{\ht\z@\z@ \dp\z@\z@ \leavevmode@ifvmode\box\z@}
142 \</2ekernel | latexrelease>
143 \<latexrelease>\EndIncludeInRelease

```



```

144 <latexrelease>\IncludeInRelease{0000/00/00}%
145 <latexrelease>{\finsm@sh}{Start LR-mode}%
146 <latexrelease>\def\finsm@sh{\ht\z@\z@ \dp\z@\z@ \box\z@}
147 <latexrelease>\EndIncludeInRelease
148 <*2ekernel>

(End definition for \smash.)

\buildrel

149 \def\buildrel#1\over#2{\mathrel{\mathop{\kern\z@#2}\limits^{#1}}}

(End definition for \buildrel.)

</2ekernel>
<*2ekernel | latexrelease>
152 <latexrelease>\IncludeInRelease{2019/10/01}%
153 <latexrelease>{\cases}{Make commands robust}%

\cases

154 \DeclareRobustCommand*\cases[1]{\left\{\,\,\vcenter{\normalbaselines\m@th
155 \ialign{##\hfil$&\quad{##}\hfil\crr#1\crr}}\right.}

(End definition for \cases.)

\matrix

156 \DeclareRobustCommand*\matrix[1]{\null\,\vcenter{\normalbaselines\m@th
157 \ialign{\hfil$##$\hfil&\quad\hfil$##$\hfil\crr
158 \mathstrut\crr\noalign{\kern-\baselineskip}
159 #1\crr\mathstrut\crr\noalign{\kern-\baselineskip}}}\,,}

(End definition for \matrix.)

\pmatrix

160 \DeclareRobustCommand*\pmatrix[1]{\left(\matrix{#1}\right)}

(End definition for \pmatrix.)

</2ekernel | latexrelease>
162 <latexrelease>\EndIncludeInRelease
163 <latexrelease>\IncludeInRelease{0000/00/00}%
164 <latexrelease>{\cases}{Make commands robust}%
165 <latexrelease>
166 <latexrelease>\kernel@make@fragile\cases
167 <latexrelease>\kernel@make@fragile\matrix
168 <latexrelease>\kernel@make@fragile\pmatrix
169 <latexrelease>
170 <latexrelease>\EndIncludeInRelease
171 <*2ekernel>

\bordermatrix

172 \def\bordermatrix#1{\begingroup \m@th
173 \@tempdima 8.75\p@
174 \setbox\z@\vbox{%
175 \def\cr{\crr\noalign{\kern2\p@\global\let\cr\endline}}%
176 \ialign{##$\hfil\kern2\p@\kern\@tempdima&\thinspace\hfil$##$\hfil
177 &\quad\hfil$##$\hfil\crr

```

```

178 \omit\strut\hfil\crr\noalign{\kern-\baselineskip}%
179 #1\crr\omit\strut\cr}}%
180 \setbox\tw@\vbox{\unvcopy\z@\global\setbox\@ne\lastbox}%
181 \setbox\tw@\hbox{\unhbox\@ne\unskip\global\setbox\@ne\lastbox}%
182 \setbox\tw@\hbox{$\kern\wd\@ne\kern-\@tempdima\left(\kern-\wd\@ne
183 \global\setbox\@ne\vbox{\box\@ne\kern2\p@}%
184 \vcenter{\kern-\ht\@ne\unvbox\z@\kern-\baselineskip}\,,\right)$}%
185 \null\;\vbox{\kern\ht\@ne\box\tw@}\endgroup}

```

(End definition for `\bordermatrix`.)

`\openup`

```

186 \def\openup{\afterassignment\openup\dimen@}
187 \def\openup{\advance\lineskip\dimen@
188 \advance\baselineskip\dimen@
189 \advance\lineskiplimit\dimen@}

```

(End definition for `\openup`.)

`\displaylines`

```

190 \newif\ifdt@p
191 \def\disply{\global\dt@ptrue\openup\jot\m@th
192 \everycr{\noalign{\ifdt@p \global\dt@pfalse \ifdim\prevdepth>-1000\p@
193 \vskip-\lineskiplimit \vskip\normallineskiplimit \fi
194 \else \penalty\interdisplaylinepenalty \fi}}}
195 \def\@lign{\tabskip\z@skip\everycr{}} % restore inside \disply
196 \def\displaylines#1{\disply \tabskip\z@skip
197 \halign{\hb@xt@\displaywidth{$\@lign\hfil\displaystyle##\hfil$}\crr
198 #1\crr}}

```

(End definition for `\displaylines`.)

`\sp`

`\sb`

```

199 \let\sp=~
200 \let\sb=_

```

(End definition for `\sp` and `\sb`.)

`\tmspace` Originally L^AT_EX only provided a small set of spacing commands for use in text and math, some of the commands like `\;` were only supported in math mode. `amsmath` normalized and provided all of them in text and math. This code has now been moved to the kernel so that it is generally available.

`\thinspace`

`\!`

`\negthinspace`

`\:`

`\medspace`

`\negmedspace`

`\;`

`\thickspace`

`\negthickspace`

```

201 </2ekernel>
202 <*2ekernel | latexrelease>
203 <latexrelease>\IncludeInRelease{2020/10/01}%
204 <latexrelease> \tmspace\amsmath spacing commands}%
205 \DeclareRobustCommand\tmspace[3]{%
206 \ifmmode\mskip#1#2\else\leavevmode@ifvmode\kern#1#3\fi\relax}

```

`\tmspace` is really meant to be an internal command so it doesn't necessarily has to be robust but it was robust in `amsmath` so we leave it like that.

In `amsmath` the text kern is `.1667em`. For compatibility reasons we keep the longer one.

```
207 \DeclareRobustCommand\,{\tmspace+\thinmuskip{.16667em}}
208 \let\thinspace\,
209 \DeclareRobustCommand\!{\tmspace-\thinmuskip{.16667em}}
210 \let\negthinspace\!
211 \DeclareRobustCommand\:{\tmspace+\medmuskip{.2222em}}
212 \let\medspace\:
```

L^AT_EX has a second name for this in its manual:

```
213 \let\>=\:
214 \DeclareRobustCommand\negmedspace{\tmspace-\medmuskip{.2222em}}
215 \DeclareRobustCommand\;{\tmspace+\thickmuskip{.2777em}}
216 \let\thickspace\;
217 \DeclareRobustCommand\negthickspace{\tmspace-\thickmuskip{.2777em}}
218 </2ekernel | latexrelease>
219 <latexrelease>\EndIncludeInRelease
220 <latexrelease>\IncludeInRelease{0000/00/00}%
221 <latexrelease>          {\tmspace}{amsmath spacing commands}%
222 <latexrelease>
223 <latexrelease>\let\tmspace@undefined
224 <latexrelease>\DeclareRobustCommand{\,}{%
225 <latexrelease>    \relax\ifmmode\mskip\thinmuskip\else\thinspace\fi
226 <latexrelease>\DeclareRobustCommand\thinspace{\leavevmode@ifvmode\kern .16667em }
227 <latexrelease>\DeclareRobustCommand\negthinspace{\leavevmode@ifvmode\kern-.16667em }
228 <latexrelease>\def\>{\mskip\medmuskip}
229 <latexrelease>\let\:=\>
230 <latexrelease>\def\;{\mskip\thickmuskip}
231 <latexrelease>\def\!{\mskip-\thinmuskip}
232 <latexrelease>
233 <latexrelease>\let\negmedspace@undefined
234 <latexrelease>\let\negthickspace@undefined
235 <latexrelease>
236 <latexrelease>\EndIncludeInRelease
237 <*2ekernel>
```

(End definition for `\tmspace` and others.)

`*`

```
238 \DeclareRobustCommand\*{\discretionary{\thinspace\the\textfont2\char2}{-}{}}
```

(End definition for ``.)*

`\:` Nickname for the medium space since `\>` is not available inside `tabbing`.

```
239 %\let\:=\>
```

(End definition for `\:`.)

`\active@math@prime` This is the definition of the active math prime.

```
240 \def\active@math@prime{^{\bgroup\prim@s}}
```

(End definition for `\active@math@prime`.)

`\prime@s`

```

241 {\catcode'\='=\active \global\let'\active@math@prime}
242 \def\prim@s{%
243   \prime\futurelet\@let@token\pr@m@s}
244 \def\pr@m@s{%
245   \ifx'\@let@token
246     \expandafter\pr@@@s
247   \else
248     \ifx^\@let@token
249       \expandafter\expandafter\expandafter\pr@@@t
250     \else
251       \egroup
252     \fi
253   \fi}
254 \def\pr@@@s#1{\prim@s}
255 \def\pr@@@t#1#2{#2\egroup}

(End definition for \prime@s.)

256 {\catcode'\_=\active \gdef_{\_}} % _ in math is
257                                     % either subscript or \_

```

1.2 Math Environments

`\(` Produces \dots with checks that `\(` isn't used in math mode, and that `\)` is only used in math mode begun with `\(`.

```

258 \</2ekernel>
259 \<latexrelease>\IncludeInRelease{2015/01/01}{\{()\}{Make \< robust}}%
260 \<*2ekernel | latexrelease>
261 \DeclareRobustCommand\<{\%
262   \relax\ifmmode\@badmath\else$\fi}%
263 \DeclareRobustCommand\<{\%
264   \relax\ifmmode\ifinner$\else\@badmath\fi\else \@badmath\fi}%
265 \</2ekernel | latexrelease>
266 \<latexrelease>\EndIncludeInRelease
267 \<latexrelease>\IncludeInRelease{0000/00/00}{\{()\}{Make \< robust}}%
268 \<latexrelease>\def\<{\%
269 \<latexrelease> \relax\ifmmode\@badmath\else$\fi}%
270 \<latexrelease>\expandafter\let\csname\string( \endcsname\@undefined
271 \<latexrelease>\def\<{\%
272 \<latexrelease> \relax\ifmmode\ifinner$\else\@badmath\fi\else \@badmath\fi}%
273 \<latexrelease>\expandafter\let\csname\string) \endcsname\@undefined
274 \<latexrelease>\EndIncludeInRelease
275 \<*2ekernel>

```

(End definition for `\(` and `\)`.)

`\[` Produces \dots with checks that `\[` isn't used in math mode, and that `\]` is only used in display math mode (though there is no real test that this display math started with `\[` and not with \dots).

```

276 \</2ekernel>
277 \<latexrelease>\IncludeInRelease{2015/01/01}{\{[\]}{Make \[ robust}}%

```

```

278 <*2ekernel | latexrelease>
279 \DeclareRobustCommand\[%
280   \relax\ifmmode
281     \@badmath
282   \else
283     \ifvmode
284       \nointerlineskip
285       \makebox[.6\linewidth]{}%
286     \fi
287     $$$%$ BRACE MATCH HACK
288   \fi
289 }%

290 \DeclareRobustCommand\[%
291   \relax\ifmmode
292     \ifinner
293       \@badmath
294     \else
295       $$$%$ BRACE MATCH HACK
296     \fi
297   \else
298     \@badmath
299   \fi
300   \ignorespaces
301 }%

302 </2ekernel | latexrelease>
303 <latexrelease>\EndIncludeInRelease
304 <latexrelease>\IncludeInRelease{0000/00/00}{\[]{-Make \[ robust}}%
305 <latexrelease>\def\[%
306 <latexrelease>   \relax\ifmmode
307 <latexrelease>     \@badmath
308 <latexrelease>   \else
309 <latexrelease>     \ifvmode
310 <latexrelease>       \nointerlineskip
311 <latexrelease>       \makebox[.6\linewidth]{}%
312 <latexrelease>     \fi
313 <latexrelease>     $$$%$ BRACE MATCH HACK
314 <latexrelease>   \fi
315 <latexrelease>}%
316 <latexrelease>\expandafter\let\csname\string[ \endcsname\@undefined

317 <latexrelease>\def\[%
318 <latexrelease>   \relax\ifmmode
319 <latexrelease>     \ifinner
320 <latexrelease>       \@badmath
321 <latexrelease>     \else
322 <latexrelease>       $$$%$ BRACE MATCH HACK
323 <latexrelease>     \fi
324 <latexrelease>   \else
325 <latexrelease>     \@badmath
326 <latexrelease>   \fi
327 <latexrelease>   \ignorespaces
328 <latexrelease>}%
329 <latexrelease>\expandafter\let\csname\string[ \endcsname\@undefined
330 <latexrelease>\EndIncludeInRelease

```

```

331 \*2ekernel)

(End definition for \[ and \].)

math Disguises for \(\dots\) and \[...\].
displaymath
332 \let\math=\(
333 \let\endmath=\)

334 \def\displaymath{\[}
335 \def\enddisplaymath{\]\@ignoretrue}

\c@equation Numbered equations, using the counter \c@equation. Note: The document style must
define \theequation etc., and do the appropriate \@addtoreset. It should also redefine
\@eqnnum if another format for the equation number is desired other than the standard
(...), or to move the equation numbers to the flushleft. (See comment on the \def of
\@eqnnum.)
336 \@definecounter{equation}
337 \def\equation{$$\refstepcounter{equation}}
338 \def\endequation{\eqno \hbox{\@eqnnum}$$\@ignoretrue}
(End definition for \c@equation.)

\@eqnnum Produces the equation number for equation and eqnarray environments. The following
definition is for flushright numbers; for flushleft numbers, see leqno.clo. The equation
number is set in black roman type even if an eqnarray environment appears in an italic
environment.
339 \def\@eqnnum{\normalfont \normalcolor (\theequation)}
(End definition for \@eqnnum.)

\stackrel A disguise for plain TeX's \stackrel.
340 \DeclareRobustCommand\stackrel[2]{\mathrel{\mathop{#2}\limits^{#1}}}
(End definition for \stackrel.)

\frac A disguise for plain TeX's \over.
341 \DeclareRobustCommand\frac[2]{\begingroup#1\endgroup\over#2}
(End definition for \frac.)

\sqrt Add an optional argument to plain's \sqrt to give the  $n$ th root of an expression  $\sqrt[n]{e}$ .
\@sqrt
342 \DeclareRobustCommand\sqrt{\@ifnextchar[\@sqrt\sqrtsign}
343 \def\@sqrt[#1]{\root #1\of}
(End definition for \sqrt and \@sqrt.)

\eqnarray Here's the eqnarray environment: Default is for left-hand side of equations to be
flushright. To make them flushleft, \let\@eqnrel = \hfil.
\@eqnrel
\@eqnsw
\if@eqnsw 344 \newcount\@eqcnt
\@eqnrel 345 \newcount\@eqpen
346 \newif\if@eqnsw\@eqnswtrue
347 \newskip\@centering
348 \@centering = 0pt plus 1000pt

```

To get a proper `\@currentlabel` we have to redefine it for the whole display. Note that we can't use `\refstepcounter` as this results in `\@currentlabel` getting restored at the wrong and thus always writing the first label to the `.aux` file.

```

349 \def\eqnarray{%
350     \stepcounter{equation}%
351     \def\@currentlabel{\p@equation\theequation}%
352     \global\@eqnswtrue
353     \m@th
354     \global\@eqcnt\z@
355     \tabskip\@centering
356     \let\\\@eqnocr
357     $$\everycr{}\halign to\displaywidth\bgroup
358         \hskip\@centering$\displaystyle\tabskip\z@skip{##}$\@eqnscr
359         &\global\@eqcnt\@ne\hskip \tw@\arraycolsep \hfil${##}$\hfil
360         &\global\@eqcnt\tw@ \hskip \tw@\arraycolsep
361         $\displaystyle{##}$\hfil\tabskip\@centering
362         &\global\@eqcnt\thr@@ \hb@xt@\z@\bgroup\hss##\egroup
363         \tabskip\z@skip
364     \cr
365 }

366 \def\endeqnarray{%
367     \@eqnocr
368     \egroup
369     \global\advance\c@equation\m@ne
370     $$\ignoretrue
371 }

372 \let\@eqnscr=\relax
(End definition for \@eqcnt and others.)

```

`\nonumber` Switches off equation numbering.

```

373 \def\nonumber{\global\@eqnswfalse}

(End definition for \nonumber.)

```

```

\@eqnocr
\@xeqnocr
\@yeqnocr
374 \protected\def\@eqnocr{%
375     {\ifnum0='}\fi
376     \@ifstar{%
377         \global\@eqpen\@M\@yeqnocr
378     }{%
379         \global\@eqpen\interdisplaylinepenalty \@yeqnocr
380     }%
381 }

382 \def\@yeqnocr{\@testopt\@xeqnocr\z@skip}

383 </2ekernel>
384 <*2ekernel | latexrelease>
385 <latexrelease>\IncludeInRelease{2020/10/01}%
386 <latexrelease>          {\@xeqnocr}{eqnarray support calc syntax}%
387 \def\@xeqnocr[#1]{%
388     \ifnum0='{ \fi}%
389     \@eqnocr

```

```

390 \noalign{\penalty\@eqpen\vskip\jot\@vspace@calcify{#1}}%
391 }
392 </2ekernel | latexrelease>
393 <latexrelease>\EndIncludeInRelease
394 <latexrelease>\IncludeInRelease{0000/00/00}%
395 <latexrelease> \{\@eqnocr\}{eqnarray support calc syntax}%
396 <latexrelease>
397 <latexrelease>\def\@eqnocr[#1]{%
398 <latexrelease> \ifnum0='{ \fi}%
399 <latexrelease> \@@eqnocr
400 <latexrelease> \noalign{\penalty\@eqpen\vskip\jot\vskip #1\relax}%
401 <latexrelease>}
402 <latexrelease>\EndIncludeInRelease
403 <*2ekernel>

```

(End definition for \@eqnocr, \@xeqnocr, and \@yeqnocr.)

\@@eqnocr

```

404 \def\@@eqnocr{\let\reserved@a\relax
405 \ifcase\@eqcnt \def\reserved@a{& &}\or \def\reserved@a{& &}%
406 \or \def\reserved@a{&}\else
407 \let\reserved@a\empty
408 \latexerror{Too many columns in eqnarray environment}\@ehc\fi
409 \reserved@a \if@eqnsw\@eqnnum\stepcounter{equation}\fi
410 \global\@eqnswtrue\global\@eqcnt\z@\cr}

```

(End definition for \@@eqnocr.)

~~eqnarray*~~ Here's the eqnarray* environment:

```

411 \let\@seqnocr=\@eqnocr
412 \namedef{eqnarray*}{\protected\def\@eqnocr{\nonumber\@seqnocr}\eqnarray}
413 \namedef{endeqnarray*}{\nonumber\endeqnarray}

```

(End definition for \@seqnocr.)

\lefteqn \lefteqn{FORMULA} typesets FORMULA in display math style flushleft in a box of width zero.

```

414 \def\lefteqn#1{\rlap{$\displaystyle #1$}}

```

(End definition for \lefteqn.)

\ensuremath In math mode, \ensuremath{text} is equivalent to text; in LR or paragraph mode, it is equivalent to \$text\$. \relax is not needed in front of the \ifmmode as \protect will be \let to \relax. This version (due to Donald Arseneau) avoids duplicating its argument in the 'then' and 'else' part of the \ifmath which is necessary in nested 'tabular' like environments. See amslatex/2104.

```

415 \DeclareRobustCommand{\ensuremath}{%
416 \ifmmode
417 \expandafter\@firstofone
418 \else
419 \expandafter\@ensuredmath
420 \fi}

```

(End definition for \ensuremath.)

`\@ensuredmath` The `\relax` stops `\ensuremath{}` starting display math.

```
421 \long\def\@ensuredmath#1{\relax#1$}
```

(End definition for `\@ensuredmath`.)

```
422 \endkernel
```

1.3 External options to the standard document classes

1.3.1 Left equation numbering

`\@eqnnum` To put the equation number on the left side of an equation we have to use a little trick. The number is shifted `\displaywidth` to the left inside a box of (approximately) zero width. This fails when the equation is too wide, the equation number then may overprint the equation itself.

```
423 \def\@eqnnum{\relax\@ensuredmath{\hbox{\normalfont\normalcolor
424 \rlap{\hspace{-\displaywidth}\hbox{\normalfont\normalcolor
425 \normalfont\normalcolor\theequation}}}}
426 \enddef
427 \enddef
```

(End definition for `\@eqnnum`.)

1.3.2 Flush left equations

To get the displayed math environments to print the contents flush left (with an indentation) we have to redefine all of L^AT_EX 2_ε's displayed math environments.

`\mathindent` The amount of indentation of the equations is stored in a register.

```
428 \def\mathindent{\newskip\mathindent}
429 \newskip\mathindent
```

The setting of `\mathindent` has to be deferred until the class file has been processed, because `\leftmargini` is still 0pt wide at the moment `fileqn.clo` is read in.

```
430 \AtEndOfClass{\mathindent\leftmargini}
```

(End definition for `\mathindent`.)

`\[` Begin display math;

```
431 \IncludeInRelease{2015/01/01}{\[}{\Make \[ robust}%
432 \DeclareRobustCommand\[\relax
433 \ifmmode\@badmath
434 \else
435 \begin{trivlist}%
436 \beginparpenalty\predisplaypenalty
437 \endparpenalty\postdisplaypenalty
438 \item[]\leavevmode
439 \hbox{\linewidth\group $m@th\displaystyle %$
440 \hspace{\mathindent}\group}
441 \fi}
442 \EndIncludeInRelease
```

```

443 \IncludeInRelease{0000/00/00}{\[]}{Make \[ robust}%
444 \renewcommand\[\relax
445     \ifmmode\@badmath
446     \else
447     \begin{trivlist}%
448     \@beginparpenalty\predisplaypenalty
449     \@endparpenalty\postdisplaypenalty
450     \item[]\leavevmode
451     \hb@xt@\linewidth\bgroup $\m@th\displaystyle %$
452     \hskip\mathindent\bgroup
453     \fi}
454 \EndIncludeInRelease

```

(End definition for \[.)

\] end display math;

```

455 \IncludeInRelease{2015/01/01}{\[]}{Make \] robust}%
456 \DeclareRobustCommand\]\relax
457     \ifmmode
458     \egroup $\hfil% $
459     \egroup
460     \end{trivlist}%
461     \else \@badmath
462     \fi}
463 \EndIncludeInRelease

```

```

464 \IncludeInRelease{0000/00/00}{\[]}{Make \] robust}%
465 \renewcommand\]\relax
466     \ifmmode
467     \egroup $\hfil% $
468     \egroup
469     \end{trivlist}%
470     \else \@badmath
471     \fi}
472 \EndIncludeInRelease

```

(End definition for \].)

equation The equation environment

```

473 \renewenvironment{equation}%
474     {\@beginparpenalty\predisplaypenalty
475     \@endparpenalty\postdisplaypenalty
476     \refstepcounter{equation}%
477     \trivlist \item[]\leavevmode
478     \hb@xt@\linewidth\bgroup $\m@th% $
479     \displaystyle
480     \hskip\mathindent}%

```

Ensure that there is at least a space between formula and equation number so that they don't bump in each other.

```

481     {$\hskip .3em minus.3em\hfil % $
482     \displaywidth\linewidth\hbox{\@eqnnum}%
483     \egroup
484     \endtrivlist}

```

eqnarray The eqnarray environment

```

485 \renewenvironment{eqnarray}{%
486   \stepcounter{equation}%
487   \def\@currentlabel{\p@equation\theequation}%
488   \global\@eqnswtrue\m@th
489   \global\@eqcnt\z@
490   \tabskip\mathindent
491   \let\=\@eqnocr
492   \setlength\abovedisplayskip{\topsep}%
493   \ifvmode
494     \addtolength\abovedisplayskip{\partopsep}%
495   \fi

```

When the documentclass uses a non-zero \parskip setting the \topsep might have a negative value to compensate for that. Therefore we add \parskip to \abovedisplayskip.

```

496   \addtolength\abovedisplayskip{\parskip}%
497   \setlength\belowdisplayskip{\abovedisplayskip}%
498   \setlength\belowdisplayshortskip{\abovedisplayskip}%
499   \setlength\abovedisplayshortskip{\abovedisplayskip}%
500   $$\everycr{}\halign to\linewidth% $$
501   \bgroup
502     \hskip\@centering
503     $\displaystyle\tabskip\z@skip{##}$\@eqnsele%
504     \global\@eqcnt\@ne \hskip \tw@arraycolsep \hfil${##}$\hfil%
505     \global\@eqcnt\tw@ \hskip \tw@arraycolsep
506     $\displaystyle{##}$\hfil \tabskip\@centering%
507     \global\@eqcnt\thr@@
508     \hb@xt@\z@\bgroup\hss##\egroup\tabskip\z@skip\cr}%
509     {\@eqnocr
510   \egroup
511   \global\advance\c@equation\m@ne$$$ $
512   \@ignoretrue
513 }
514 \fleqn

```

File I

ltlists.dtx

1 List, and related environments

The generic commands for creating an indented environment – `enumerate`, `itemize`, `quote`, etc – are:

```
\list{<LABEL>}{<COMMANDS>} ... \endlist
```

which can be invoked by the user as the list environment. The LABEL argument specifies item labeling. COMMANDS contains commands for changing the horizontal and vertical spacing parameters.

Each item of the environment is begun by the command `\item[ITEMLABEL]` which produces an item labeled by ITEMLABEL. If the argument is missing, then the LABEL argument of the `\list` command is used as the item label.

The label is formed by putting `\makelabel{<ITEMLABEL>}` in an hbox whose width is either its natural width or else `\labelwidth`, whichever is larger. The `\list` command defines `\makelabel` to have the default definition:

```
\makelabel{<ARG>} == BEGIN \hfil ARG END
```

which, for a label of width less than `\labelwidth`, puts the label flushright, `\labelsep` to the left of the item's text. However, `\makelabel` can be `\let` to another command by the `\list`'s COMMANDS argument.

A `\usecounter{<foo>}` command in the second argument causes the counter *foo* to be initialized to zero, and stepped by every `\item` command without an argument. (`\label` commands within the list refer to this counter.)

When you leave a list environment, returning either to an enclosing list or normal text mode, LaTeX begins a new paragraph if and only if you leave a blank line after the `\end` command. This is accomplished by the `\@endparenv` command.

Blank lines are ignored every other reasonable place–i.e.:

- Between the `\begin{list}` and the first `\item`,
- Between the `\item` and the text of that item.
- Between the end of the last item and the `\end{list}`.

For an environment like quotation, in which items are not labeled, the entire environment is a single item. It is defined by letting `\quotation == \list{}{...}\item\relax`. (Note the `\relax`, there in case the first character in the environment is a '['.) The spacing parameters provide a great deal of flexibility in designing the format, including the ability to let the indentation of the first paragraph be different from that of the subsequent ones.

The trivlist environment is equivalent to a list environment whose second argument sets the following parameter values:

`\leftmargin = 0`: causes no indentation of left margin

`\labelwidth = 0`: see below for precise effect this has.

`\itemindent = 0`: with a null label, makes first paragraph have no indentation. Succeeding paragraphs have `\parindent` indentation. To give first paragraph same indentation, set `\itemindent = \parindent` before the `\item[]`.

Every `\item` in a `trivlist` environment must have an argument—in many cases, this will be the null argument (`\item[]`). The `trivlist` environment is mainly used for paragraphing environments, like `verbatim`, in which there is no margin change. It provides the same vertical spacing as the `list` environment, and works reasonably well when it occurs immediately after an `\item` command in an enclosing list.

1.1 List and Trivlist

The following variables are used inside a list environment:

`\totalleftmargin` The distance that the prevailing left margin is indented from the outermost left margin,

`\linewidth` The width of the current line. Must be initialized to `\hsize`.

`\listdepth` A count for holding current list nesting depth.

`\makelabel` A macro with a single argument, used to generate the label from the argument (given or implied) of the `\item` command. Initialized to `\mklab` by the `\list` command. This command must produce some stretch—i.e., an `\hfil`.

`\inlabel` A switch that is false except between the time an `\item` is encountered and the time that `TeX` actually enters horizontal mode. Should be tested by commands that can be messed up by the list environment's use of `\everypar`.

`\box\@labels` When `\inlabel = true`, it holds the labels to be put out by `\everypar`.

`\noparitem` A switch set by `\list` when `\inlabel = true`. Handles the case of a `\list` being the first thing in an item.

`\noparlist` A switch set true for a list that begins an item. No `\topsep` space is added before or after `\item`'s such a list.

`\newlist` Set true by `\list`, set false by the first text (by `\everypar`).

`\noitemarg` Set true when executing an `\item` with no explicit argument. Used to save space. To save time, make two separate `\@item` commands.

`\nmbrlist` Set true by `\usecounter` command, causes list to be numbered.

`\listctr` `\def`'ed by `\usecounter` to name of counter.

`\noskipsec` A switch set true by a sectioning command when it is creating an in-text heading with `\everypar`.

Throughout a list environment, `\hsize` is the width of the current line, measured from the outermost left margin to the outermost right margin. Environments like `tabbing` should use `\linewidth` instead of `\hsize`.

Here are the parameters of a list that can be set by commands in the `\list's` COMMANDS argument. These parameters are all `TeX` skips or dimensions (defined by `\newskip` or `\newdimen`), so the usual `TeX` or `LATeX` commands can be used to set them. The commands will be executed in `vmode` if and only if the `\list` was preceded by a `\par` (or something like an `\end{list}`), so the spacing parameters can be set according to whether the list is inside a paragraph or is its own paragraph.

1.2 Vertical Spacing (skips)

`\topsep`: Space between first item and preceding paragraph.

`\partopsep`: Extra space added to `\topsep` when environment starts a new paragraph (is called in `vmode`).

`\itemsep`: Space between successive items.

`\parsep`: Space between paragraphs within an item – the `\parskip` for this environment.

1.3 Penalties

`\@beginparpenalty`: put at the beginning of a list

`\@endparpenalty`: put at end of list

`\@itempenalty`: put between items.

1.4 Horizontal Spacing (dimens)

`\leftmargin`: space between left margin of enclosing environment (or of page if top level list) and left margin of this list. Must be nonnegative.

`\rightmargin`: analogous.

`\listparindent`: extra indentation at beginning of every paragraph of a list except the one started by the `\item` command. May be negative! Usually, labeled lists have `\listparindent` equal to zero.

`\itemindent`: extra indentation added right BEFORE an item label.

`\labelwidth`: nominal width of box that contains the label. If the natural width of the label \leq `\labelwidth`, then the label is flushed right inside a box of width `\labelwidth` (with an `\hfil`). Otherwise, a box of the natural width is employed, which causes an indentation of the text on that line.

`\labelsep`: space between end of label box and text of first item.

1.5 Default Values

Defaults for the list environment are set as follows. First, `\rightmargin`, `\listparindent` and `\itemindent` are set to 0pt. Then, one of the commands `\@listi`, `\@listii`, ... , `\@listvi` is called, depending upon the current level of the list. The `\@list ...` commands should be defined by the document style. A convention that the document style should follow is to set `\leftmargin` to `\leftmargini`, ..., `\leftmarginvi` for the appropriate level. Items that aren't changed may be left alone, but everything that could possibly be changed must be reset. *Historical L^AT_EX 2.09 comments (not necessarily accurate any more)*:

```
\list{LABEL}{COMMANDS} ==  
  BEGIN  
    if \@listdepth > 5  
      then LaTeX error: 'Too deeply nested'  
      else \@listdepth :=G \@listdepth + 1
```

```

fi
\rightmargin      := 0pt
\listparindent    := 0pt
\itemindent       := 0pt
\eval{@list \romannumeral\the\@listdepth} %% Set default values:
\itemlabel        :=L LABEL
\makelabel        == \@mklab
@nmbrlist         :=L false
COMMANDS

\@trivlist          % commands common to \list and \trivlist

\parskip           :=L \parsep
\parindent         :=L \listparindent
\linewidth         :=L \linewidth - \rightmargin - \leftmargin
\@totalleftmargin :=L \@totalleftmargin + \leftmargin
\parshape 1 \@totalleftmargin \linewidth
\ignorespaces      % gobble space up to \item
END

\endlist == BEGIN \@listdepth :=G \@listdepth -1
              \endtrivlist
              END

\@trivlist ==
BEGIN
  if @newlist = T then \@noitemerr fi
                      %% This command removed for some forgotten reason.
  \@topsepadd :=L \topsep
  if @noskipsec then leave vertical mode fi %% Added 11 Jun 85
  if vertical mode
  then \@topsepadd :=L \@topsepadd + \partopsep
  else \unskip \par % remove glue from end of last line
  fi
  if @inlabel = true
  then @noparitem :=L true
       @noparlist :=L true
  else @noparlist :=L false
       \@topsep    :=L \@topsepadd
  fi
  \@topsep        :=L \@topsep + \parskip %% Change 4 Sep 85
  \leftskip       :=L 0pt % Restore paragraphing parameters
  \rightskip      :=L \@rightskip
  \parfillskip    :=L 0pt + 1fil

NOTE: \@setpar called on every \list in case \par has been
temporarily munged before the \list command.
\@setpar{if @newlist = false then {\@@par} fi}
\@newlist      :=G T
\@outerparskip :=L \parskip

```

```

END

\trivlist ==
BEGIN
  \parsep      := \parskip
  @nmbrlist := F
  \@trivlist
  \labelwidth := 0
  \leftmargin := 0
  \itemindent := \parindent
  \@itemlabel :=L "empty"      %% added 93/12/13
  \makelabel{LABEL} == LABEL
END

\endtrivlist ==
BEGIN
  if @inlabel = T then \indent fi
  if horizontal mode then \unskip \par fi
  if @noparlist = true
    else if \lastskip > 0
      then \@tempkipa := \lastskip
           \vskip - \lastskip
           \vskip \@tempkipa -\@outerparskip + \parskip
      fi
    \endparenv
  fi
END

\@endparenv ==
BEGIN
  \addpenalty{@endparpenalty}
  \addvspace{@topsepadd}
  \endgroup    %% ends the \begin command's \begingroup
  \par == BEGIN
           \@restorepar
           \everypar{}
           \par
        END
  \everypar == BEGIN remove \lastbox \everypar{} END
  \begingroup %% to match the \end commands \endgroup
END

\item == BEGIN if math mode then WARNING fi
           if next char = [
             then \@item
             else @noitemarg := true
                  \@item[@itemlabel]
           fi
        END

\@item[LAB] ==

```



```

BEGIN
if @noperitem = true
then @noperitem := false
    % NOTE: then clause hardly every taken,
    % so made a macro \@donoperitem
    \box\@labels :=G \hbox{\hskip -\leftmargin
                        \box\@labels
                        \hskip \leftmargin }

if @minipage = false then
    \@tempskipa := \lastskip
    \vskip -\lastskip
    \vskip \@tempskipa + \@outerparskip - \parskip
fi
else if @inlabel = true
then \indent \par % previous item empty.
fi
if hmode then 2 \unskip's
    % To remove any space at end of prev.
    % paragraph that could cause a blank line.
    \par
fi
if @newlist = T
then if @nobreak = T % Kludge if list follows \section
then \addvspace{\@outerparskip - \parskip}
else \addpenalty{\@beginparpenalty}
    \addvspace{\@topsep}
    \addvspace{-\parskip} %% added 4 Sep 85
fi
else \addpenalty{\@itempenalty}
    \addvspace{\itemsep}
fi
@inlabel :=G true
fi

\everypar{ @minipage :=G F
@newlist :=G F
if @inlabel = true
then @inlabel :=G false
    \hskip -\parindent
    \box\@labels
    \penalty 0
    %% 3 Oct 85 - allow line break here
    \box\@labels :=G null
fi
\everypar{} }

@nobreak :=G false
if @noitemarg = true
then @noitemarg := false
if @nmbrlist
then \refstepcounter{\@listctr}

```

```

fi      fi
\@tempboxa :=L \hbox{\makelabel{LAB}}
\box\@labels :=G \@labels \hskip \itemindent
               \hskip - (\labelwidth + \labelsep)
               if \wd \@tempboxa > \labelwidth
                   then \box\@tempboxa
                   else \hbox to \labelwidth {\unhbox\@tempboxa}
               fi
               \hskip\labelsep
\ignorespaces %gobble space up to text
END

```

```

\makelabel{LABEL} == ERROR %% default to catch lonely \item

```

```

\usecounter{CTR} == BEGIN @nmbrlist :=L true
                        \@listctr == CTR
                        \setcounter{CTR}{0}
                        END

```

DEFINE \dimen's and \count
End of historical L^AT_EX 2.09 comments.

```

\topskip
\partopsep 1 \*2kernel)
\itemsep    2 \newskip\topsep
\parsep     3 \newskip\partopsep
\@topsep    4 \newskip\itemsep
\@topsepadd 5 \newskip\parsep
\outerparskip 6 \newskip\@topsep
              7 \newskip\@topsepadd
              8 \newskip\@outerparskip

```

(End definition for \topskip and others.)

```

\leftmargin
\rightmargin 9 \newdimen\leftmargin
\listparindent 10 \newdimen\rightmargin
\itemindent   11 \newdimen\listparindent
\labelwidth   12 \newdimen\itemindent
\labelsep     13 \newdimen\labelwidth
\@totalleftmargin 14 \newdimen\labelsep
               15 \newdimen\linewidth
               16 \newdimen\@totalleftmargin \@totalleftmargin=\z@

```

(End definition for \leftmargin and others.)

```

\leftmargini
\leftmarginii 17 \newdimen\leftmargini
\leftmarginiii 18 \newdimen\leftmarginii
\leftmarginiv 19 \newdimen\leftmarginiii
\leftmarginv 20 \newdimen\leftmarginiv
\leftmarginvi 21 \newdimen\leftmarginv
                22 \newdimen\leftmarginvi

                (End definition for \leftmargini and others.)

\@listdepth
\@itempenalty 23 \newcount\@listdepth \@listdepth=0
\@beginparpenalty 24 \newcount\@itempenalty
\@endparpenalty 25 \newcount\@beginparpenalty
                26 \newcount\@endparpenalty

                (End definition for \@listdepth and others.)

\@labels
                27 \newbox\@labels

                (End definition for \@labels.)

\if@inlabel
\@inlabelfalse 28 \newif\if@inlabel \@inlabelfalse
\@inlabeltrue
                (End definition for \if@inlabel, \@inlabelfalse, and \@inlabeltrue.)

\if@newlist
\@newlistfalse 29 \newif\if@newlist \@newlistfalse
\@newlisttrue
                (End definition for \if@newlist, \@newlistfalse, and \@newlisttrue.)

\if@nparitem
\@nparitemfalse 30 \newif\if@nparitem \@nparitemfalse
\@nparitemtrue
                (End definition for \if@nparitem, \@nparitemfalse, and \@nparitemtrue.)

\if@nparlist
\@nparlistfalse 31 \newif\if@nparlist \@nparlistfalse
\@nparlisttrue
                (End definition for \if@nparlist, \@nparlistfalse, and \@nparlisttrue.)

\if@noitemarg
\@noitemargfalse 32 \newif\if@noitemarg \@noitemargfalse
\@noitemargtrue
                (End definition for \if@noitemarg, \@noitemargfalse, and \@noitemargtrue.)

\if@newlist
\@newlistfalse 33 \newif\if@nmbolist \@nmbolistfalse
\@newlisttrue
                (End definition for \if@newlist, \@newlistfalse, and \@newlisttrue.)

```

`\list`

```

34 \def\list#1#2{%
35   \ifnum \@listdepth >5\relax
36     \@toodeep
37   \else
38     \global\advance\@listdepth\@ne
39   \fi
40   \rightmargin\z@
41   \listparindent\z@
42   \itemindent\z@
43   \csname @list\romannumeral\the\@listdepth\endcsname
44   \def\@itemlabel{#1}%
45   \let\makelabel\@mklab
46   \@nmbrlistfalse
47   #2\relax
48   \@trivlist
49   \parskip\parsep
50   \parindent\listparindent
51   \advance\linewidth -\rightmargin
52   \advance\linewidth -\leftmargin
53   \advance\@totalleftmargin \leftmargin
54   \parshape \@ne \@totalleftmargin \linewidth
55   \ignorespaces}

```

(End definition for \list.)

`\par@deathcycles`

```

56 \newcount\par@deathcycles

```

(End definition for \par@deathcycles.)

`\@trivlist` Because `\par` is sometimes made a no-op it is possible for a missing `\item` to produce a loop that does not fill memory and so never gets trapped by T_EX. We thus need to trap this here by setting `\par` to count the number of times a paragraph is called with no progress being made started.

```

57 \def\@trivlist{%
58   \if@noskipsec \leavevmode \fi
59   \@topsepadd \topsep
60   \ifvmode
61     \advance\@topsepadd \partopsep
62   \else
63     \unskip \par
64   \fi
65   \if@inlabel
66     \@nparitemtrue
67     \@nparlisttrue
68   \else
69     \if@newlist \@noitemerr \fi
70     \@nparlistfalse
71     \@topsep \@topsepadd
72   \fi
73   \advance\@topsep \parskip
74   \leftskip \z@skip
75   \rightskip \@rightskip
76   \parfillskip \@flushglue

```

```

77 \par@deathcycles \z@
78 \@setpar{\if@newlist
79         \advance\par@deathcycles \@ne
80         \ifnum \par@deathcycles >\@m
81         \@noitemerr
82         {\@@par}%
83         \fi
84         \else
85         {\@@par}%
86         \fi}%
87 \global \@newlisttrue
88 \@outerparskip \parskip}

```

(End definition for \@trivlist.)

\trivlist

```

89 \def\trivlist{%
90 \parsep\parskip
91 \@nmbrlistfalse
92 \@trivlist
93 \labelwidth\z@
94 \leftmargin\z@
95 \itemindent\z@

```

We initialise \@itemlabel so that a trivlist with an \item not having an optional argument doesn't produce an error message.

```

96 \let\@itemlabel\@empty
97 \def\makelabel##1{##1}}

```

(End definition for \trivlist.)

\endlist

```

98 \def\endlist{%
99 \global\advance\@listdepth\m@ne
100 \endtrivlist}

```

(End definition for \endlist.)

The definition of \trivlist used to be in ltspc.dtx so that other commands could be 'let to it'. They now use \def.

\endtrivlist

```

101 \def\endtrivlist{%
102 \if@inlabel
103 \leavevmode
104 \global \@inlabelfalse
105 \fi
106 \if@newlist
107 \@noitemerr
108 \global \@newlistfalse
109 \fi
110 \ifhmode\unskip \par

```

We also check if we are in math mode and issue an error message if so (hoping that \@currenvir resolves suitably). Otherwise the usual "perhaps a missing item" error will get triggered later which is confusing.

```

111 \else

```

```

112     \inmatherr{\end{\@currenvir}}}%
113   \fi
114   \if@nolist \else
115     \ifdim\lastskip >\z@
116       \@tempskipa\lastskip \vskip -\lastskip
117       \advance\@tempskipa\parskip \advance\@tempskipa -\@outerparskip
118       \vskip\@tempskipa
119     \fi
120   \endparenv
121 \fi
122 }

```

(End definition for `\endtrivlist`.)

`\@endparenv` To suppress the paragraph indentation in text immediately following a paragraph-making environment, `\everypar` is changed to remove the space, and `\par` is redefined to restore `\everypar`. Instead of redefining `\par` and `\everypar`, `\@endparenv` was changed to set the `@endpe` switch, letting `\end` redefine `\par` and `\everypar`.

This allows paragraph-making environments to work right when called by other environments. (Changed 27 Oct 86)

```

123 \def\@endparenv{%
124   \addpenalty\@endparpenalty\addvspace\@topsepadd\@endpetrue}
125 \langle latexrelease \rangle \IncludeInRelease{2015/01/01}{\@doendpe}{clubpenalty fix}%
126 \def\@doendpe{\@endpetrue
127   \def\par{\@restorepar

```

If a section heading changes `\clubpenalty` to keep lines after it together then this modification is restored via the `\everypar` mechanism at the start of the next paragraph. As we destroy the contents of this token here we explicitly set `\clubpenalty` back to its default.

```

128       \clubpenalty\@clubpenalty
129   \everypar{\par\@endpefalse}\everypar

```

Use `\setbox0=\lastbox` instead of `\hskip -\parindent` so that a `\noindent` becomes a no-op when used before a line immediately following a list environment (23 Oct 86).

```

130       {\setbox\z@\lastbox}%
131   \everypar{\@endpefalse}}
132 \langle latexrelease \rangle \EndIncludeInRelease
133 \langle latexrelease \rangle \IncludeInRelease{0000/00/00}{\@doendpe}{clubpenalty fix}%
134 \langle latexrelease \rangle \def\@doendpe{\@endpetrue
135 \langle latexrelease \rangle   \def\par{\@restorepar\everypar{\par\@endpefalse}\everypar
136 \langle latexrelease \rangle     {\setbox\z@\lastbox}\everypar{\@endpefalse}}
137 \langle latexrelease \rangle \EndIncludeInRelease

```

(End definition for `\@endparenv` and `\@doendpe`.)

```

\if@endpe
\@endpefalse
\@endpetrue
138 \newif\if@endpe
139 \endpefalse

```

(End definition for `\if@endpe`, `\@endpefalse`, and `\@endpetrue`.)

```

\@mklab
140 \def\@mklab#1{\hfil #1}
(End definition for \@mklab.)

\item
141 \def\item{%
142   \@inmatherr\item
143   \@ifnextchar [\@item{\@noitemargtrue \@item[\@itemlabel]}}
(End definition for \item.)

\@donoparitem
144 \def\@donoparitem{%
145   \@noparitemfalse
146   \global\setbox\@labels\hbox{\hskip -\leftmargin
147                                   \unhbox\@labels
148                                   \hskip \leftmargin}%
149   \if@minipage\else
150     \@tempskipa\lastskip
151     \vskip -\lastskip
152     \advance\@tempskipa\@outerparskip
153     \advance\@tempskipa -\parskip
154     \vskip\@tempskipa
155   \fi}
(End definition for \@donoparitem.)

\@item
156 \def\@item[#1]{%
157   \if@noparitem
158     \@donoparitem
159   \else
160     \if@inlabel
161       \indent \par
162     \fi
163     \ifhmode
164       \unskip\unskip \par
165     \fi
166     \if@newlist
167       \if@nobreak
168         \@nbitem
169       \else
170         \addpenalty\@beginparpenalty
171         \addvspace\@topsep
172         \addvspace{-\parskip}%
173       \fi
174     \else
175       \addpenalty\@itempenalty
176       \addvspace\itemsep
177     \fi
178     \global\@inlabeltrue
179   \fi
180   \everypar{%
181     \@minipagefalse
182     \global\@newlistfalse

```

This `\if@inlabel` check is needed in case an item starts of inside a group so that `\everypar` does not become empty outside that group.

```
183     \if@inlabel
184     \global\@inlabelfalse
```

The paragraph indent is now removed by using `\setbox...` since this makes `\noindent` a no-op here, as it should be. Thus the following comment is redundant but is left here for the sake of future historians: this next command was changed from an `hskip` to a `kern` to avoid a break point after the `parindent` box: the skip could cause a line-break if a very long label occurs in `raggedright` setting. If `\noindent` was used after `\item` want to cancel the `\itemindent` skip. This case can be detected as the indentation box will be void.

```
185     {\setbox\z@\lastbox
186     \ifvoid\z@
187     \kern-\itemindent
188     \fi}%

189     \box\@labels
190     \penalty\z@
191     \fi
```

This code is intended to prevent a page break after the first line of an item that comes immediately after a section title. It may be sensible to always forbid a page break after one line of an item? As with all such settings of `\clubpenalty` it is local so will have no effect if the item starts in a group.

Only resetting `\@nobreak` when it is true is now essential since now it is sometimes set locally.

```
192     \if@nobreak
193     \@nobreakfalse
194     \clubpenalty \@M
195     \else
196     \clubpenalty \@clubpenalty
197     \everypar{}%
198     \fi}%

199     \if@noitemarg
200     \@noitemargfalse
201     \if@nmbrlist

202     \refstepcounter\@listctr
203     \fi
204     \fi
```

We use `\sbox` to support colour commands.

```
205     \sbox\@tempboxa{\makelabel{#1}}%
206     \global\setbox\@labels\hbox{%
207     \unhbox\@labels
208     \hskip \itemindent
209     \hskip -\labelwidth
210     \hskip -\labelsep
211     \ifdim \wd\@tempboxa >\labelwidth
212     \box\@tempboxa
```



```

213     \else
214       \hbox to\labelwidth {\unhbox\@tempboxa}%
215     \fi
216     \hskip \labelsep}%
217   \ignorespaces}

(End definition for \@item.)

\makelabel

218 \def\makelabel#1{%
219   \@latex@error{Lonely \string\item--perhaps a missing
220     list environment}\@ehc}

(End definition for \makelabel.)

\@nbitem

221 \def\@nbitem{%
222   \@tempskipa\@outerparskip
223   \advance\@tempskipa -\parskip
224   \addvspace\@tempskipa}

(End definition for \@nbitem.)

\usecounter

225 \def\usecounter#1{\@nmbrrlisttrue\def\@listctr{#1}\setcounter{#1}\z@}

(End definition for \usecounter.)

```

1.6 Itemize and Enumerate

Enumeration is done with four counters: `enumi`, `enumii`, `enumiii` and `enumiv`, where `enumN` controls the numbering of the Nth level enumeration. The label is generated by the commands `\labelenumi` ... `\labelenumiv`, which should be defined by the document style. Note that `\p@enumN\theenumN` defines the output of a `\ref` command. A typical definition might be:

```

\def\theenumii{\alph{enumii}}
\def\p@enumii{\theenumi}
\def\labelenumii{(\theenumii)}

```

which will print the labels as ‘(a)’, ‘(b)’, ... and print a `\ref` as ‘3a’.

The item numbers are moved to the right of the label box, so they are always a distance of `\labelsep` from the item.

`\@enumdepth` holds the current enumeration nesting depth.

Itemization is controlled by four commands: `\labelitemi`, `\labelitemii`, `\labelitemiii`, and `\labelitemiv`. To cause the second-level list to be bulleted, you just define `\labelitemii` to be `•`. `\@itemspacing` and `\@itemdepth` are the analogs of `\@enumspacing` and `\@enumdepth`.

Historical L^AT_EX 2.09 comments (not necessarily accurate any more):

```

\enumerate ==
BEGIN
  if \@enumdepth > 3
    then errormessage: “Too deeply nested”.
    else \@enumdepth :=L \@enumdepth + 1

```

```

        \@enumctr :=L eval(enum@\romannumeral\the\@enumdepth)
        \list{\label{\@enumctr}}
            {\usecounter{\@enumctr}
             \makelabel{LABEL} == \hss \llap{LABEL}}
    fi
END

\endenumerate == \endlist
End of historical LATEX 2.09 comments.

\@enumdepth
226 \newcount\@enumdepth \@enumdepth = 0
    (End definition for \@enumdepth.)

\c@enumi
\c@enumii 227 \@definecounter{enumi}
\c@enumii 228 \@definecounter{enumii}
\c@enumiv 229 \@definecounter{enumiii}
230 \@definecounter{enumiv}
    (End definition for \c@enumi and others.)

enumerate
231 \def\enumerate{%
232   \ifnum \@enumdepth >\thr@@\toodeep\else
233     \advance\@enumdepth\@ne
234     \edef\@enumctr{enum\romannumeral\the\@enumdepth}%

235     \expandafter
236     \list
237       \csname label\@enumctr\endcsname
238       {\usecounter{\@enumctr}\def\makelabel##1{\hss\llap{##1}}}%
239   \fi}
240 \let\endenumerate =\endlist

Historical LATEX 2.09 comments (not necessarily accurate any more):
\itemize ==
BEGIN
  if \@itemdepth > 3
  then errormessage: 'Too deeply nested'.
  else \@itemdepth :=L \@itemdepth + 1
    \@itemitem == eval(labelitem\romannumeral\the\@itemdepth)
    \list{\@nameuse{\@itemitem}}
      {\makelabel{LABEL} == \hss \llap{LABEL}}
  fi
END

\enditemize == \endlist

End of historical LATEX 2.09 comments.

```

```

\@itemdepth
241 \newcount\@itemdepth \@itemdepth = 0
    (End definition for \@itemdepth.)

itemize
242 \def\itemize{%
243   \ifnum \@itemdepth >\thr@@\toodeep\else
244     \advance\@itemdepth\@ne
245     \edef\@itemitem{labelitem\romannumeral\the\@itemdepth}%

246     \expandafter
247     \list
248     \csname\@itemitem\endcsname
249     {\def\makelabel##1{\hss\llap{##1}}}%
250   \fi}

251 \let\enditemize =\endlist
252 \endkernel

```

File J

ltboxes.dtx

1 L^AT_EX Box commands

<code>\makebox</code>	<code>\makebox[⟨wid⟩][⟨pos⟩]{⟨obj⟩}</code> Puts <code>⟨obj⟩</code> in an <code>\hbox</code> of width <code>⟨wid⟩</code> , positioned by <code>⟨pos⟩</code> . The possible <code>⟨pos⟩</code> are: s stretched, l flushleft, r flushright, c (default) centred. If <code>⟨wid⟩</code> is missing, then <code>⟨pos⟩</code> is also missing and <code>⟨obj⟩</code> is put in an <code>\hbox</code> of its natural width. <code>\makebox(⟨x⟩,⟨y⟩)[⟨pos⟩]{⟨obj⟩}</code> Puts <code>⟨obj⟩</code> in an <code>\hbox</code> of width <code>x * \unitlength</code> and height <code>y * \unitlength</code> . <code>⟨pos⟩</code> arguments are s, l, r or c (default) for stretched, flushleft, flushright or centred, and t or b for top, bottom – or combinations like tr or rb. Default for horizontal and vertical are centered. Note that in this picture mode version of <code>\makebox</code> a [b] aligns on the <i>bottom</i> of the text as documented. If you want to align on the <i>baseline</i> use <code>\makebox(,) [b]{\raisebox{0pt}{\height}[0pt]{xyz}}</code> or <code>\makebox(,) [b]{\smash{xyz}}</code>
<code>\mbox</code>	<code>\mbox{⟨obj⟩}</code> The same as <code>\makebox{⟨obj⟩}</code> , but is more efficient as no checking for optional arguments is done.
<code>\newsavebox</code>	<code>\newsavebox{⟨cmd⟩}</code> : If <code>⟨cmd⟩</code> is undefined, then defines it to be a T _E X box register.
<code>\savebox</code>	<code>\savebox{⟨cmd⟩} ...</code> : <code>⟨cmd⟩</code> is defined to be a T _E X box register, and the ‘...’ are any <code>\makebox</code> arguments. It is like <code>\makebox</code> , except it doesn’t produce text but saves the value in <code>\box ⟨cmd⟩</code> .
<code>\sbox</code>	<code>\sbox{⟨cmd⟩}{⟨obj⟩}</code> is an efficient abbreviation for <code>\savebox{⟨cmd⟩}{⟨obj⟩}</code> .
<code>lrbox</code>	<code>\begin{lrbox}{⟨cmd⟩}⟨text⟩\end{lrbox}</code> is equivalent to <code>\sbox{⟨cmd⟩}{⟨text⟩}</code> except that any white space at the beginning and end of <code>⟨text⟩</code> is ignored.
<code>\framebox</code>	<code>\framebox ...</code> : like <code>\makebox</code> , except it puts a ‘frame’ around the box. The frame is made of lines of thickness <code>\fboxrule</code> , separated by space <code>\fboxsep</code> from the text – except for <code>\framebox(X,Y) ...</code> , where the thickness of the lines is as for the picture environment, and there is no separation added.
<code>\fbox</code>	<code>\fbox{⟨obj⟩}</code> is an abbreviation for <code>\framebox{⟨obj⟩}</code> .
<code>\parbox</code>	<code>\parbox[⟨pos⟩][⟨height⟩][⟨inner-pos⟩]{⟨width⟩}{⟨text⟩}</code> : Makes a box with <code>\hsize</code> <code>⟨width⟩</code> , positioned by <code>⟨pos⟩</code> as follows: c : <code>\vcenter</code> (placed in <code>\$...\$</code> if not in math mode) b : <code>\vbox</code> t : <code>\vtop</code> default value is c. Sets <code>\hsize := ⟨width⟩</code> and calls <code>\@parboxrestore</code> , which does the following: Restores the original definitions of: <code>\par</code> <code>\</code> <code>\-</code> <code>\,</code> <code>\‘</code> <code>\=</code>

Resets the following parameters:

<code>\parindent</code>	<code>= 0pt</code>	
<code>\parskip</code>	<code>= 0pt</code>	added 20 Jan 87
<code>\linewidth</code>	<code>= \hsize</code>	
<code>\@totalleftmargin</code>	<code>= 0pt</code>	
<code>\leftskip</code>	<code>= 0pt</code>	
<code>\rightskip</code>	<code>= 0pt</code>	
<code>\@rightskip</code>	<code>= 0pt</code>	
<code>\parfillskip</code>	<code>= 0pt plus 1fil</code>	
<code>\lineskip</code>	<code>= \normallineskip</code>	
<code>\baselineskip</code>	<code>= \normalbaselineskip</code>	

Calls `\sloppy`

Note: `\@arrayparboxrestore` same as `\@parboxrestore` but it doesn't restore `\@`.

minipage : Similar to `\parbox`, except it also makes this look like a page by setting `\textwidth == \columnwidth == box width`

changes footnotes by redefining:

```

\@mpfn == mpfootnote
\thempfn == \thempfootnote
\@footnotetext == \@mpfootnotetext

```

resets the following list environment parameters:

```

\@listdepth == \@mplistdepth

```

where `\@mplistdepth` is initialized to zero,

and executes `\@minipagerestore` to allow the document style to reset any other parameters it desires. It sets `@minipage` true, and resets `\everypar` to set it false. This switch keeps `\addvspace` from putting space at the top of a minipage.

Change added 24 May 89: `\minipage` sets `@minipage` globally; `\endminipage` resets it false.

<code>\rule</code>	<code>\rule[⟨raised⟩]{⟨width⟩}{⟨height⟩}</code> : Makes a <code>⟨width⟩*⟨height⟩</code> rule, raised <code>⟨raised⟩</code> .
<code>\underline</code>	<code>\underline{⟨text⟩}</code> : Makes an underlined hbox with <code>⟨text⟩</code> in it.
<code>\raisebox</code>	<code>\raisebox{⟨distance⟩}[⟨height⟩][⟨depth⟩]{⟨box⟩}</code> :

Raises `⟨box⟩` up by `⟨distance⟩` length (down if `⟨distance⟩` negative). Makes T_EX think that the new box extends `⟨height⟩` above the line and `⟨depth⟩` below, for a total vertical length of `⟨height⟩+⟨depth⟩`. Default values of `⟨height⟩` & `⟨depth⟩` = actual height and depth of box in new position.

```

1  ⟨*2ekernel⟩
2  \message{boxes,}

```

\makebox `\makebox` User level command just looks for optional [or (.

```

3  ⟨/2ekernel⟩
4  ⟨latexrelease⟩\IncludeInRelease{2015/01/01}%
5  ⟨latexrelease⟩          {⟨makebox⟩}{Make  ⟨makebox robust⟩}%
6  ⟨*2ekernel | latexrelease⟩
7  \DeclareRobustCommand⟨makebox⟩{
8    \leavevmode
9    \@ifnextchar(%)
10     \@makepicbox
11     {\@ifnextchar[⟨makebox⟩\mbox]}%
12  ⟨/2ekernel | latexrelease⟩
13  ⟨latexrelease⟩\EndIncludeInRelease
14  ⟨latexrelease⟩\IncludeInRelease{0000/00/00}%
15  ⟨latexrelease⟩          {⟨makebox⟩}{Make  ⟨makebox robust⟩}%

```

```

16 <latexrelease>\def\makebox{%
17 <latexrelease> \leavevmode
18 <latexrelease> \@ifnextchar(%)
19 <latexrelease> \@makepicbox
20 <latexrelease> {\@ifnextchar[\@makebox\mbox}}%
21 <latexrelease>\expandafter\let\cename makebox \endcsname\undefined
22 <latexrelease>\EndIncludeInRelease
23 <*2ekernel>

```

(End definition for \makebox.)

\mbox The basic horizontal box command for L^AT_EX.

```
24 \DeclareRobustCommand\mbox[1]{\leavevmode\hbox{#1}}
```

(End definition for \mbox.)

\@makebox Look for a possible second optional argument (defaults to c).

```

25 \def\@makebox[#1]{%
26 \@ifnextchar [{\@imakebox[#1]}{\@imakebox[#1][c]}}

```

(End definition for \@makebox.)

\@begin@tempboxa Helper macro for supporting \height, \width etc. Grab #1 into \@tempboxa and measure it.

```

27 \long\def\@begin@tempboxa#1#2{%
28 \begingroup
29 \setbox\@tempboxa#1{\color@begingroup#2\color@endgroup}%
30 \def\width{\wd\@tempboxa}%
31 \def\height{\ht\@tempboxa}%
32 \def\depth{\dp\@tempboxa}%
33 \let\totalheight\@ovri
34 \totalheight\height
35 \advance\totalheight\depth}

```

(End definition for \@begin@tempboxa.)

\@end@tempboxa End the group started by \@begin@tempboxa, so that the scope of \height only includes the ‘length’ argument to the user-command.

```
36 \let\@end@tempboxa\endgroup
```

(End definition for \@end@tempboxa.)

\bm@c Set up spacing.

```

\bm@l 37 \def\bm@c{\hss\unhbox\@tempboxa\hss}
\bm@r 38 \def\bm@l{\unhbox\@tempboxa\hss}\let\bm@t\bm@l
\bm@s 39 \def\bm@r{\hss\unhbox\@tempboxa}\let\bm@b\bm@r
\bm@t 40 \def\bm@s{\unhbox\@tempboxa}
\bm@b

```

(End definition for \bm@c and others.)

\@imakebox Internal form of \makebox.

```

41 \long\def\@imakebox[#1][#2]#3{%
42 \@begin@tempboxa\hbox{#3}%
43 \setlength\@tempdima{#1}% support calc
44 \hb@xt@\@tempdima{\cename bm@#2\endcsname}%
45 \@end@tempboxa}

```

(End definition for \makebox.)

\@makepicbox Picture mode form of \makebox.

```

46 \def\@makepicbox(#1,#2){%
47   \ifnextchar[{\@makepicbox(#1,#2)}{\@makepicbox(#1,#2) []}}

```

(End definition for \@makepicbox.)

\@imakepicbox picture mode version

```

48 \if2kernel
49 \if*2kernel | latexrelease
50 \if latexrelease \IncludeInRelease{2020/10/01}%
51 \if latexrelease {\@imakepicbox}{default units}%
52 \long\def\@imakepicbox(#1,#2)[#3]#4{%
53   \@defaultunitsset\@tempdimc{#2}\unitlength
54   \vbox to\@tempdimc
55     {\let\mb@b\vss \let\mb@l\hss \let\mb@r\hss
56      \let\mb@t\vss
57      \@tfor\reserved@a :=#3\do{%
58        \if s\reserved@a
59          \let\mb@l\relax \let\mb@r\relax
60        \else
61          \expandafter\let\csname mb@\reserved@a\endcsname\relax
62        \fi}%
63      \mb@t
64      \@defaultunitsset\@tempdimc{#1}\unitlength
65      \hb@xt@\@tempdimc{\mb@l #4\mb@r}%
66      \mb@b

```

This kern ensures that a b option aligns on the bottom of the text rather than the baseline. this is the documented behaviour in the L^AT_EX Book. The kern is removed in compatibility mode.

```

67   \kern\z@}}
68 \if2kernel | latexrelease
69 \if latexrelease \EndIncludeInRelease
70 \if latexrelease \IncludeInRelease{0000/00/00}%
71 \if latexrelease {\@imakepicbox}{default units}%
72 \if latexrelease \long\def\@imakepicbox(#1,#2)[#3]#4{%
73 \if latexrelease \vbox to#2\unitlength
74 \if latexrelease {\let\mb@b\vss \let\mb@l\hss \let\mb@r\hss
75 \if latexrelease \let\mb@t\vss
76 \if latexrelease \@tfor\reserved@a :=#3\do{%
77 \if latexrelease \if s\reserved@a
78 \if latexrelease \let\mb@l\relax \let\mb@r\relax
79 \if latexrelease \else
80 \if latexrelease \expandafter\let\csname mb@\reserved@a\endcsname\relax
81 \if latexrelease \fi}%
82 \if latexrelease \mb@t
83 \if latexrelease \hb@xt@ #1\unitlength{\mb@l #4\mb@r}%
84 \if latexrelease \mb@b
85 \if latexrelease \kern\z@}}
86 \if latexrelease \EndIncludeInRelease
87 \if*2kernel

```

(End definition for \imakepicbox.)

`\set@color` This macro is initially a no-op, but the color package will redefine it to insert a `\special`.
88 `\let\set@color\relax`

(End definition for \set@color.)

`\color@begingroup` In the past these macros were initially no-ops, and the color package redefined them to be `\begingroup`, `\endgroup`, `\begingroup\set@color`,
`\color@endgroup` `\hbox\bgroup\color@begingroup`, `\color@endgroup\egroup`. and `\set to main document`
`\color@setgroup` `color` respectively.
`\normalcolor`

Nowadays we always set the group already in the kernel as this makes the coding simpler.

`\color@hbox`
`\color@vbox`
`\color@endbox`

```

89 </2ekernel>
90 <*2ekernel | latexrelease>
91 <latexrelease>\IncludeInRelease{2021/06/01}%
92 <latexrelease>                {\color@begingroup}{color group settings}%
93 \let\color@begingroup\begingroup
94 \def\color@endgroup{\endgraf\endgroup}
95 \def\color@setgroup{\color@begingroup}          % changed further in color package
96 \let\normalcolor\relax                        % remains untouched; only changed in a color pa
97 \def\color@hbox{\hbox\bgroup\color@begingroup}
98 \def\color@vbox{\vbox\bgroup\color@begingroup}
99 \def\color@endbox{\color@endgroup\egroup}
100 </2ekernel | latexrelease>
101 <latexrelease>\EndIncludeInRelease
102 <latexrelease>\IncludeInRelease{0000/00/00}%
103 <latexrelease>                {\color@begingroup}{color group settings}%
104 <latexrelease>
105 <latexrelease>\let\color@begingroup\relax
106 <latexrelease>\let\color@endgroup\relax
107 <latexrelease>\let\color@setgroup\relax
108 <latexrelease>\let\normalcolor\relax
109 <latexrelease>\let\color@hbox\relax
110 <latexrelease>\let\color@vbox\relax
111 <latexrelease>\let\color@endbox\relax
112 <latexrelease>
113 <latexrelease>\EndIncludeInRelease
114 <*2ekernel>

```

(End definition for \color@begingroup and others.)

`\newsavebox` Allocate a new ‘savebox’.
115 `\def\newsavebox#1{\@ifdefinable{#1}{\newbox#1}}`

(End definition for \newsavebox.)

`\savebox` Save #1 in a box register.

```

116 </2ekernel>
117 <latexrelease>\IncludeInRelease{2015/01/01}%
118 <latexrelease>                {\savebox}{Make \savebox robust}%
119 <*2ekernel | latexrelease>
120 \DeclareRobustCommand\savebox[1]{%
121   \@ifnextchar(%)

```



```

122     {\@savepicbox#1}{\@ifnextchar[\@savebox#1]{\sbox#1}}}%
123 </2ekernel | latexrelease>
124 <latexrelease>\EndIncludeInRelease
125 <latexrelease>\IncludeInRelease{0000/00/00}%
126 <latexrelease>           {\savebox}{Make \savebox robust}%
127 <latexrelease>\def\savebox#1{%
128 <latexrelease>  \@ifnextchar(%)
129 <latexrelease>    {\@savepicbox#1}{\@ifnextchar[\@savebox#1]{\sbox#1}}}%
130 <latexrelease>\expandafter\let\csname savebox \endcsname\undefined
131 <latexrelease>\EndIncludeInRelease
132 <*2ekernel>

```

(End definition for \savebox.)

\sbox Save #1 in a box register.

```

133 \DeclareRobustCommand\sbox[2]{\setbox#1\hbox{%
134   \color@setgroup#2\color@endgroup}}

```

(End definition for \sbox.)

\@savebox Look for second optional argument.

```

135 \def\@savebox#1[#2]{%
136   \@ifnextchar [\@isavebox#1[#2]]{\@isavebox#1[#2][c]}}

```

(End definition for \@savebox.)

\@isavebox

```

137 \long\def\@isavebox#1[#2][#3]#4{%
138   \sbox#1{\@imakebox[#2][#3]{#4}}}

```

(End definition for \@isavebox.)

\@savepicbox Picture mode version of \savebox.

```

139 \def\@savepicbox#1(#2,#3){%
140   \@ifnextchar[%]
141     {\@isavepicbox#1(#2,#3)}{\@isavepicbox#1(#2,#3) []}}

```

(End definition for \@savepicbox.)

\@isavepicbox Picture mode version of \savebox.

```

142 \long\def\@isavepicbox#1(#2,#3)[#4]#5{%
143   \sbox#1{\@imakepicbox(#2,#3)[#4]{#5}}}

```

(End definition for \@isavepicbox.)

\lrbox **lrbox**: the new environment form of \sbox. Use \aftergroup tricks to enable a *local* assignment to be made to the box, in a way that it still has an effect *outside* the **lrbox** environment.

```

144 \def\lrbox#1{%
145   \edef\reserved@a{%
146     \endgroup
147     \setbox#1\hbox{%
148       \begingroup\aftergroup}%
149     \def\noexpand\@currenvir{\@currenvir}%
150     \def\noexpand\@currenvline{\on@line}}%

```

```

151 \reserved@a
152 \@endpfalse
153 \color@setgroup
154 \ignorespaces}

(End definition for \lrbox.)

\endlrbox End the lrbox environment.
155 \def\endlrbox{\unskip\color@endgroup}

(End definition for \endlrbox.)

\usebox unchanged
156 \DeclareRobustCommand\usebox[1]{\leavevmode\copy #1\relax}

(End definition for \usebox.)

\frame The following definition of \frame was written by Pavel Curtis (Extra space removed 14
Jan 88) RmS 92/08/24: Replaced occurrence of \@halfwidth by \@wholewidth
157 \DeclareRobustCommand\frame[1]{%
158 \leavevmode
159 \hbox{%
160 \hskip-\@wholewidth
161 \vbox{%
162 \vskip-\@wholewidth
163 \hrule \@height\@wholewidth
164 \hbox{%
165 \vrule\@width\@wholewidth
166 #1%
167 \vrule\@width\@wholewidth}%
168 \hrule \@height\@wholewidth
169 \vskip-\@wholewidth}%
170 \hskip-\@wholewidth}}

(End definition for \frame.)

\fbxrule user level parameters,
\fbxsep 171 \newdimen\fbxrule
172 \newdimen\fbxsep

(End definition for \fbxrule and \fbxsep.)

\fbx Abbreviated framed box command.
173 \DeclareRobustCommand\fbx[1]{%
174 \leavevmode
175 \setbox\@tempboxa\hbox{%
176 \color@begingroup
177 \kern\fbxsep{#1}\kern\fbxsep
178 \color@endgroup}%
179 \@frameb@x\relax}

(End definition for \fbx.)

```

`\framebox` Framed version of `\makebox`.

```

180 </2ekernel>
181 <latexrelease>\IncludeInRelease{2015/01/01}%
182 <latexrelease>          {\framebox}{Make \framebox robust}%
183 <*2ekernel | latexrelease>
184 \DeclareRobustCommand\framebox{%
185   \@ifnextchar(%)
186     \@framepicbox{\@ifnextchar[\@framebox\fbbox}}%
187 </2ekernel | latexrelease>
188 <latexrelease>\EndIncludeInRelease
189 <latexrelease>\IncludeInRelease{0000/00/00}%
190 <latexrelease>          {\framebox}{Make \framebox robust}%
191 <latexrelease>\def\framebox{%
192 <latexrelease>  \@ifnextchar(%)
193 <latexrelease>    \@framepicbox{\@ifnextchar[\@framebox\fbbox}}%
194 <latexrelease>\expandafter\let\csname framebox \endcsname\@undefined
195 <latexrelease>\EndIncludeInRelease
196 <*2ekernel>

```

(End definition for \framebox.)

`\@framebox` Deal with optional arguments.

```

197 \def\@framebox[#1]{%
198   \@ifnextchar[%]
199     {\@framebox[#1]}%
200     {\@framebox[#1][c]}}

```

(End definition for \@framebox.)

`\@ifframebox` The handling the optional arguments. In order to set the whole box, including the frame to the specified dimension, we first determine that dimension from the natural size of the text, #3. calculated width.

```

201 \long\def\@ifframebox[#1][#2]#3{%
202   \leavevmode
203   \@begin@tempboxa\hbox{#3}%
204   \setlength\@tempdima{#1}%
205   \setbox\@tempboxa\hb@xt@\@tempdima
206     {\kern\fbboxsep\csname bm@#2\endcsname\kern\fbboxsep}%
207   \@frameb@x{\kern-\fbboxrule}%
208   \@end@tempboxa}

```

(End definition for \@ifframebox.)

`\@frameb@x` Common part of `\framebox` and `\fbbox`. #1 is a negative kern in the `\framebox` case so that the vertical rules do not add to the width of the box.

```

209 \def\@frameb@x#1{%
210   \@tempdima\fbboxrule
211   \advance\@tempdima\fbboxsep
212   \advance\@tempdima\dp\@tempboxa
213   \hbox{%
214     \lower\@tempdima\hbox{%
215       \vbox{%
216         \hrule\@height\fbboxrule
217         \hbox{%

```

```

218         \vrule\@width\fbboxrule
219         #1%
220         \vbox{%
221             \vskip\fbboxsep
222             \box\@tempboxa
223             \vskip\fbboxsep}%
224         #1%
225         \vrule\@width\fbboxrule}%
226     \hrule\@height\fbboxrule}%
227         }%
228     }%
229 }

```

(End definition for \@frameb@x.)

\@framepicbox Picture mode version.

```

230 \def\@framepicbox(#1,#2){%
231     \@ifnextchar[{\@ifframepicbox(#1,#2)}{\@ifframepicbox(#1,#2) []}]

```

(End definition for \@framepicbox.)

\@ifframepicbox Picture mode version.

```

232 \long\def\@ifframepicbox(#1,#2)[#3]#4{%
233     \frame{\@imakepicbox(#1,#2)[#3]{#4}}

```

(End definition for \@ifframepicbox.)

\parbox The main vertical-box command for L^AT_EX.

```

234 \if2ekernel
235 \<latexrelease>\IncludeInRelease{2015/01/01}%
236 \<latexrelease>          {\parbox}{Make \parbox robust}%
237 \if2ekernel | latexrelease
238 \DeclareRobustCommand\parbox{%
239     \@ifnextchar[%]
240         \@iparbox
241         {\@iiiparbox c\relax[s]}}%
242 \if2ekernel | latexrelease
243 \<latexrelease>\EndIncludeInRelease
244 \<latexrelease>\IncludeInRelease{0000/00/00}%
245 \<latexrelease>          {\parbox}{Make \parbox robust}%
246 \<latexrelease>\def\parbox{%
247 \<latexrelease>     \@ifnextchar[%]
248 \<latexrelease>     \@iparbox
249 \<latexrelease>     {\@iiiparbox c\relax[s]}}%
250 \<latexrelease>\expandafter\let\csname parbox \endcsname\@undefined
251 \<latexrelease>\EndIncludeInRelease
252 \if2ekernel

```

(End definition for \parbox.)

\@iparbox Optional argument handling.

```

253 \def\@iparbox[#1]{%
254     \@ifnextchar[%]
255         {\@iiiparbox{#1}}%
256         {\@iiiparbox{#1}\relax[s]}

```

(End definition for \@iparbox.)

\@iiparbox Optional argument handling.

```

257 \def\@iiparbox#1[#2]{%
258   \@ifnextchar[%
259     {\@iiparbox{#1}{#2}}%
260     {\@iiparbox{#1}{#2}[#1]}}

```

(End definition for \@iiparbox.)

\@iiparbox The internal version of \parbox.

```

\@parboxto
261 \let\@parboxto\@empty
262 \long\def\@iiparbox#1#2[#3]#4#5{%
263   \leavevmode
264   \@pboxswfalse
265   \setlength\@tempdima{#4}%
266   \@begin@tempboxa\vbox{\hsize\@tempdima\@parboxrestore#5\@par}%
267   \ifx\relax#2\else
268     \setlength\@tempdimb{#2}%
269     \edef\@parboxto{to\the\@tempdimb}%
270   \fi
271   \if#1b\vbox
272   \else\if #1t\vtop
273   \else\ifmmode\vcenter
274   \else\@pboxswtrue $\vcenter
275   \fi\fi\fi
276   \@parboxto{\let\hss\vss\let\unhbox\unvbox
277     \csname bm#3\endcsname}%
278   \if@pboxsw \m@th$\fi
279   \@end@tempboxa}

```

(End definition for \@iiparbox and \@parboxto.)

\@arrayparboxrestore Restore various paragraph parameters.

The rationale for allowing two normally global flags to be set locally here was stated originally by Donald Arseneau and extended by Chris Rowley. It is because these flags are only set globally to true by section commands, and these should never appear within boxes or, indeed, in any group; and they are only ever set globally to false when they are definitely true.

If anyone is unhappy with this argument then both flags should be treated as in \set@nbreak; otherwise this command will be redundant.

```

280 </2ekernel>
281 <latexrelease>\IncludeInRelease{2017-04-15}%
282 <latexrelease>          {\normallineskiplimit}
283 <latexrelease>          {reset \lineskiplimit}%
284 <*2ekernel | latexrelease>
285 \def\@arrayparboxrestore{%
286   \let\if@nbreak\iffalse
287   \let\if@noskipsec\iffalse
288   \let\par\@par
289   \let\-\@dischyph

```

Redefined accents to allow changes in font encoding

```

290 \let'\@acci\let'\@accii\let\=\@acciii
291 \parindent\z@ \parskip\z@skip
292 \everypar{}%
293 \linewidth\hsize
294 \@totalleftmargin\z@
295 \leftskip\z@skip \rightskip\z@skip \@rightskip\z@skip
296 \parfillskip\@flushglue
297 \lineskip\normallineskip

298 \lineskiplimit\normallineskiplimit

299 \baselineskip\normalbaselineskip
300 \sloppy}
301 /2ekernel | latexrelease)

302 <latexrelease>\EndIncludeInRelease
303 <latexrelease>\IncludeInRelease{0000-00-00}%
304 <latexrelease> \{\normallineskiplimit}
305 <latexrelease> \{reset \lineskiplimit}%
306 <latexrelease>\def\@arrayparboxrestore{%
307 <latexrelease> \let@if@nobreak\iffalse
308 <latexrelease> \let@if@noskipsec\iffalse
309 <latexrelease> \let\par\@par
310 <latexrelease> \let-\@dischyph
311 <latexrelease> \let'\@acci\let'\@accii\let\=\@acciii
312 <latexrelease> \parindent\z@ \parskip\z@skip
313 <latexrelease> \everypar{}%
314 <latexrelease> \linewidth\hsize
315 <latexrelease> \@totalleftmargin\z@
316 <latexrelease> \leftskip\z@skip \rightskip\z@skip \@rightskip\z@skip
317 <latexrelease> \parfillskip\@flushglue \lineskip\normallineskip
318 <latexrelease> \baselineskip\normalbaselineskip
319 <latexrelease> \sloppy}
320 <latexrelease>\EndIncludeInRelease
321 *2ekernel)

```

(End definition for \@arrayparboxrestore.)

\parboxrestore Restore various paragraph parameters, and also \.

```

322 \def\@parboxrestore{\@arrayparboxrestore\let\\\@normalcr}

```

(End definition for \parboxrestore.)

\if@minipage Switch that is true at the start of a minipage.

```

323 \def\@minipagefalse{\global\let@if@minipage\iffalse}
324 \def\@minipagetrue {\global\let@if@minipage\iftrue}
325 \@minipagefalse

```

(End definition for \if@minipage.)

\minipage Essentially an environment form of \parbox.

```

326 \def\minipage{%
327 \ifnextchar[%]
328 \@iminipage
329 {\@iiiminipage c\relax[s]}}

```

(End definition for \minipage.)

\@iminipage Optional argument handling.

```
330 \def\@iminipage[#1]{%
331   \@ifnextchar[%
332     {\@iminipage{#1}}%
333     {\@iiiminipage{#1}\relax[s]}}
```

(End definition for \@iminipage.)

\@iiiminipage Optional argument handling.

```
334 \def\@iiiminipage#1[#2]{%
335   \@ifnextchar[%
336     {\@iiiminipage{#1}{#2}}%
337     {\@iiiminipage{#1}{#2}[#1]}}
```

(End definition for \@iiiminipage.)

\@iiiminipage Internal form of minipage.

```
338 \def\@iiiminipage#1#2[#3]#4{%
339   \leavevmode
340   \@pboxswfalse
341   \setlength\@tempdima{#4}%
342   \def\@mpargs{{#1}{#2}[#3]{#4}}%
343   \setbox\@tempboxa\vbox\bgroup
344     \color@begingroup
345     \hsize\@tempdima
346     \textwidth\hsize \columnwidth\hsize
347     \@parboxrestore
348     \def\@mpfn{\mpfootnote}\def\thempfn{\thempfootnote}\c@mpfootnote\z@
349     \let\@footnotetext\@mpfootnotetext
350     \let\@listdepth\@mplistdepth \@mplistdepth\z@
351     \@minipagerestore
352     \@setminipage}
```

(End definition for \@iiiminipage.)

\@minipagerestore Hook so that other styles can reset other commands in a minipage.

```
353 \let\@minipagerestore=\relax
```

(End definition for \@minipagerestore.)

\endminipage

```
354 \def\endminipage{%
355   \par
356   \unskip
357   \ifvoid\@mpfootins\else
358     \vskip\skip\@mpfootins
359     \normalcolor
360     \footnoterule
361     \unvbox\@mpfootins
362   \fi
363   \@minipagefalse   %% added 24 May 89
364   \color@endgroup
365   \egroup
366   \expandafter\@iiiparbox\@mpargs{\unvbox\@tempboxa}}
```

(End definition for \endminipage.)

\@mplistdepth Versions of \@listdepth and \footins local to minipage.

```
\@mpfootins 367 \newcount\@mplistdepth
368 \newinsert\@mpfootins
```

(End definition for \@mplistdepth and \@mpfootins.)

\@mpfootnotetext Minipage version of \@footnotetext.

Final \strut added 27 Mar 89, on suggestion by Don Hosek

```
369 </2ekernel>
370 <*2ekernel | latexrelease>
371 <latexrelease>\IncludeInRelease{2021/06/01}%
372 <latexrelease>          {\@mpfootnotetext}{footnotetext tagging}%
373 \long\def\@mpfootnotetext#1{%
374   \global\setbox\@mpfootins\vbox{%
375     \unvbox\@mpfootins
376     \reset@font\footnotesize
377     \hsize\columnwidth
378     \@parboxrestore
379     \protected@edef\@currentlabel
380       {\csname p@mpfootnote\endcsname\@thefnmark}%
381     \color@begingroup
382       \@makefntext{%
383         \rule\z@\footnotesep\ignorespaces#1\@finalstrut\strutbox}%
384     \par
385     \color@endgroup}}
386 </2ekernel | latexrelease>
387 <latexrelease>\EndIncludeInRelease

388 <latexrelease>\IncludeInRelease{0000/00/00}%
389 <latexrelease>          {\@mpfootnotetext}{footnotetext tagging}%
390 <latexrelease>
391 <latexrelease>\long\def\@mpfootnotetext#1{%
392 <latexrelease>   \global\setbox\@mpfootins\vbox{%
393 <latexrelease>     \unvbox\@mpfootins
394 <latexrelease>     \reset@font\footnotesize
395 <latexrelease>     \hsize\columnwidth
396 <latexrelease>     \@parboxrestore
397 <latexrelease>     \protected@edef\@currentlabel
398 <latexrelease>       {\csname p@mpfootnote\endcsname\@thefnmark}%
399 <latexrelease>     \color@begingroup
400 <latexrelease>       \@makefntext{%
401 <latexrelease>         \rule\z@\footnotesep\ignorespaces#1\@finalstrut\strutbox}%
402 <latexrelease>       \color@endgroup}}
403 <latexrelease>
404 <latexrelease>\EndIncludeInRelease
405 <*2ekernel>
```

(End definition for \@mpfootnotetext.)

```
406 \newif\if@pboxsw
```


`\rule` Draw a rule of the specified size.

```

407 </2ekernel>
408 <latexrelease>\IncludeInRelease{2015/01/01}%
409 <latexrelease>          {\rule}{Make \rule robust}%
410 <*2ekernel | latexrelease>
411 \DeclareRobustCommand\rule{\@ifnextchar[\@rule{\@rule[\z@]}}%
412 </2ekernel | latexrelease>
413 <latexrelease>\EndIncludeInRelease
414 <latexrelease>\IncludeInRelease{0000/00/00}%
415 <latexrelease>          {\rule}{Make \rule robust}%
416 <latexrelease>\def\rule{\@ifnextchar[\@rule{\@rule[\z@]}}%
417 <latexrelease>\expandafter\let\csname rule \endcsname\@undefined
418 <latexrelease>\EndIncludeInRelease
419 <*2ekernel>

```

(End definition for \rule.)

`\@rule` Internal form of `\rule`.

```

420 \def\@rule[#1]#2#3{%
421   \leavevmode
422   \hbox{%
423     \setlength\@tempdima{#1}%
424     \setlength\@tempdimb{#2}%
425     \setlength\@tempdimc{#3}%
426     \advance\@tempdimc\@tempdima
427     \vrule\@width\@tempdimb\@height\@tempdimc\@depth-\@tempdima}}

```

(End definition for \@rule.)

`@@underline` Saved primitive `\underline`.

```

428 \let\@@underline\underline

```

(End definition for @@underline.)

`\underline` L^AT_EX version works outside math.

```

429 \DeclareRobustCommand\underline[1]{%
430   \relax
431   \ifmmode\@@underline{#1}%
432   \else $\@@underline{\hbox{#1}}\m@th$\relax\fi}

```

(End definition for \underline.)

`\raisebox` Raise a box, and change its vertical dimensions.

```

433 </2ekernel>
434 <latexrelease>\IncludeInRelease{2015/01/01}%
435 <latexrelease>          {\raisebox}{Make \raisebox robust}%
436 <*2ekernel | latexrelease>
437 \DeclareRobustCommand\raisebox[1]{%
438   \leavevmode
439   \@ifnextchar[\@rsbox{#1}]{\@irsbox{#1}[]}}
440 </2ekernel | latexrelease>
441 <latexrelease>\EndIncludeInRelease
442 <latexrelease>\IncludeInRelease{0000/00/00}%
443 <latexrelease>          {\raisebox}{Make \raisebox robust}%
444 <latexrelease>\def\raisebox#1{%

```

```

445 <latexrelease> \leavevmode
446 <latexrelease> \ifnextchar[{\@rsbox{#1}}{\@irsbox{#1}[]}]
447 <latexrelease>\expandafter\let\csname raisebox \endcsname\@undefined
448 <latexrelease>\EndIncludeInRelease
449 (*2ekernel)

(End definition for \raisebox.)

\@rsbox Optional argument handling.
450 \def\@rsbox#1[#2]{%
451   \ifnextchar[{\@iirsbox{#1}[#2]}{\@irsbox{#1}[#2]}}

(End definition for \@rsbox.)

\@argsbox ...

(End definition for \@argsbox.)

\@irsbox Internal version of \raisebox (less than two optional args).
452 \long\def\@irsbox#1[#2]#3{%
453   \@begin@tempboxa\hbox{#3}%
454   \setlength\@tempdima{#1}%
455   \ifx\#2\\\else\setlength\@tempdimb{#2}\fi
456   \setbox\@tempboxa\hbox{\raise\@tempdima\box\@tempboxa}%
457   \ifx\#2\\\else\ht\@tempboxa\@tempdimb\fi
458   \box\@tempboxa
459   \@end@tempboxa}

(End definition for \@irsbox.)

\@iirsbox Internal version of \raisebox (two optional args).
460 \long\def\@iirsbox#1[#2] [#3]#4{%
461   \@begin@tempboxa\hbox{#4}%
462   \setlength\@tempdima{#1}%
463   \setlength\@tempdimb{#2}%
464   \setlength\dimen@{#3}%
465   \setbox\@tempboxa\hbox{\raise\@tempdima\box\@tempboxa}%
466   \ht\@tempboxa\@tempdimb
467   \dp\@tempboxa\dimen@
468   \box\@tempboxa
469   \@end@tempboxa}

(End definition for \@iirsbox.)

\@finalstrut This macro adds a special strut the depth of the box given as #1, and height and width
Opt. It is used for ensuring that the last line of a paragraph has the correct depth in
‘p’ columns of tables and in footnotes. In vertical mode nothing is done, as adding the
strut (as done in 2.09) would start a new paragraph. It would be possible to inspect
\prevdepth to check the depth of the just-completed paragraph, but we do not do that
here. Actually we do even less now, skip the vmode test as it broke tabular ‘p’ columns.
.

The \nobreak was added (1995/10/31) to allow hyphenation of the final word of the
paragraph.
470 \def\@finalstrut#1{%
471   \unskip\ifhmode\nobreak\fi\vrule\@width\z@\@height\z@\@depth\dp#1}

(End definition for \@finalstrut.)

```

1.1 Some low-level constructs

The following commands are basically inherited from plain T_EX.

```
\leftline These macros place text on a full line either centred or left or right adjusted.
\rightline
\centerline
  @@line
```

```
472 \def\@@line{\hb@xt@\hsize}
473 \DeclareRobustCommand\leftline[1]{\@@line{#1\hss}}
474 \DeclareRobustCommand\rightline[1]{\@@line{\hss#1}}
475 \DeclareRobustCommand\centerline[1]{\@@line{\hss#1\hss}}
```

(End definition for \leftline and others.)

```
\rlap These macros place text to the left or right of the current reference point without taking
\llap up space.
\clap
```

```
476 \DeclareRobustCommand\rlap[1]{\hb@xt@\z@{#1\hss}}
477 \DeclareRobustCommand\llap[1]{\hb@xt@\z@{\hss#1}}
```

And here is the version that centers, it was initially introduced by `mathtools`.

```
478 \DeclareRobustCommand\clap[1]{\hb@xt@\z@{\hss#1\hss}}
```

(End definition for \rlap, \llap, and \clap.)

```
479 \</2ekernel>
```

File K

lftab.dtx

1 Tabbing, Tabular and Array Environments

This section deals with ‘Lining It Up in Columns’. First the `tabbing` environment is defined, and then in second part, `tabular` together with its variants, `tabular*` and `array`.

Note that the `tabular` defined here is essentially the original L^AT_EX 2.09 version, not the extended version described in *The L^AT_EX Companion*. Use the `array` package to obtain the extended version.

1.1 tabbing

Historical L^AT_EX 2.09 comments (not necessarily accurate any more):

`\dimen(\@firsttab + i)` = distance of tab stop `i` from left margin
0 <= `i` <= 15 (?).

`\dimen\@firsttab` is initialized to `\@totalleftmargin`, so it starts at the prevailing left margin.

`\@maxtab` = number of highest defined tab register
probably = `\@firsttab + 12`

`\@nxttabmar` = tab stop number of next line’s left margin

`\@curtabmar` = tab stop number of current line’s left margin

`\@curtab` = number of the current tab. At start of line,
it equals `\@curtabmar`

`\@hightab` = largest tab number currently defined.

`\@tabpush` = depth of `\pushtab`’s

`\box\@curline` = contents of current line, excluding left margin
skip, and excluding contents of current field

`\box\@curfield` = contents of current field

`@rjfield` = switch: T iff the last field of the line should
be right-justified at the right margin.

`\tabbingsep` = distance left by the `\’` command between the
current position and the field that is
“left-shifted”.

UTILITY MACROS

`\@stopfield` : closes the current field

`\@addfield` : adds the current field to the current line.

`\@contfield` : continues the current field

`\@startfield` : begins the next field

`\@stopline` : closes the current line and outputs it

```

\@startline : starts the next line
\@ifatmargin : an \if that is true iff the current line.
                has width zero

\@startline ==
BEGIN
  \@curtabmar :=G \@nxttabmar
  \@curtab :=G \@curtabmar
  \box\@curline :=G null
  \@startfield
  \strut
END

\@stopline ==
BEGIN
  \unskip
  \@stopfield
  if @rjfield = T
    then @rjfield :=G F
      \@tempdima := \@totalleftmargin + \linewidth
      \hb@xt@ \@tempdima{\@itemfudge
                          \hskip \dimen\@curtabmar
                          \box\@curline
                          \hfil
                          \box\@curfield}
    else \@addfield
      \hbox {\@itemfudge
            \hskip \dimen\@curtabmar
            \box\@curline}
    fi
  END

\@startfield ==
BEGIN
  \box\@curfield :=G \hbox {
END

\@stopfield ==
BEGIN
  }
END

\@contfield ==
BEGIN
  \box\@curfield :=G \hbox { \unhbox\@currfield %%} brace matching
END

\@addfield ==
BEGIN
  \box\@curline :=G \unbox\@curline * \unbox\@curfield
END

```

```

\@ifatmargin ==
BEGIN
  if dim of box\@curline = 0pt then
  END

\tabbing ==
BEGIN
  \lineskip :=L 0pt
  \> == \@rtab
  \< == \@ltab
  \= == \@settab
  \+ == \@tabplus
  \- == \@tabminus
  \‘ == \@tabrj
  \’ == \@tablab
  \\\ == BEGIN \@stopline \@startline END
  \\[DIST] == BEGIN
    \@stopline \vskip DIST \@startline\ignorespaces END
  \\\* == BEGIN \@stopline \penalty 10000 \@startline END
  \\\*[DIST] == BEGIN \@stopline \penalty 10000 \vskip DIST
    \@startline\ignorespaces END
  \@hightab := \@nxrtabmar :=G \@firsttab
  \@tabpush :=G 0
  \dimen\@firsttab := \@totalleftmargin
  @rjfield :=G F
  \trivlist \item\relax
  if @minipage = F then \vskip \parskip fi
  \box\@tabfbox = \rlap{\indent\the\everypar}
    % note: \the\everypar sets @inlabel :=G F
  \@itemfudge == BEGIN \box\@tabfbox END
  \@startline
  \ignorespaces
END

\@endtabbing ==
BEGIN
  \@stopline
  if \@tabpush > 0 then error message: "unmatched \poptabs" fi
  \endtrivlist
END

\@rtab ==
BEGIN
  \@stopfield
  \@addfield
  if \@curtab < \@hightab
    then \@curtab :=G \@curtab + 1
    else error message "Undefined Tab" fi

```

```

\@tempdima := \dimen\@curtab - \dimen\@curtabmar
              - width of box \@curline
\box\@curline :=G \hbox{\unhbox\@curline + \hskip\@tempdima}
\@startfield
END

\@settab ==
BEGIN
  \@stopfield
  \@addfield
  if \@curtab < \@maxtab
    then \@curtab :=G \@curtab+1
    else error message: "Too many tabs"      fi
  if \@curtab > \@hightab
    then \@hightab :=L \@curtab      fi
  \dimen\@curtab :=L \dimen\@curtabmar + width of \box\@curline
  \@startfield
END

\@ltab ==
BEGIN
  \@ifatmargin
  then if \@curtabmar > \@firsttab
    then \@curtab :=G \@curtab - 1
        \@curtabmar :=G \@curtabmar - 1
    else error message "Too many untab"      fi
  else error message "Left tab in middle of line"
  fi
END

\@tabplus ==
BEGIN
  if \@nxxtabmar < \@hightab
    then \@nxxtabmar :=G \@nxxtabmar+1
    else error message "Undefined tab"
  fi
END

\@tabminus ==
BEGIN
  if \@nxxtabmar > \@firsttab
    then \@nxxtabmar :=G \@nxxtabmar-1
    else error message "Too many untab"
  fi
END

\@tabrj ==
BEGIN \@stopfield
  \@addfield
  @rjfield :=G T

```

```

\@startfield
END

\@tablab ==
BEGIN \@stopfield
\box\@curline G:= \hbox{\box\@curline %% 'G' added 17 Jun 86
\hskip - width of \box\@curfield
\hskip -\tabbingsep
\box\@curfield
\hskip \tabbingsep }

\@startfield
END

\pushtabs ==
BEGIN
\@stopfield
\@tabpush :=G \@tabpush + 1
\beginngroup
\@contfield
END

\poptabs ==
BEGIN
\@stopfield
if \@tabpush > 0
then \endgroup
\@tabpush :=G \@tabpush - 1
else error message: "Too many \poptabs"
fi
\@contfield
END

```

End of historical L^AT_EX 2.09 comments.

\a The accents \‘, \’ , and \= that have been redefined inside a tabbing environment can be called by typing \a‘, \a’ , and \a=. The macro \a is defined in `ltoutenc.dtx`.

(End definition for \a.)

The ‘2ekernel’ code ensures that a `\usepackage{autotabg}` is essentially ignored if a ‘full’ format is being used that has picture mode already in the format.

```

1 <2ekernel>\expandafter\let\csname ver@autotabg.sty\endcsname\fmtversion

\@firsttab
\@maxtab
2 <*2ekernel>
3 \newdimen\@gtempa
4 \chardef\@firsttab=\the\allocationnumber
5 \newdimen\@gtempa\newdimen\@gtempa\newdimen\@gtempa\newdimen\@gtempa
6 \newdimen\@gtempa\newdimen\@gtempa\newdimen\@gtempa\newdimen\@gtempa
7 \newdimen\@gtempa\newdimen\@gtempa\newdimen\@gtempa\newdimen\@gtempa
8 \newdimen\@gtempa
9 \chardef\@maxtab=\the\allocationnumber
10 \dimen\@firsttab=0pt

```



```

(End definition for \@firsttab and \@maxtab.)

\@nxttabmar
\@curtabmar 11 \newcount\@nxttabmar
  \@curtab 12 \newcount\@curtabmar
  \@hightab 13 \newcount\@curtab
  \@tabpush 14 \newcount\@hightab
            15 \newcount\@tabpush

(End definition for \@nxttabmar and others.)

\@curline
\@curfield 16 \newbox\@curline
\@tabfbox 17 \newbox\@curfield
          18 \newbox\@tabfbox

(End definition for \@curline, \@curfield, and \@tabfbox.)

\if@rjfield
  19 \newif\if@rjfield

(End definition for \if@rjfield.)

\@startline It is, in some sense, an error if the current margin tab setting is higher than the value of
            \@hightab (which is a local variable). That this is allowed is a fundamental design flaw
            which is not going to be corrected now.
  20 \def\@startline{%
  21   \ifnum \@nxttabmar >\@hightab
  22     \@badtab
  23     \global\@nxttabmar \@hightab
  24   \fi
  25   \global\@curtabmar \@nxttabmar
  26   \global\@curtab \@curtabmar
  27   \global\setbox\@curline \hbox {%
  28     \@startfield
  29     \strut}

(End definition for \@startline.)

\@stopline
  30 \def\@stopline{%
  31   \unskip
  32   \@stopfield
  33   \if@rjfield
  34     \global\@rjfieldfalse
  35     \@tempdima\@totalleftmargin
  36     \advance\@tempdima\linewidth
  37     \hb@xt@\@tempdima{%
  38       \@itemfudge\hskip\dimen\@curtabmar
  39       \box\@curline
  40       \hfil
  41       \box\@curfield}%
  42   \else
  43     \@addfield
  44     \hbox{\@itemfudge\hskip\dimen\@curtabmar\box\@curline}%
  45   \fi}

```

```

(End definition for \@stopline.)

\@startfield
46 \def\@startfield{%
47   \global\setbox\@curfield\hbox\bgroup\color@begingroup}

(End definition for \@startfield.)

\@stopfield
48 \def\@stopfield{%
49   \color@endgroup\egroup}

(End definition for \@stopfield.)

\@contfield
50 \def\@contfield{%
51   \global\setbox\@curfield\hbox\bgroup\color@begingroup
52   \unhbox\@curfield}

(End definition for \@contfield.)

\@addfield
53 \def\@addfield{\global\setbox\@curline\hbox{\unhbox
54   \@curline\unhbox\@curfield}}

(End definition for \@addfield.)

\@ifatmargin
55 \def\@ifatmargin{\ifdim \wd\@curline =\z@}

(End definition for \@ifatmargin.)

\@tabcr
56 \protected\def\@tabcr{\@stopline \ifstar{\penalty \@M \@tabcr}\@tabcr}

(End definition for \@tabcr.)

\@xtabcr
57 \def\@xtabcr{\@ifnextchar[\@itabcr{\@startline\ignorespaces}}

(End definition for \@xtabcr.)

\@itabcr
58 \</2ekernel>
59 \< *2ekernel | latexrelease>
60 \< latexrelease>\IncludeInRelease{2020/10/01}%
61 \< latexrelease> \{\@itabcr\}{Tabbing calc syntax}%
62 \def\@itabcr[#1]{\@vspace@calcify{#1}\@startline\ignorespaces}
63 \</2ekernel | latexrelease>
64 \< latexrelease>\EndIncludeInRelease
65 \< latexrelease>\IncludeInRelease{0000/00/00}%
66 \< latexrelease> \{\@itabcr\}{Tabbing calc syntax}%
67 \< latexrelease>
68 \< latexrelease>\def\@itabcr[#1]{\vskip #1\@startline\ignorespaces}
69 \< latexrelease>\EndIncludeInRelease
70 \< *2ekernel>

```

```

tabbing We use \relax to prevent \item from scanning too far.
\begin{tabbing}
71 \def\tabbing{\lineskip \z@skip\let\>\@rtab\let\<\@ltab\let\=\@settab
72 \let\+\@tabplus\let\-\@tabminus\let\'\@tabrj\let\'\@tablab
73 \let\=\@tabcr
74 \@hightab\@firsttab
75 \global\@nxttabmar\@firsttab
76 \dimen\@firsttab\@totalleftmargin
77 \global\@tabpush\z@ \global\@rjfieldfalse
78 \trivlist \item\relax
79 \if@minipage\else\vskip\parskip\fi

80 \setbox\@tabfbox\hbox{%
81 \rlap{\hskip\@totalleftmargin\indent\the\everypar}}%
82 \def\@itemfudge{\box\@tabfbox}%
83 \@startline\ignorespaces}

\end{tabbing}
84 \def\endtabbing{%
85 \@stopline\ifnum\@tabpush >\z@ \@badpoptabs \fi\endtrivlist}

Omitted \global added to \@rtab 17 Jun 86
\@rtab
86 \def\@rtab{\@stopfield\@addfield\ifnum \@curtab<\@hightab
87 \global\advance\@curtab \@ne \else\@badtab\fi
88 \@tempdima\dimen\@curtab
89 \advance\@tempdima -\dimen\@curtabmar
90 \advance\@tempdima -\wd\@curline
91 \global\setbox\@curline\hbox{\unhbox\@curline\hskip\@tempdima}%
92 \@startfield\ignorespaces}

\@settab
93 \def\@settab{\@stopfield\@addfield
94 \ifnum \@curtab <\@maxtab
95 \ifnum\@curtab =\@hightab
96 \advance\@hightab \@ne
97 \fi
98 \global\advance\@curtab \@ne
99 \else
100 \@latexerror{Tab overflow}\@ehd
101 \fi
102 \dimen\@curtab \dimen\@curtabmar
103 \advance\dimen\@curtab \wd\@curline
104 \@startfield
105 \ignorespaces}

\@ltab
106 \def\@ltab{\@ifatmargin\ifnum\@curtabmar >\@firsttab
107 \global\advance\@curtab \m@ne \global\advance\@curtabmar\m@ne\else
108 \@badtab\fi\else
109 \@latexerror{\string\<\space in mid line}\@ehd\fi\ignorespaces}

\@tabplus
110 \def\@tabplus{%
111 \ifnum\@nxttabmar<\@hightab

```

```

112     \global\advance\@nxttabmar\@ne
113 \else
114     \@badtab
115 \fi
116 \ignorespaces}

\@tabminus 117 \def\@tabminus{%
118     \ifnum\@nxttabmar>\@firsttab
119         \global\advance\@nxttabmar\m@ne
120     \else
121         \@badtab
122     \fi
123     \ignorespaces}

\@tabrj 124 \def\@tabrj{%
125     \@stopfield\@addfield\global\@rjfieldtrue\@startfield\ignorespaces}

\setbox\@curline made \global in \@tablab. 17 Jun 86

\@tablab 126 \def\@tablab{%
127     \@stopfield
128     \global\setbox\@curline\hbox{%
129         \box\@curline
130         \hskip-\wd\@curfield \hskip-\tabbingsep
131         \box\@curfield
132         \hskip\tabbingsep}%
133     \@startfield
134     \ignorespaces}

135 \</2kernel>
136 \<*2kernel | latexrelease>
137 \<latexrelease>\IncludeInRelease{2019/10/01}%
138 \<latexrelease>                {\pushtabs}{Make commands robust}%

\pushtabs 139 \DeclareRobustCommand\pushtabs{%
140     \@stopfield\@addfield\global\advance\@tabpush \@ne \begingroup
141         \@contfield}

\poptabs It is, in some sense, an error if, after the endgroup, the current tab setting is higher
          than the new value of \@hightab (which is a local variable). That this is allowed is a
          fundamental design flaw which is not going to be corrected now.

142 \DeclareRobustCommand\poptabs{\@stopfield\@addfield
143     \ifnum \@tabpush >\z@
144         \endgroup
145         \global\advance\@tabpush \m@ne
146         \ifnum \@curtab >\@hightab
147             \global \@curtab \@hightab
148             \@badtab
149         \fi
150     \else
151         \@badpoptabs
152     \fi
153     \@contfield}

```

```

154 \DeclareRobustCommand\kill{\@stopfield\@startline\ignorespaces}
(End definition for \itabcr and others.)
155 </2ekernel | latexrelease>
156 <latexrelease>\EndIncludeInRelease
157 <latexrelease>\IncludeInRelease{0000/00/00}%
158 <latexrelease>          {\pushtabs}{Make commands robust}%
159 <latexrelease>
160 <latexrelease>\kernel@make@fragile\pushtabs
161 <latexrelease>\kernel@make@fragile\poptabs
162 <latexrelease>\kernel@make@fragile\kill
163 <latexrelease>
164 <latexrelease>\EndIncludeInRelease
165 <*2ekernel>

```

`\tabbingsep`

```

166 \newdimen\tabbingsep
(End definition for \tabbingsep.)

```

1.2 array and tabular environments

Historical L^AT_EX 2.09 comments (not necessarily accurate any more):

ARRAY PARAMETERS:

`\arraycolsep`
: half the width separating columns in an array environment

`\tabcolsep`
: half the width separating columns in a tabular environment

`\arrayrulewidth`
: width of rules

`\doublerulesep`
: space between adjacent rules in array or tabular

`\arraystretch`
: line spacing in array and tabular environments is done by placing a strut in every row of height and depth `\arraystretch` times the height and depth of the strut produced by an ordinary `\strut` command.

PREAMBLE:

The PREAMBLE argument of an array or tabular environment can contain the following:

`l,r,c` : indicate where entry is to be placed.
`|` : for vertical rule
`@{EXP}` : inserts the text EXP in every column.
`\arraycolsep` or `\tabcolsep` spacing is suppressed.
`*{N}{PRE}` : equivalent to writing N copies of PRE in the preamble.
PRE may contain `*{N'}{EXP'}` expressions.
`p{LEN}` : makes entry in parbox of width LEN.

SPECIAL ARRAY COMMANDS:

`\multicolumn{N}{FORMAT}{ITEM}` : replaces the next N column items by ITEM, formatted according to FORMAT. FORMAT should contain at most one l,r or c. If it contains none, then ITEM is ignored.

`\vline` : draws a vertical line the height of the current row. May appear in an array element entry.

`\hline` : draws a horizontal line between rows. Must appear either before the first entry (to appear above the first row) or right after a `\\` command. If followed by another `\hline`, then adds a `\vskip` of `\doublerulesep`.

`\cline{i-j}` : draws horizontal lines between rows covering columns i through j, inclusive. Multiple commands may follow one another to provide lines covering several disjoint columns

`\extracolsep{WIDTH}` : for use inside an @ in the preamble. Causes a WIDTH space to be added between columns for the rest of the columns. This is in addition to the ordinary intercolumn space.

```

\array ==
  BEGIN
    \@acol    == \@arrayacol
    \@classz  == \@arrayclassz
    \@classiv == \@arrayclassiv
    \\        == \@arraycr
    \@halignto == NULL
    \@tabarray
  END

\endarray{NAME} == BEGIN \crrc }} END

\tabular ==
  BEGIN
    \@halignto == NULL
    \@tabular
  END

\tabular*{WIDTH} ==
  BEGIN
    \@halignto == to WIDTH
    \@tabular
  END

\@tabular ==
  BEGIN
    \leavevmode
    \hbox { $
      \@acol    == \@tabacol

```

```

\@classz == \@tabclassz
\@classiv == \@tabclassiv
\\ == \@tabularcr
\@tabarray
END

\endtabular == BEGIN \crrc}} $} END

\@tabarray == if next char = [ then \@array else \@array[c] fi

\@array[POS]{PREAMBLE} ==
BEGIN
  define \@arstrutbox to make \@arstrut produce strut of height
    and depth \arraystretch times the height and
    depth of a normal strut.
  \@mkpream{PREAMBLE}
  \@preamble == \halign \@halignto {\tabskip=0pt\@arstrut
    eval{\@preamble}\tabskip = 0pt\cr %%}
  \@startpbox == \@@startpbox
  \@endpbox == \@@endpbox
  if POS = t then \vtop
    else if POS = b then \vbox
      else \vcenter
        fi
      fi
    {
      \par ==L {} % changed 92/09/18
      \@sharp == #
      \protect == \relax
      \lineskip :=L 0pt
      \baselineskip :=L 0pt
      \@preamble
    }
  END

\@arraycr ==
BEGIN
  $ %% Prevents extra space at end of row's last entry.
  if next char = [
    then \@argarraycr
    else $ \cr %% Needed to balance $
  END

\@argarraycr[LENGTH] ==
BEGIN
  $ %% Needed to balance $ of \@arraycr
  if LENGTH > 0
    then \@tempdima := depth of \@arstrutbox + LENGTH
      \vrule height 0pt width 0pt depth \@tempdima
      \cr
    else \cr \noalign{\vskip LENGTH}
  END

```

END

`\@tabularcr` and `\@argtabularcr` same as `\@arraycr` and `\@argarraycr`
except without the extra `$`'s.

End of historical L^AT_EX 2.09 comments.

`\extracolsep` This command needs to expand during the tabular preamble construction so can't be robust.

```
167 \def\extracolsep#1{\tabskip #1\relax}
```

(End definition for \extracolsep.)

`\array`

```
168 \def\array{\let\@acol\@arrayacol \let\@classz\@arrayclassz
```

```
169 \let\@classiv\@arrayclassiv
```

```
170 \let\\\@arraycr\let\@halignto\@empty\@tabarray}
```

(End definition for \array.)

`\endarray`

```
\endtabular 171 \def\endarray{\crcr\egroup\egroup}
```

```
\endtabular* 172 \def\endtabular{\crcr\egroup\egroup $\egroup}
```

```
173 \expandafter \let \csname endtabular*\endcsname = \endtabular
```

(End definition for \endarray, \endtabular, and \endtabular.)*

`\tabular`

```
174 \def\tabular{\let\@halignto\@empty\@tabular}
```

(End definition for \tabular.)

`\tabular*` Note that the change to use `\setlength` slightly alters the timing of the expansion and use of the length in `#1` but this is very unlikely to have any practical effect.

```
175 \@namedef\tabular*#1{%
```

```
176 \setlength\dimen@{#1}%
```

```
177 \edef\@halignto{to\the\dimen@}\@tabular}
```

(End definition for \tabular.)*

`\@tabular`

```
178 \def\@tabular{\leavevmode \hbox \bgroup $\let\@acol\@tabacol
```

```
179 \let\@classz\@tabclassz
```

```
180 \let\@classiv\@tabclassiv \let\\\@tabularcr\@tabarray}
```

(End definition for \@tabular.)

`\@tabarray` RmS 91/11/04 added `\m@th`.

```
181 \def\@tabarray{\m@th\@ifnextchar[\@array{\@array[c]}}
```

(End definition for \@tabarray.)

RmS 1993/11/03 changed `\halign` to `\ialign` and removed superfluous `\tabskip` assignment

`\@array`

```
182 \def\@array[#1]#2{%
```

```
183 \if #1t\vtop \else \if#1b\vbox \else \vcenter \fi\fi
```


184 **\bgroup**

This next bit of code sets up the strut and then builds the halign and its preamble according to the specification in the second argument.

This code has been moved inside the box. A side effect of this has been to expose what was a buglet in the previous version: since the `\@arstrut` below is expanded and contains an `\ifmmode` then it could produce an unnecessary extra box in every row, thus wasting ‘lots of’ main memory.

```
185    \setbox\@arstrutbox\hbox{%
186       \vrule \@height\arraystretch\ht\strutbox
187       \@depth\arraystretch \dp\strutbox
188       \@width\z@}%
189    \@mkpream{#2}%
190    \edef\@preamble{%
191       \ialign \noexpand\@halignto
192       \bgroup \@arstrut \@preamble \tabskip\z@skip \cr}%
```

That is the end of setting up the preamble; now we reset things before executing the halign built-up in `\@preamble`. The restorations could be done by introducing an extra group, thus saving tokens.

```
193    \let\@startpbox\@startpbox \let\@endpbox\@endpbox
194    \let\@tabularnewline\%
195    \let\par\@empty
196    \let\@sharp##%
197    \set@typeset@protect
198    \lineskip\z@skip\baselineskip\z@skip
```

If the parsing of the preamble goes wrong there may be some characters left which $\mathrm{T}_{\mathrm{E}}\mathrm{X}$ then tries to typeset, i.e., we would be in horizontal mode. That would produce an endless loop because the `\halign` expects vertical mode thus issues a `\par` but that is a no-op at this point. So we better test this case issue some error message and make a crude recovery by ending that horizontal mode with force. A better fix would be to ensure that we never pick up more than a single character token (not done).

```
199    \ifhmode \@preamerr\z@ \@@par\fi
200    \@preamble}
```

(End definition for \array.)

\@arraycr Array version of `\@array`.

```
201    \protected\def\@arraycr{%
202        ${\ifnum0=‘}\fi\@ifstar\@xarraycr\@xarraycr}
```

(End definition for \arraycr.)

\@arraycr

```
203    \def\@xarraycr{\@ifnextchar[\@argarraycr{\ifnum0=‘{\fi}{}\cr}}
```

(End definition for \arraycr.)

\@argarraycr

```
204    \def\@argarraycr[#1]{%
205       \ifnum0=‘{\fi}{}\ifdim #1>\z@ \@xargarraycr{#1}\else
206       \@yargarraycr{#1}\fi}
```

(End definition for \argarraycr.)

```

\tabularnewline Tabular version of \\.
207 \let\tabularnewline\relax

(End definition for \tabularnewline.)

\@tabularcr
208 \protected\def\@tabularcr{%
209   {\ifnum0='}\fi\@ifstar\@xtabularcr\@xtabularcr}

(End definition for \@tabularcr.)

\@xtabularcr
210 \def\@xtabularcr{\@ifnextchar[\@argtabularcr{\ifnum0='{ \fi}\cr}}

(End definition for \@xtabularcr.)

\@argtabularcr
211 \def\@argtabularcr[#1]{%
212   \ifnum0='{ \fi}%
213   \ifdim #1>\z@
214     \unskip\@xargarraycr{#1}%
215   \else
216     \@yargarraycr{#1}%
217   \fi}

(End definition for \@argtabularcr.)

\@xargarraycr
218 \def\@xargarraycr#1{\@tempdima #1\advance\@tempdima \dp \@arstrutbox
219   \vrule \@height\z@ \@depth\@tempdima \@width\z@ \cr}

(End definition for \@xargarraycr.)

\@yargarraycr
220 \</2ekernel>
221 \<*2ekernel | latexrelease>
222 \<latexrelease>\IncludeInRelease{2020/10/01}%
223 \<latexrelease>{\@yargarraycr}{tabular support calc syntax}%
224 \def\@yargarraycr#1{\cr\noalign{\@vspace@calcify{#1}}}%
225 \</2ekernel | latexrelease>
226 \<latexrelease>\EndIncludeInRelease
227 \<latexrelease>\IncludeInRelease{0000/00/00}%
228 \<latexrelease>{\@yargarraycr}{tabular support calc syntax}%
229 \<latexrelease>
230 \<latexrelease>\def\@yargarraycr#1{\cr\noalign{\vskip #1}}
231 \<latexrelease>\EndIncludeInRelease
232 \<*2ekernel>

(End definition for \@yargarraycr.)

```

`\multicolumn` *Historical L^AT_EX 2.09 comments (not necessarily accurate any more):*

```

\multicolumn{NUMBER}{FORMAT}{ITEM} ==
BEGIN
\multispan{NUMBER}
\beginngroup
\@addamp == null
\@mkpream{FORMAT}
\@sharp == ITEM
\protect == \relax
\@startpbox == \@startpbox
\@endpbox == \@endpbox
\@arstrut
\@preamble
\endgroup
END

```

End of historical L^AT_EX 2.09 comments.

The command `\def\@addamp{}` was removed from `\multicolumn` on 6 Dec 86 because it caused embedded array environments not to work. I think that it was included originally to prevent an error message if the 2nd argument to the `\multicolumn` command had two column specifiers.

8 Feb 89 — `\hbox{}` added after `\@preamble` to correct bug that occurred if `\multicolumn` preceded `\[D]` with $D > 0$, caused by `\[]` command doing an `\unskip`, which removed `\tabcolsep` glue inserted by `\multicolumn`.

This has been made long so that, for example, a `p`-column can contain multiple paragraphs; maybe the arguments of `@`-expressions should also be able to contain multiple paragraphs.

```

233 \long\def\multicolumn#1#2#3{\multispan{#1}\beginngroup
234   \@mkpream{#2}%
235   \def\@sharp{#3}\set@typeset@protect
236   \let\@startpbox\@startpbox\let\@endpbox\@endpbox
237   \@arstrut \@preamble\hbox{\endgroup\ignorespaces}

```

(End definition for \multicolumn.)

Historical L^AT_EX 2.09 comments (not necessarily accurate any more):

Codes for classes and character numbers of array, tabular and multicolumn arguments.

Character	Class	Number
c	0	0
l	0	1
r	0	2
	1	-
@	2	-
p	3	-
{@-exp}	4	-
{p-arg}	5	-

`\@testpach \foo` : expands `\foo`, which should be an array parameter

token, and sets \@chclass and \@chnum to its class and number. Uses \@lastchclass to distinguish 4 and 5

Preamble error codes

0: 'illegal character'
 1: 'Missing @-exp'
 2: 'Missing p-arg'

```
\@addamp ==
  BEGIN if \@firstamp = true then \@firstamp := false
        else &                                fi
  END
```

```
\@mkpream TOKENLIST ==
  BEGIN
    \@firstamp      := T
    \@lastchclass   := 6
    \@preamble      == null
    \@sharp         == \relax
    \@protect       == BEGIN \noexpand\protect\noexpand END
    \@startpbox     == \relax
    \@endpbox       == \relax
    \@expast{TOKENLIST}
    for \@nextchar := expand(\reserved@a)
      do \@testpach{\@nextchar}
        case of \@chclass
          0 -> \@classz
          1 -> \@classi
          ...
          5 -> \@classv
        end case
        \@lastchclass := \@chclass
      od
    case of \@lastchclass
      0 -> \hskip \arraycolsep           % lrc
      1 ->                                % l
      2 -> \@preamerr1 % 'Missing @-exp' % @
      3 -> \@preamerr2 % 'Missing p-arg' % p
      4 ->                                % @-exp
      5 -> \hskip \arraycolsep           % p-exp
    end case
  END
```

```
\@arrayclassz ==
  BEGIN
    \@preamble := \@preamble *
    case of \@lastchclass
      0 -> \hskip \arraycolsep \@addamp \hskip \arraycolsep
      1 -> \@addamp \hskip \arraycolsep
      2 -> % impossible
```

```

3 -> % impossible
4 -> \@addamp
5 -> \hskip \arraycolsep \@addamp \hskip \arraycolsep
6 -> \@addamp \hskip \arraycolsep
end case
* case of \@chnum
0 -> \hfil$\relax\@sharp$\hfil
1 -> $\relax\@sharp$\hfil
2 -> \hfil$\relax\@sharp$
end case

END

\@tabclassz == similar to \@arrayclassz

\@classi ==
BEGIN
  \@preamble := \@preamble *
  case of \@lastchclass
    0 -> \hskip \arraycolsep \@arrayrule
    1 -> \hskip \doublerulesep \@arrayrule
    2 -> % impossible
    3 -> % impossible
    4 -> \@arrayrule
    5 -> \hskip \arraycolsep \@arrayrule
    6 -> \@arrayrule
  end case

END

\@classii ==
BEGIN
  \@preamble := \@preamble *
  case of \@lastchclass
    0 ->
    1 -> \hskip .5\arrayrulewidth
    2 -> % impossible
    else ->
  end case

END

\@classiii ==
BEGIN
  \@preamble := \@preamble *
  case of \@lastchclass
    0 -> \hskip \arraycolsep \@addamp \hskip \arraycolsep
    1 -> \@addamp \hskip \arraycolsep
    2 -> % impossible
    3 -> % impossible
    4 -> \@addamp
    5 -> \hskip \arraycolsep \@addamp \hskip \arraycolsep
    6 -> \@addamp \hskip \arraycolsep

```

```

                                end case
END

\@arrayclassiv ==
    BEGIN \@preamble := \@preamble * $ \@nextchar$ END

\@tabclassiv == same as \@arrayclassv except without the $ ... $

\@classv ==
    BEGIN
        \@preamble :=
            \@preamble * \@startpbox{\@nextchar}\ignorespaces\@sharp
                                \@endpbox
    END

\@expast{S}:
    Sets \reserved@a := S with all instances of *{N}{STRING}
    replaced by N copies of STRING, where N > 0. An *
    appearing inside braces is ignored, but *-expressions
    inside STRING are expanded, so nested *-expressions are
    handled properly.

\@expast{S} == BEGIN \@expast S *0x\@@ END

\@expast S1 *{N}{S2} S3 \@@ ==
    BEGIN
        \reserved@a := S1
        \@tempcnta := N
        if \@tempcnta > 0
            then while \@tempcnta > 0 do \reserved@a := \reserved@a S2
                                \@tempcnta := \@tempcnta - 1 od
                \reserved@b == \@expast
            else \reserved@b == \@exnoop
            fi
        \expandafter \reserved@b \reserved@a S3 \@@
    END
End of historical LATEX 2.09 comments.

\@exnoop
238 \def\@exnoop #1\@@{}
(End definition for \@exnoop.)

\@expast
239 \def\@expast#1{\@expast #1*0x\@@}
(End definition for \@expast.)

```

```

\@xexpast
240 \def\@xexpast#1*#2#3#4\@@{%
241   \edef\reserved@a{#1}%
242   \@tempcnta#2\relax
243   \ifnum\@tempcnta>\z@
244     \@whilenum\@tempcnta>\z@do
245       {\edef\reserved@a{\reserved@a#3}\advance\@tempcnta \m@ne}%
246     \let\reserved@b\@xexpast
247   \else
248     \let\reserved@b\@xexnoop
249   \fi
250   \expandafter\reserved@b\reserved@a #4\@@}

(End definition for \@xexpast.)

\if@firstamp
\@addamp 251 \newif\if@firstamp
252 \def\@addamp{%
253   \if@firstamp
254     \@firstampfalse
255   \else
256     \edef\@preamble{\@preamble &}%
257   \fi}

(End definition for \if@firstamp and \@addamp.)

\@arrayacol
\@tabacol 258 \def\@arrayacol{\edef\@preamble{\@preamble \hskip \arraycolsep}}
\@ampacol 259 \def\@tabacol{\edef\@preamble{\@preamble \hskip \tabcolsep}}
\@acolampacol 260 \def\@ampacol{\@addamp \@acol}
261 \def\@acolampacol{\@acol\@addamp\@acol}

(End definition for \@arrayacol and others.)

\@mkpream
262 \def\@mkpream#1{\@firstamptrue\@lastchclass6
263   \let\@preamble\@empty
264   \let\protect\@unexpandable@protect
265   \let\@sharp\relax
266   \let\@startpbox\relax\let\@endpbox\relax
267   \@expast{#1}%
268   \expandafter\@tfor \expandafter
269     \@nextchar \expandafter:\expandafter=\reserved@a\do
270     {\@testpach\@nextchar
271       \ifcase \@chclass \@classz \or \@classi \or \@classii \or \@classiii
272       \or \@classiv \or \@classv \fi\@lastchclass\@chclass}%
273   \ifcase \@lastchclass \@acol
274     \or \or \@preamerr \@ne\or \@preamerr \tw@\or \or \@acol \fi}

(End definition for \@mkpream.)

```

`\@arrayclassz`

```
275 \def\@arrayclassz{\ifcase \@lastchclass \@acolampacol \or \@ampacol \or
276 \or \or \@addamp \or
277 \@acolampacol \or \@firstampfalse \@acol \fi
278 \edef\@preamble{\@preamble
279 \ifcase \@chnum
280 \hfil$\relax\@sharp$\hfil \or $\relax\@sharp$\hfil
281 \or \hfil$\relax\@sharp$\fi}}
```

(End definition for \@arrayclassz.)

`\@tabclassz` RmS 91/08/14 inserted extra braces around entry for NFSS

```
282 \def\@tabclassz{%
283 \ifcase\@lastchclass
284 \@acolampacol
285 \or
286 \@ampacol
287 \or
288 \or
289 \or
290 \@addamp
291 \or
292 \@acolampacol
293 \or
294 \@firstampfalse\@acol
295 \fi
296 \edef\@preamble{%
297 \@preamble{%
298 \ifcase\@chnum
299 \hfil
300 \hskip1sp%
301 \ignorespaces\@sharp\unskip\hfil
302 \or
303 \hskip1sp\ignorespaces\@sharp\unskip\hfil
304 \or
305 \hfil\hskip1sp\ignorespaces\@sharp\unskip
306 \fi}}}
```

(End definition for \@tabclassz.)

`\@classi`

```
307 \def\@classi{%
308 \ifcase\@lastchclass
309 \@acol\@arrayrule
310 \or
311 \@addtopreamble{\hskip \doublerulesep}\@arrayrule
312 \or
313 \or
314 \or
315 \@arrayrule
316 \or
317 \@acol\@arrayrule
318 \or
```



```

319     \@arrayrule
320     \fi}

(End definition for \@classi.)

\@classii

321 \def\@classii{%
322     \ifcase\@lastchclass
323     \or
324     \@addtopreamble{\hskip .5\arrayrulewidth}%
325     \fi}

(End definition for \@classii.)

\@classiii

326 \def\@classiii{\ifcase \@lastchclass \@acolampacol \or
327     \@addamp\@acol \or
328     \or \or \@addamp \or
329     \@acolampacol \or \@ampacol \fi}

(End definition for \@classiii.)

\@tabclassiv

330 \def\@tabclassiv{\@addtopreamble\@nextchar}

(End definition for \@tabclassiv.)

\@arrayclassiv

331 \def\@arrayclassiv{\@addtopreamble{$\@nextchar$}}

(End definition for \@arrayclassiv.)

\@classv

332 \def\@classv{\@addtopreamble{\@startpbox{\@nextchar}\ignorespaces
333     \@sharp\@endpbox}}

(End definition for \@classv.)

\@addtopreamble

334 \def\@addtopreamble#1{\edef\@preamble{\@preamble #1}}

(End definition for \@addtopreamble.)

\@chclass
\@lastchclass
\@chnum
335 \newcount\@chclass
336 \newcount\@lastchclass
337 \newcount\@chnum

(End definition for \@chclass, \@lastchclass, and \@chnum.)

\arraycolsep
\@tabcolsep
\arrayrulewidth
\@doublerulesep
338 \newdimen\arraycolsep
339 \newdimen\@tabcolsep
340 \newdimen\arrayrulewidth
341 \newdimen\@doublerulesep

```

(End definition for \arraycolsep and others.)

\arraystretch

```
342 \def\arraystretch{1} % Default value.
```

(End definition for \arraystretch.)

\@arstrutbox

\@arstrut

```
343 \newbox\@arstrutbox
```

```
344 \def\@arstrut{%
```

```
345 \relax\ifmmode\copy\@arstrutbox\else\unhcopy\@arstrutbox\fi}
```

(End definition for \@arstrutbox and \@arstrut.)

\@arrayrule

```
346 \def\@arrayrule{\@addtopreamble{\hskip -.5\arrayrulewidth
```

```
347 \vrule \@width \arrayrulewidth\hskip -.5\arrayrulewidth}}
```

(End definition for \@arrayrule.)

\@testpatch

```
348 \def\@testpach#1{\@chclass \ifnum \@lastchclass=\tw@ 4 \else
```

```
349 \ifnum \@lastchclass=3 5 \else
```

```
350 \z@ \if #1c\@chnum \z@ \else
```

```
351 \if #1l\@chnum \@ne \else
```

```
352 \if #1r\@chnum \tw@ \else
```

```
353 \@chclass \if #1|\@ne \else
```

```
354 \if #1@\tw@ \else
```

```
355 \if #1p3 \else \z@ \@preamerr 0\fi
```

```
356 \fi \fi \fi \fi \fi \fi
```

```
357 \fi}
```

(End definition for \@testpatch.)

\hline

```
358 \def\hline{%
```

```
359 \noalign{\ifnum0='}\fi\hrule \@height \arrayrulewidth \futurelet
```

```
360 \reserved@a\@xhline}
```

(End definition for \hline.)

\@xhline

```
361 \def\@xhline{\ifx\reserved@a\hline
```

```
362 \vskip\doublerulesep
```

Measure from the middle of the rules.

```
363 \vskip-\arrayrulewidth
```

```
364 \fi
```

```
365 \ifnum0='{ \fi}}
```

(End definition for \@xhline.)

\vline

```
366 \def\vline{\vrule \@width \arrayrulewidth}
```

(End definition for \vline.)

`\cline` The old L^AT_EX2.09 implementation of `\cline` used up quite a lot of memory and two precious count registers. This new (1995/09/14) implementation does not use any count registers. It is coded in a way that depends heavily on the definition of `\multispan` so that command has been moved here from the file `ltplain.dtx`.

These counters are no longer declared.

`\newcount\@cla`

`\newcount\@clb`

367 `\def\cline#1{\@cline#1\@nil}`

368 `\def\@cline#1-#2\@nil{%`

369 `\omit`

Use the counter from `\multispan`.

370 `\@multicnt#1%`

371 `\advance\@multispan\m@ne`

372 `\ifnum\@multicnt=\@ne\@firstofone{&\omit}\fi`

373 `\@multicnt#2%`

374 `\advance\@multicnt-#1%`

375 `\advance\@multispan\@ne`

The original had `\unskip` at this point, but how could a skip get here ???

376 `\leaders\hrule\@height\arrayrulewidth\hfill`

377 `\cr`

This is back spacing is fairly horrible, but it is what happened in the old version... An alternative would be to make `\cline` look ahead for a following `\cline` as does `\hline`. This would alter the spacing in existing documents so keep the old version in the kernel. Perhaps a package should do this differently.

378 `\noalign{\vskip-\arrayrulewidth}}`

(End definition for `\cline` and `\@cline`.)

`\mscount` The `\mscount` counter is no longer declared, saving a csname and a register. It is declared in compatibility mode.

(End definition for `\mscount`.)

`\multispan` Modify `\multispan` slightly from its plain T_EX definition to allow more efficient code sharing with `\multicolumn`. Also share a count register with `\multipt`.

`\@multispan` `\sp@n` 379 `\def\multispan{\omit\@multispan}`

380 `\def\@multispan#1{%`

381 `\@multicnt#1\relax`

382 `\loop\ifnum\@multicnt>\@ne \sp@n\repeat}`

383 `\def\sp@n{\span\omit\advance\@multicnt\m@ne}`

(End definition for `\multispan`, `\@multispan`, and `\sp@n`.)

`\@startpbox` Helper macros for ‘p’ columns.

`\@endpbox` `\@startpbox{<width>} text \egroup` is essentially `\parbox{<width>}{<text>}`

`\@endpbox` is essentially `\unskip \strut \par \egroup\hfil` (Changed 14 Jan 89)
(changed again 1994/05/13)

384 `\def\@startpbox#1{\vtop\bgroup \setlength\hsize{#1}\@arrayparboxrestore}`

```
385 \def\@endpbox{\@finalstrut\@arstrutbox\par\egroup\hfil}
```

14 Jan 89: Def of \@endpbox changed from
`\def\@endpbox{\par\vskip\dp\@arstrutbox\egroup\hfil}`
so vertical spacing works out right if the last line of a ‘p’ entry has a descender.

(End definition for \@startpbox and \@endpbox.)

```
\@@startpbox
```

```
\@@endpbox
```

```
386 \let\@@startpbox=\@startpbox
```

```
387 \let\@@endpbox=\@endpbox
```

(End definition for \@@startpbox and \@@endpbox.)

```
388 </2ekernel>
```

File L

ltpictur.dtx

1 Picture Mode

Picture mode commands. In addition to the commands available in L^AT_EX2.09, This section adds the new `\qbezier` command for drawing curves.

`\qbezier` `\qbezier[N](AX,AY)(BX,BY)(CX,CY)` plots a quadratic Bezier curve from (*AX,AY*) to (*CX,CY*), with (*BX,BY*) as the third Bezier point, using *N*+1 points equally spaced parametrically. If *N* = 0 (the default value), then a sufficient number of points are used to draw a connected curve—except that at most `\qbeziermax`+1 points are drawn. A “point” is a square of side `\@wholewidth`.

`\bezier` In addition, to be compatible with the old `bezier` package, a variant of this command, `\bezier`, is defined, in which the first argument is not optional.

Historical L^AT_EX 2.09 comments (not necessarily accurate any more):

<code>\unitlength</code>	= value of dimension argument
<code>\@wholewidth</code>	= current line width
<code>\@halfwidth</code>	= half of current line width
<code>\@linefnt</code>	= font for drawing lines
<code>\@circlefnt</code>	= font for drawing circles

`\linethickness{DIM}` : Sets the width of horizontal and vertical lines in a picture to DIM. Does not change width of slanted lines or circles. Width of all lines reset by `\thinlines` and `\thicklines`

```
\picture(XSIZE,YSIZE)(XORG,YORG)
  BEGIN
    \@picht :=L YSIZE * \unitlength
    box \@picbox :=
      \hb@xt@ XSIZE * \unitlength
      {\hskip -XORG * \unitlength
       \lower YORG * \unitlength
       \hbox{
         \ignorespaces      %% added 13 June 89
       }
    }
  END
```

```
\endpicture ==
  BEGIN
    } \hss }
    height of \@picbox := \@picht
    depth of \@picbox := 0
    \mbox{\box\@picbox}    %% change 26 Aug 91
  END
```

```
\put(X, Y){OBJ} ==
  BEGIN
```

```

\@killglue
\raise Y * \unitlength \hb@xt@ 0pt { \hskip X * \unitlength
                                OBJ \hss }

\ignorespaces
END

\multiput(X,Y)(DELX,DELY){N}{OBJ} ==
BEGIN
\@killglue
\@multicnt := N
\@xdim := X * \unitlength
\@ydim := Y * \unitlength
while \@multicnt > 0
do \raise \@ydim \hb@xt@ 0pt { \hskip \@xdim
                                OBJ \hss }

\@multicnt := \@multicnt - 1
\@xdim := \@xdim + DELX * \unitlength
\@ydim := \@ydim + DELY * \unitlength
od
\ignorespaces
END

\shortstack[POS]{TEXT} : Makes a \vbox containing TEXT stacked as
a one-column array, positioned l, r or c as indicated by POS.

```

End of historical L^AT_EX 2.09 comments.

The ‘2ekernel’ code ensures that a `\usepackage{autopict}` is essentially ignored if a ‘full’ format is being used that has picture mode already in the format.

```

1 \<2ekernel\>\expandafter\let\csname ver@autopict.sty\endcsname\fmtversion

\@wholewidth
\@halfwidth
2 \<*2ekernel\>
3 \newdimen\@wholewidth
4 \newdimen\@halfwidth

(End definition for \@wholewidth and \@halfwidth.)

\unitlength
5 \newdimen\unitlength \unitlength =1pt

(End definition for \unitlength.)

\@picbox
\@picht
6 \newbox\@picbox
7 \newdimen\@picht

(End definition for \@picbox and \@picht.)

```

`\@defaultunitsset` Set a length register, #1, accepting number or an etex length expression, #2, with default unit, #3.

The register name in #1 can be prefixed by `\advance` so that the register is incremented by the supplied value.

`\@defaultunitsset{\advance\@vxx}{\textwidth-15pt}\unitlength`

#3 can be a literal unit such as `cm` or a length register such as `\unitlength`.

This is used in all `picture` commands that take picture coordinates. So `\put(2,2)` as previously but now `\put(\textwidth-5cm,0.4\textheight)` Note that you can only use expressions with lengths, `\put(1+2,0)` is not supported.

```

8 </2ekernel>
9 <*2ekernel | latexrelease>
10 <latexrelease>\IncludeInRelease{2020/10/01}%
11 <latexrelease>          {\@defaultunitsset}{default units}%
12 \def\@defaultunitsset#1#2#3{%
13   \@defaultunits#1\dimexpr#2#3\relax\relax\@nnil}
14 </2ekernel | latexrelease>

15 <latexrelease>\EndIncludeInRelease
16 <latexrelease>\IncludeInRelease{0000/00/00}%
17 <latexrelease>          {\@defaultunitsset}{default units}%
18 <latexrelease>\let\@defaultunitsset\undefined
19 <latexrelease>\EndIncludeInRelease
20 <*2ekernel>

```

(End definition for \@defaultunitsset.)

`\picture` #1 should be white space.

#1 should be a ((eating any white space before the bracket),

```

\pictur@ 21 \long\def\picture#1{\pictur@#1}
22 \def\pictur@(#1){%
23   \ifnextchar({\@picture(#1)}{\@picture(#1)(0,0)}}

```

(End definition for \picture and \pictur@.)

`\@picture`

```

24 </2ekernel>
25 <*2ekernel | latexrelease>
26 <latexrelease>\IncludeInRelease{2020/10/01}%
27 <latexrelease>          {\@picture}{default units}%
28 \def\@picture(#1,#2)(#3,#4){%
29   \@defaultunitsset\@picht{#2}\unitlength
30   \@defaultunitsset\@tempdimc{#1}\unitlength
31   \setbox\@picbox\hbext@\@tempdimc\bgroup
32     \@defaultunitsset\@tempdimc{#3}\unitlength
33     \hskip -\@tempdimc
34     \@defaultunitsset\@tempdimc{#4}\unitlength
35     \lower\@tempdimc\hbox\bgroup
36     \ignorespaces}
37 </2ekernel | latexrelease>

```

```

38 <latexrelease>\EndIncludeInRelease
39 <latexrelease>\IncludeInRelease{0000/00/00}%
40 <latexrelease>          {\@picture}{default units}%
41 <latexrelease>\def\@picture(#1,#2)(#3,#4){%
42 <latexrelease>  \picht#2\unitlength
43 <latexrelease>  \setbox\@picbox\hb@xt@#1\unitlength\bgroup
44 <latexrelease>    \hskip -#3\unitlength
45 <latexrelease>    \lower #4\unitlength\hbox\bgroup
46 <latexrelease>      \ignorespaces}
47 <latexrelease>\EndIncludeInRelease
48 <*2ekernel>

```

(End definition for \@picture.)

\endpicture

```

49 \def\endpicture{%
50   \egroup\hss\egroup
51   \ht\@picbox\@picht\dp\@picbox\z@
52   \mbox{\box\@picbox}}

```

(End definition for \endpicture.)

In the definitions of \put and \multiput, \hskip was replaced by \kern just in case arg #3 = “plus”. (Bug detected by Don Knuth. changed 20 Jul 87).

```

53 </2ekernel>
54 <*2ekernel | latexrelease>
55 <latexrelease>\IncludeInRelease{2020/10/01}%
56 <latexrelease>          {\put}{default units}%
57 <latexrelease>\expandafter\let\csname put \endcsname \@undefined
58 \long\def\put(#1,#2)#3{%
59   \@killglue
60   \@defaultunitsset\@tempdimc{#2}\unitlength
61   \raise\@tempdimc
62   \hb@xt@\z@{%
63     \@defaultunitsset\@tempdimc{#1}\unitlength
64     \kern\@tempdimc
65     #3\hss}%
66   \ignorespaces}
67 </2ekernel | latexrelease>
68 <latexrelease>\EndIncludeInRelease
69 <latexrelease>\IncludeInRelease{0000/00/00}%
70 <latexrelease>          {\put}{default units}%
71 <latexrelease>\expandafter\let\csname put \endcsname \@undefined
72 <latexrelease>\long\def\put(#1,#2)#3{%
73 <latexrelease>  \@killglue\raise#2\unitlength
74 <latexrelease>  \hb@xt@\z@{\kern#1\unitlength #3\hss}%
75 <latexrelease>  \ignorespaces}
76 <latexrelease>\EndIncludeInRelease
77 <*2ekernel>

```

\multiput #3 had better be a (.

```

78 </2ekernel>
79 <*2ekernel | latexrelease>
80 <latexrelease>\IncludeInRelease{2020/10/01}%

```



```

81 <latexrelease>                {\multiput}{default units}%
82 <latexrelease>\expandafter\let\csname multiput \endcsname \@undefined
83 \def\multiput(#1,#2)#3{%
84   \@defaultunitsset\@xdim{#1}\unitlength
85   \@defaultunitsset\@ydim{#2}\unitlength
86   \@multiput{ }
87 </2ekernel | latexrelease>

88 <latexrelease>\EndIncludeInRelease
89 <latexrelease>\IncludeInRelease{0000/00/00}%
90 <latexrelease>                {\multiput}{default units}%
91 <latexrelease>\expandafter\let\csname multiput \endcsname \@undefined
92 <latexrelease>\def\multiput(#1,#2)#3{%
93 <latexrelease>   \@xdim #1\unitlength
94 <latexrelease>   \@ydim #2\unitlength
95 <latexrelease>   \@multiput{ }
96 <latexrelease>\EndIncludeInRelease
97 <*2ekernel>

```

(End definition for \multiput.)

\@multiput

```

98 </2ekernel>
99 <*2ekernel | latexrelease>
100 <latexrelease>\IncludeInRelease{2020/10/01}%
101 <latexrelease>                {\@multiput}{default units}%
102 \long\def\@multiput(#1,#2)#3#4{%
103   \@killglue\@multicnt #3\relax
104   \@whilenum \@multicnt >\z@ \do
105     {\raise\@ydim\hb@xt@{\z@{\kern\@xdim #4\hss}}%
106      \advance\@multicnt\m@ne
107      \@defaultunitsset{\advance\@xdim}{#1}\unitlength
108      \@defaultunitsset{\advance\@ydim}{#2}\unitlength}%
109   \ignorespaces}
110 </2ekernel | latexrelease>

111 <latexrelease>\EndIncludeInRelease
112 <latexrelease>\IncludeInRelease{0000/00/00}%
113 <latexrelease>                {\@multiput}{default units}%
114 <latexrelease>\long\def\@multiput(#1,#2)#3#4{%
115 <latexrelease>   \@killglue\@multicnt #3\relax
116 <latexrelease>   \@whilenum \@multicnt >\z@ \do
117 <latexrelease>     {\raise\@ydim\hb@xt@{\z@{\kern\@xdim #4\hss}}%
118 <latexrelease>      \advance\@multicnt\m@ne
119 <latexrelease>      \advance\@xdim#1\unitlength\advance\@ydim#2\unitlength}%
120 <latexrelease>   \ignorespaces}
121 <latexrelease>\EndIncludeInRelease
122 <*2ekernel>

```

(End definition for \@multiput.)

\@killglue

```

123 \def\@killglue{\unskip\@whiledim \lastskip >\z@\do{\unskip}}

```

(End definition for \@killglue.)

```

\thinlines
\thicklines 124 \DeclareRobustCommand\thinlines{\let\@linefnt\tenln
125 \let\@circlefnt\tencirc
126 \@wholewidth\fontdimen8\tenln \@halfwidth .5\@wholewidth}
127 \DeclareRobustCommand\thicklines{\let\@linefnt\tenlnw
128 \let\@circlefnt\tencircw
129 \@wholewidth\fontdimen8\tenlnw \@halfwidth .5\@wholewidth}

(End definition for \thinlines and \thicklines.)

\linethickness

130 \DeclareRobustCommand*\linethickness[1]
131 {\@wholewidth #1\relax \@halfwidth .5\@wholewidth \ignorespaces}

(End definition for \linethickness.)

\ishortstack

132 \def\shortstack{\@ifnextchar[\@shortstack{\@shortstack[c]}}

(End definition for \ishortstack.)

\@ishortstack

133 \def\@shortstack[#1]{%
134 \leavevmode
135 \vbox\bgroup
136 \baselineskip-\p@\lineskip 3\p@
137 \let\mb@l\hss\let\mb@r\hss
138 \expandafter\let\csname mb@#1\endcsname\relax
139 \let\\\@stackcr
140 \@ishortstack}

(End definition for \@ishortstack.)

\@ishortstack

141 \def\@ishortstack#1{\ialign{\mb@l {##}\unskip\mb@r\cr #1\cr}\egroup}

(End definition for \@ishortstack.)

\@stackcr
\@ixstackcr

142 \protected\def\@stackcr{\@ifstar\@ixstackcr\@ixstackcr}
143 \def\@ixstackcr{\@ifnextchar[\@istackcr{\cr\ignorespaces}}

(End definition for \@stackcr and \@ixstackcr.)

\@istackcr

144 </2ekernel>
145 <*2ekernel | latexrelease>
146 <latexrelease>\IncludeInRelease{2020/10/01}%
147 <latexrelease> {\@istackcr}{\shortstack calc support}%
148 \def\@istackcr[#1]{\cr\noalign{\@vspace@calcify{#1}}\ignorespaces}
149 </2ekernel | latexrelease>

```

```

150 <latexrelease>\EndIncludeInRelease
151 <latexrelease>\IncludeInRelease{0000/00/00}%
152 <latexrelease>{\@istackcr}{\shortstack calc support}%
153 <latexrelease>
154 <latexrelease>\def\@istackcr[#1]{\cr\noalign{\vskip #1}\ignorespaces}
155 <latexrelease>\EndIncludeInRelease
156 <*2ekernel>

(End definition for \@istackcr.)
Historical LATEX 2.09 comments (not necessarily accurate any more):
\line(X,Y){LEN} ==
BEGIN
  \@xarg := X
  \@yarg := Y
  \@linelen := LEN * \unitlength
  if \@xarg = 0
    then \@vline
    else if \@yarg = 0
      then \@hline
      else \@sline
    if
  if
END

\@sline ==
BEGIN
  if \@xarg < 0
    then @negarg := T
      \@xarg := -\@xarg
      \@yyarg := -\@yarg
    else @negarg := F
      \@yyarg := \@yarg
  fi
  \@tempcnta := |\@yyarg|
  if \@tempcnta > 6
    then error: 'LATEX ERROR: Illegal \line or \vector argument.'
      \@tempcnta := 0
  fi
  \box\@linechar := \hbox{\@linefont \@getlinechar(\@xarg,\@yyarg) }
  if \@yarg > 0 then \@upordown = \raise
    \@clnht := 0
  else \@upordown = \lower
    \@clnht := height of \box\@linechar
  fi
  \@clnwd := width of \box\@linechar
  if @negarg
    then \hskip - width of \box\@linechar
      \reserved@a == \hskip - 2* width of box \@linechar
    else \reserved@a == \relax
  fi
  %% Put out integral number of line segments

```

```

while \@clnwd < \@linelen
do
  \@upordown \@clnht \copy\@linechar
  \reserved@a
  \@clnht := \@clnht + ht of \box\@linechar
  \@clnwd := \@clnwd + width of \box\@linechar
od

%% Put out last segment
\@clnht := \@clnht - height of \box\@linechar
\@clnwd := \@clnwd - width of \box\@linechar
\@tempdima := \@linelen - \@clnwd
\@tempdimb := \@tempdima - width of \box\@linechar
if @negarg then \hskip -\@tempdimb
else \hskip \@tempdimb
fi
\@tempdima := 1000 * \@tempdima
\@tempcnta := \@tempdima / width of \box\@linechar
\@tempdima := (\@tempcnta * ht of \box\@linechar)/1000
\@clnht := \@clnht + \@tempdima
if \@linelen < width of \box\@linechar
then \hskip width of \box\@linechar
else \hbox{\@upordown \@clnht \copy\@linechar}
fi
END

\@hline ==
BEGIN
if \@xarg < 0 then \hskip -\@linelen \fi
\vrule height \@halfwidth depth \@halfwidth width \@linelen
if \@xarg < 0 then \hskip -\@linelen \fi
END

\@vline == if \@yarg < 0 \@downline else \@upline fi

\@getlinechar(X,Y) ==
BEGIN
\@tempcnta := 8*X - 9
if Y > 0
then \@tempcnta := \@tempcnta + Y
else \@tempcnta := \@tempcnta - Y + 64
fi
\char\@tempcnta
END

\vector(X,Y){LEN} ==
BEGIN
\@xarg := X
\@yarg := Y
\@linelen := LEN * \unitlength

```

```

if \@xarg = 0
  then \@vvector
  else if \@yarg = 0
    then \@hvector
    else \@svector
  if
if
END

\@hvector ==
BEGIN
  \@hline
  {\@linefnt if \@xarg < 0 then \@getlarrow(1,0)
    else \@getrarrow(1,0)
  fi}
END

\@vvector == if \@yarg < 0 \@downvector else \@upvector fi

\@svector ==
BEGIN
  \@sline
  \@tempcnta := | \@yarg |
  if \@tempcnta < 5
    then \hskip - width of \box\@linechar
      \@upordown \@clnht \hbox
        {\@linefnt
          if @negarg then \@getlarrow(\@xarg,\@yyarg)
            else \@getrarrow(\@xarg,\@yyarg)
          fi }
    else error: 'LATEX ERROR: Illegal \line or \vector argument.'
  fi
END

\@getlarrow(X,Y) ==
BEGIN
  if Y = 0
    then \@tempcnta := '33
  else \@tempcnta := 16 * X - 9
    \@tempcntb := 2 * Y
    if \@tempcntb > 0
      then \@tempcnta := \@tempcnta + \@tempcntb
      else \@tempcnta := \@tempcnta - \@tempcntb + 64
    fi
  fi
  \char\@tempcnta
END

\@getrarrow(X,Y) ==
BEGIN

```

```

\@tempcntb := |Y|
case of \@tempcntb
  0 : \@tempcnta := '55
  1 : if X < 3
      then \@tempcnta := 24*X - 6
      else if X = 3
          then \@tempcnta := 49
          else \@tempcnta := 58 fi
      fi
  2 : if X < 3
      then \@tempcnta := 24*X - 3
      else \@tempcnta := 51 % X must = 3
      fi
  3 : \@tempcnta := 16*X - 2
  4 : \@tempcnta := 16*X + 7
endcase
if Y < 0
  then \@tempcnta := \@tempcnta + 64
  fi
\char\@tempcnta
END

```

End of historical L^AT_EX 2.09 comments.

`\if@negarg`

```

157 \newif\if@negarg

```

(End definition for \if@negarg.)

`\line`

```

158 </2ekernel>
159 <*2ekernel | latexrelease>
160 <latexrelease>\IncludeInRelease{2020/10/01}%
161 <latexrelease>          {\line}{default units}%
162 <latexrelease>\expandafter\let\csname line \endcsname\@undefined
163 \def\line(#1,#2)#3{\@xarg #1\relax \@yarg #2\relax
164   \@defaultunitsset\@linelen{#3}\unitlength
165   \ifdim\@linelen<\z@\@badlinearg\else
166     \ifnum\@xarg =\z@ \@vline
167     \else \ifnum\@yarg =\z@ \@hline \else \@sline\fi
168   \fi
169   \fi}
170 </2ekernel | latexrelease>

171 <latexrelease>\EndIncludeInRelease
172 <latexrelease>\IncludeInRelease{0000/00/00}%
173 <latexrelease>          {\line}{default units}%
174 <latexrelease>\expandafter\let\csname line \endcsname\@undefined
175 <latexrelease>\def\line(#1,#2)#3{\@xarg #1\relax \@yarg #2\relax
176 <latexrelease>  \@linelen #3\unitlength
177 <latexrelease>  \ifdim\@linelen<\z@\@badlinearg\else
178 <latexrelease>    \ifnum\@xarg =\z@ \@vline
179 <latexrelease>    \else \ifnum\@yarg =\z@ \@hline \else \@sline\fi
180 <latexrelease>    \fi

```

```

181 <latexrelease> \fi}
182 <latexrelease>\EndIncludeInRelease
183 <*2kernel>

```

(End definition for \line.)

\@sline

```

184 \def\@sline{%
185   \ifnum\@xarg<\z@ \@negargtrue \@xarg -\@xarg \@yyarg -\@yarg
186   \else \@negargfalse \@yyarg \@yarg \fi
187   \ifnum \@yyarg >\z@ \@tempcnta\@yyarg \else \@tempcnta -\@yyarg \fi
188   \ifnum\@tempcnta>6 \@badlinearg\@tempcnta\z@ \fi
189   \ifnum\@xarg>6 \@badlinearg\@xarg \@ne \fi
190   \setbox\@linechar\hbox{\@linefnt\@getlinechar(\@xarg,\@yyarg)}%

```

If we have something like \line(5,5){30} the \@linechar will not contain a char and later on we will end in an infinite loop. So we check the width of the box and put in something as an emergency fix if necessary.

```

191 \ifdim\wd\@linechar=\z@
192   \setbox\@linechar\hbox{.}%
193   \@badlinearg
194 \fi
195 \ifnum \@yarg >\z@ \let\@upordown\raise \@clnht\z@
196   \else\let\@upordown\lower \@clnht \ht\@linechar\fi
197 \@clnwd \wd\@linechar
198 \if@negarg
199   \hskip -\wd\@linechar \def\reserved@a{\hskip -2\wd\@linechar}%
200 \else
201   \let\reserved@a\relax
202 \fi
203 \@whiledim \@clnwd <\@linelen \do
204   {\@upordown\@clnht\copy\@linechar
205     \reserved@a
206     \advance\@clnht \ht\@linechar
207     \advance\@clnwd \wd\@linechar}%
208 \advance\@clnht -\ht\@linechar
209 \advance\@clnwd -\wd\@linechar
210 \@tempdima\@linelen\advance\@tempdima -\@clnwd
211 \@tempdimb\@tempdima\advance\@tempdimb -\wd\@linechar
212 \if@negarg \hskip -\@tempdimb \else \hskip \@tempdimb \fi
213 \multiply\@tempdima \@m
214 \@tempcnta \@tempdima
215 \@tempdima \wd\@linechar \divide\@tempcnta \@tempdima
216 \@tempdima \ht\@linechar \multiply\@tempdima \@tempcnta
217 \divide\@tempdima \@m
218 \advance\@clnht \@tempdima
219 \ifdim \@linelen <\wd\@linechar
220   \hskip \wd\@linechar

```

Warn if line gets so short that it can't be printed. But don't warn if it is exactly zero since that was probably deliberate (e.g., to get a vector head only).

```

221 \ifdim \@linelen = \z@
222 \else
223   \@picture@warn
224 \fi

```

```
225 \else\@upordown\@clnht\copy\@linechar\fi}
```

(End definition for \@sline.)

\@hline

```
226 \def\@hline{\ifnum \@xarg <\z@ \hskip -\@linelen \fi
227 \vrule \@height \@halfwidth \@depth \@halfwidth \@width \@linelen
228 \ifnum \@xarg <\z@ \hskip -\@linelen \fi}
```

(End definition for \@hline.)

\@getlinechar

```
229 \def\@getlinechar(#1,#2){\@tempcnta#1\relax\multiply\@tempcnta 8%
230 \advance\@tempcnta -9\ifnum #2>\z@ \advance\@tempcnta #2\relax\else
231 \advance\@tempcnta -#2\relax\advance\@tempcnta 64 \fi
232 \char\@tempcnta}
```

(End definition for \@getlinechar.)

\vector

```
233 </2ekernel>
234 <*2ekernel | latexrelease>
235 <latexrelease>\IncludeInRelease{2020/10/01}%
236 <latexrelease> \{\vector\}{default units}%
237 <latexrelease>\expandafter\let\csname vector \endcsname\@undefind
238 \def\vector(#1,#2)#3{\@xarg #1\relax \@yarg #2\relax
239 \@tempcnta \ifnum\@xarg<\z@ -\@xarg\else\@xarg\fi
240 \ifnum\@tempcnta<5\relax
241 \@defaultunitsset\@linelen{#3}\unitlength
242 \ifdim\@linelen<\z@\@badlinearg\else
243 \ifnum\@xarg =\z@ \@vvector
244 \else \ifnum\@yarg =\z@ \@hvector \else \@svector\fi
245 \fi
246 \fi
247 \else\@badlinearg\fi}
248 </2ekernel | latexrelease>
249 <latexrelease>\EndIncludeInRelease
250 <latexrelease>\IncludeInRelease{0000/00/00}%
251 <latexrelease> \{\vector\}{default units}%
252 <latexrelease>\expandafter\let\csname vector \endcsname\@undefind
253 <latexrelease>\def\vector(#1,#2)#3{\@xarg #1\relax \@yarg #2\relax
254 <latexrelease> \@tempcnta \ifnum\@xarg<\z@ -\@xarg\else\@xarg\fi
255 <latexrelease> \ifnum\@tempcnta<5\relax
256 <latexrelease> \@linelen #3\unitlength
257 <latexrelease> \ifdim\@linelen<\z@\@badlinearg\else
258 <latexrelease> \ifnum\@xarg =\z@ \@vvector
259 <latexrelease> \else \ifnum\@yarg =\z@ \@hvector \else \@svector\fi
260 <latexrelease> \fi
261 <latexrelease> \fi
262 <latexrelease> \else\@badlinearg\fi}
263 <latexrelease>\EndIncludeInRelease
264 <*2ekernel>
```

(End definition for \vector.)


```

\@hvector
265 \def\@hvector{\@hline\hb@xt@\z@\@linefnt
266 \ifnum \@xarg <\z@ \@getlarrow(1,0)\hss\else
267 \hss\@getrarrow(1,0)\fi}}

(End definition for \@hvector.)

\@vvector
268 \def\@vvector{\ifnum \@yarg <\z@ \@downvector \else \@upvector \fi}

(End definition for \@vvector.)

\@svector
269 \def\@svector{\@sline
270 \@tempcnta\@yarg \ifnum\@tempcnta <\z@ \@tempcnta -\@tempcnta\fi
271 \ifnum\@tempcnta <5%
272 \hskip -\wd\@linechar
273 \@upordown\@clnht \hbox{\@linefnt \if@negarg
274 \@getlarrow(\@xarg,\@yyarg)\else \@getrarrow(\@xarg,\@yyarg)\fi}%
275 \else\@badlinearg\fi}

(End definition for \@svector.)

\@getlarrow
276 \def\@getlarrow(#1,#2){\ifnum #2=\z@ \@tempcnta 27 % '33
277 \else
278 \@tempcnta #1\relax\multiply\@tempcnta \sixt@@n
279 \advance\@tempcnta -9 \@tempcntb #2\relax\multiply\@tempcntb \tw@
280 \ifnum \@tempcntb >\z@ \advance\@tempcnta \@tempcntb
281 \else\advance\@tempcnta -\@tempcntb\advance\@tempcnta 64
282 \fi\fi\char\@tempcnta}

(End definition for \@getlarrow.)

\@getrarrow
283 \def\@getrarrow(#1,#2){\@tempcntb #2\relax
284 \ifnum\@tempcntb <\z@ \@tempcntb -\@tempcntb\relax\fi
285 \ifcase \@tempcntb\relax \@tempcnta 45 % '55
286 \or
287 \ifnum #1<\thr@@ \@tempcnta #1\relax\multiply\@tempcnta
288 24\advance\@tempcnta -6 \else \ifnum #1=\thr@@ \@tempcnta 49
289 \else\@tempcnta 58 \fi\fi\or
290 \ifnum #1<\thr@@ \@tempcnta=#1\relax\multiply\@tempcnta
291 24\advance\@tempcnta -\thr@@ \else \@tempcnta 51 \fi\or
292 \@tempcnta #1\relax\multiply\@tempcnta
293 \sixt@@n \advance\@tempcnta -\tw@ \else
294 \@tempcnta #1\relax\multiply\@tempcnta
295 \sixt@@n \advance\@tempcnta 7 \fi\ifnum #2<\z@ \advance\@tempcnta 64 \fi
296 \char\@tempcnta}

(End definition for \@getrarrow.)

\@vline
297 \def\@vline{\ifnum \@yarg <\z@ \@downline \else \@upline\fi}

```

(End definition for \vline.)

\@upline

```
298 \def\@upline{%
299   \hb@xt@\z@{\hskip -\@halfwidth \vrule \@width \@wholewidth
300   \@height \@linelen \@depth \z@\hss}}
```

(End definition for \@upline.)

\@downline

```
301 \def\@downline{%
302   \hb@xt@\z@{\hskip -\@halfwidth \vrule \@width \@wholewidth
303   \@height \z@ \@depth \@linelen \hss}}
```

(End definition for \@downline.)

\@upvector

```
304 \def\@upvector{\@upline\setbox\@tempboxa\hbox{\@linefnt\char 54}% '66
305   \raise \@linelen \hb@xt@\z@{\lower \ht\@tempboxa\box\@tempboxa\hss}}
```

(End definition for \@upvector.)

\@downvector

```
306 \def\@downvector{\@downline\lower \@linelen
307   \hb@xt@\z@{\@linefnt\char 63 % '77
308   \hss}}
```

(End definition for \@downvector.)

Historical L^AT_EX 2.09 comments (not necessarily accurate any more):

```
\dashbox{D}(X,Y) ==
BEGIN
leave vertical mode
\hb@xt@ Opt {
  \baselineskip := Opt
  \lineskip      := Opt
%% HORIZONTAL DASHES
  \@dashdim := X * \unitlength
  \@dashcnt := \@dashdim + 200 % to prevent roundoff error
  \@dashdim := D * \unitlength
  \@dashcnt := \@dashcnt / \@dashdim
  if \@dashcnt is odd
  then \@dashdim := Opt
      \@dashcnt := (\@dashcnt + 1) / 2
  else \@dashdim := \@dashdim / 2
      \@dashcnt := \@dashcnt / 2 - 1
      \box\@dashbox := \hbox{\vrule height \@halfwidth
                             depth \@halfwidth width \@dashdim}
      \put(0,0){\copy\@dashbox}
      \put(0,Y){\copy\@dashbox}
      \put(X,0){\hskip -\@dashdim\copy\@dashbox}
      \put(X,Y){\hskip -\@dashdim\box\@dashbox}
  \@dashdim := 3 * \@dashdim
fi
```

```

\box\@dashbox := \hbox{\vrule height \@halfwidth
                        depth \@halfwidth width D * \unitlength
                        \hskip D * \unitlength}

\@tempcnta := 0
\put(0,0){\hskip \@dashdim
          while \@tempcnta < \@dashcnt
            do \copy\@dashbox
              \@tempcnta := \@tempcnta + 1
            od
          }
\@tempcnta := 0
\put(0,Y){\hskip \@dashdim
          while \@tempcnta < \@dashcnt
            do \copy\@dashbox
              \@tempcnta := \@tempcnta + 1
            od
          }

%% vertical dashes
\@dashdim := Y * \unitlength
\@dashcnt := \@dashdim + 200 % to prevent roundoff error
\@dashdim := D * \unitlength
\@dashcnt := \@dashcnt / \@dashdim
if \@dashcnt is odd
then \@dashdim := 0pt
    \@dashcnt := (\@dashcnt + 1) / 2
else \@dashdim := \@dashdim / 2
    \@dashcnt := \@dashcnt / 2 - 1
    \box\@dashbox := \hbox{\hskip -\@halfwidth
                          \vrule width \@wholewidth
                          height \@dashdim }

    \put(0,0){\copy\@dashbox}
    \put(X,0){\copy\@dashbox}
    \put(0,Y){\lower\@dashdim\copy\@dashbox}
    \put(X,Y){\lower\@dashdim\copy\@dashbox}
    \@dashdim := 3 * \@dashdim
fi
\box\@dashbox := \hbox{\vrule width \@wholewidth
                        height D * \unitlength      }

\@tempcnta := 0
\put(0,0){\hskip -\halfwidth
          \vbox{while \@tempcnta < \@dashcnt
                do \vskip D*\unitlength
                  \copy\@dashbox
                  \@tempcnta := \@tempcnta + 1
                od
                \vskip \@dashdim
              } }
\@tempcnta := 0
\put(X,0){\hskip -\halfwidth

```

```

        \vbox{while \@tempcnta < \@dashcnt
            do \vskip D*\unitlength
              \copy\@dashbox
              \@tempcnta := \@tempcnta + 1
            od
            \vskip \@dashdim
          }
    }
}      % END DASHES

```

\@makepicbox(X,Y)

END

End of historical L^AT_EX 2.09 comments.

\dashbox

```

309 \</2kernel>
310 \<*2kernel | latexrelease>
311 \<latexrelease>\IncludeInRelease{2020/10/01}%
312 \<latexrelease>{\dashbox}{default units}%
313 \<latexrelease>\expandafter\let\csname dashbox \endcsname\undefind
314 \def\dashbox#1(#2,#3){\leavevmode\hb@xt@\z@{\baselineskip \z@skip
315 \lineskip \z@skip
316 \@defaultunitsset\@dashdim{#2}\unitlength
317 \@dashcnt \@dashdim \advance\@dashcnt 200
318 \@defaultunitsset\@dashdim{#1}\unitlength
319 \divide\@dashcnt \@dashdim
320 \ifodd\@dashcnt\@dashdim \z@
321 \advance\@dashcnt \@one \divide\@dashcnt \tw@
322 \else \divide\@dashdim \tw@ \divide\@dashcnt \tw@
323 \advance\@dashcnt \m@ne
324 \setbox\@dashbox \hbox{\vrule \@height \@halfwidth \@depth \@halfwidth
325 \@width \@dashdim}\put(0,0){\copy\@dashbox}%
326 \put(0,#3){\copy\@dashbox}%
327 \put(#2,0){\hskip-\@dashdim\copy\@dashbox}%
328 \put(#2,#3){\hskip-\@dashdim\box\@dashbox}%
329 \multiply\@dashdim \thr@@
330 \fi
331 \setbox\@dashbox \hbox{%
332   \@defaultunitsset\@tempdimc{#1}\unitlength
333   \vrule \@height \@halfwidth \@depth \@halfwidth \@width \@tempdimc
334   \hskip\@tempdimc}%
335 \@tempcnta\z@
336 \put(0,0){\hskip\@dashdim \@whilenum \@tempcnta <\@dashcnt
337 \do{\copy\@dashbox\advance\@tempcnta \@one }}\@tempcnta\z@
338 \put(0,#3){\hskip\@dashdim \@whilenum \@tempcnta <\@dashcnt
339 \do{\copy\@dashbox\advance\@tempcnta \@one }}%
340 \@defaultunitsset\@dashdim{#3}\unitlength
341 \@dashcnt \@dashdim \advance\@dashcnt 200
342 \@defaultunitsset\@dashdim{#1}\unitlength
343 \divide\@dashcnt \@dashdim
344 \ifodd\@dashcnt \@dashdim \z@
345 \advance\@dashcnt \@one \divide\@dashcnt \tw@
346 \else

```

```

347 \divide\@dashdim \tw@ \divide\@dashcnt \tw@
348 \advance\@dashcnt \m@ne
349 \setbox\@dashbox\hbox{\hskip -\@halfwidth
350 \vrule \@width \@wholewidth
351 \@height \@dashdim}\put(0,0){\copy\@dashbox}%
352 \put(#2,0){\copy\@dashbox}%
353 \put(0,#3){\lower\@dashdim\copy\@dashbox}%
354 \put(#2,#3){\lower\@dashdim\copy\@dashbox}%
355 \multiply\@dashdim \thr@@
356 \fi
357 \@defaultunitsset\@tempdimb{#1}\unitlength
358 \setbox\@dashbox\hbox{%
359 \vrule \@width \@wholewidth \@height\@tempdimb}%
360 \@tempcnta\z@
361 \put(0,0){\hskip -\@halfwidth \vbox{\@whilenum \@tempcnta <\@dashcnt
362 \do{\vskip\@tempdimb\copy\@dashbox\advance\@tempcnta \@ne }%
363 \vskip\@dashdim}}\@tempcnta\z@
364 \put(#2,0){\hskip -\@halfwidth \vbox{\@whilenum \@tempcnta<\@dashcnt
365 \do{\vskip\@tempdimb\copy\@dashbox\advance\@tempcnta \@ne }%
366 \vskip\@dashdim}}}\@makepicbox(#2,#3)}
367 /2kernel | latexrelease)

368 \latexrelease\EndIncludeInRelease
369 \latexrelease\IncludeInRelease{0000/00/00}%
370 \latexrelease \dashbox\default units}%
371 \latexrelease\expandafter\let\csname dashbox \endcsname\@undefind
372 \latexrelease\def\dashbox#1(#2,#3){%
373 \latexrelease\leavevmode\hbext@z@{\baselineskip \z@skip
374 \latexrelease\lineskip \z@skip
375 \latexrelease\@dashdim #2\unitlength
376 \latexrelease\@dashcnt \dashdim \advance\@dashcnt 200
377 \latexrelease\@dashdim #1\unitlength\divide\@dashcnt \dashdim
378 \latexrelease\ifodd\@dashcnt\@dashdim \z@
379 \latexrelease\advance\@dashcnt \@ne \divide\@dashcnt \tw@
380 \latexrelease\else \divide\@dashdim \tw@ \divide\@dashcnt \tw@
381 \latexrelease\advance\@dashcnt \m@ne
382 \latexrelease\setbox\@dashbox \hbox{%
383 \latexrelease \vrule \@height \@halfwidth \@depth \@halfwidth
384 \latexrelease \@width \dashdim}\put(0,0){\copy\@dashbox}%
385 \latexrelease\put(0,#3){\copy\@dashbox}%
386 \latexrelease\put(#2,0){\hskip-\dashdim\copy\@dashbox}%
387 \latexrelease\put(#2,#3){\hskip-\dashdim\box\@dashbox}%
388 \latexrelease\multiply\@dashdim \thr@@
389 \latexrelease\fi
390 \latexrelease\setbox\@dashbox \hbox{%
391 \latexrelease \vrule \@height \@halfwidth \@depth \@halfwidth
392 \latexrelease \@width #1\unitlength\hskip #1\unitlength}\@tempcnta\z@
393 \latexrelease\put(0,0){\hskip\@dashdim \@whilenum \@tempcnta <\@dashcnt
394 \latexrelease\do{\copy\@dashbox\advance\@tempcnta \@ne }}\@tempcnta\z@
395 \latexrelease\put(0,#3){\hskip\@dashdim \@whilenum \@tempcnta <\@dashcnt
396 \latexrelease\do{\copy\@dashbox\advance\@tempcnta \@ne }}%
397 \latexrelease\@dashdim #3\unitlength
398 \latexrelease\@dashcnt \dashdim \advance\@dashcnt 200
399 \latexrelease\@dashdim #1\unitlength\divide\@dashcnt \dashdim
400 \latexrelease\ifodd\@dashcnt \dashdim \z@

```

```

401 <latexrelease>\advance\@dashcnt \@ne \divide\@dashcnt \tw@
402 <latexrelease>\else
403 <latexrelease>\divide\@dashdim \tw@ \divide\@dashcnt \tw@
404 <latexrelease>\advance\@dashcnt \m@ne
405 <latexrelease>\setbox\@dashbox\hbox{\hskip -\@halfwidth
406 <latexrelease>\vrule \@width \@wholewidth
407 <latexrelease>\@height \@dashdim}\put(0,0){\copy\@dashbox}%
408 <latexrelease>\put(#2,0){\copy\@dashbox}%
409 <latexrelease>\put(0,#3){\lower\@dashdim\copy\@dashbox}%
410 <latexrelease>\put(#2,#3){\lower\@dashdim\copy\@dashbox}%
411 <latexrelease>\multiply\@dashdim \thr@@
412 <latexrelease>\fi
413 <latexrelease>\setbox\@dashbox\hbox{\vrule \@width \@wholewidth
414 <latexrelease>\@height #1\unitlength}\@tempcnta\z@
415 <latexrelease>\put(0,0){%
416 <latexrelease> \hskip -\@halfwidth \vbox{\@whilenum \@tempcnta <\@dashcnt
417 <latexrelease> \do{\vskip #1\unitlength\copy\@dashbox
418 <latexrelease> \advance\@tempcnta\@ne }%
419 <latexrelease> \vskip\@dashdim}}\@tempcnta\z@
420 <latexrelease>\put(#2,0){%
421 <latexrelease> \hskip -\@halfwidth \vbox{\@whilenum \@tempcnta<\@dashcnt
422 <latexrelease> \do{\vskip #1\unitlength\copy\@dashbox
423 <latexrelease> \advance\@tempcnta \@ne }%
424 <latexrelease> \vskip\@dashdim}}\@makepicbox(#2,#3)}
425 <latexrelease>\EndIncludeInRelease
426 <*2kernel>

```

(End definition for \dashbox.)

Historical L^AT_EX 2.09 comments (not necessarily accurate any more):

CIRCLES AND OVALS

USER COMMANDS:

`\circle{D}` : Produces the circle with the diameter as close as possible to $D * \text{\unitlength}$. `\put(X,Y){\circle{D}}` puts the circle with its center at (X,Y).

`\oval(X,Y)` : Makes an oval as round as possible that fits in the rectangle of width $X * \text{\unitlength}$ and height $Y * \text{\unitlength}$. The reference point is the center.

`\oval(X,Y)[POS]` : Save as `\oval(X,Y)` except it draws only the half or quadrant of the oval indicated by POS. E.G., `\oval(X,Y)[t]` draws just the top half and `\oval(X,Y)[br]` draws just the bottom right quadrant. In all cases, the reference point is the same as the unqualified `\oval(X,Y)` command.

`\@ovvert {DELTA1} {DELTA2}` : Makes a vbox containing either the left side or the right side of the oval being constructed. The baseline will coincide with the outside bottom edge of the oval; the left side of the box will coincide with the left edge of the vertical

rule. The width of the box will be `\@tempdima`.
`DELTA1` and `DELTA2` are added to the character number in `\@tempcnta`
to get the characters for the top and bottom quarter circle pieces.

`\@ovhorz` : Makes an hbox containing the straight rule for either the
top or the bottom of the oval being constructed. The baseline
will coincide with bottom edge of the rule; the left side of
the box will coincide with the left side of the oval.
The width of the box will be `\@ovxx`.

`\@getcirc {DIAM}` : Sets `\@tempcnta` to the character number
of the top-right quarter circle with the largest
diameter less than or equal to `DIAM`.
Sets `\@tempboxa` to an hbox containing that character.
Sets `\@tempdima` to `\wd \@tempboxa`, which is the distance
from the circle's left outside edge to its right
inside edge.
(These characters are like those described in the
TeXbook, pp. 389-90.)

```
\@getcirc {DIAM} ==
BEGIN
  \@tempcnta      := integer coercion of (DIAM + 2pt)
                                + 2pt added 1 Nov 88
  \@tempcnta      := \@tempcnta / integer coercion of 4pt
  if \@tempcnta > 10
    then \@tempcnta := 10 fi
  if \@tempcnta > 0
    then \@tempcnta := \@tempcnta-1
    else LaTeX Warning: Oval too small.
  fi
  \@tempcnta      := 4 * \@tempcnta
  \@tempboxa      := \hbox{\@circlefont \char \@tempcnta}
  \@tempdima      := \wd \@tempboxa
END
```

```
\@put{X}{Y}{OBJ} ==
BEGIN
  \raise Y \hb@xt@ 0pt{\hskip X OBJ \hss}
END
```

```
\@oval(X,Y)[POS] ==
BEGIN
  \beginngroup
  \boxmaxdepth := \maxdimen
  @ovt := @ovb := @ovl := @ovr := true
  for all E in POS
    do @ovE := false od
  \@ovxx      := X * \unitlength
  \@ovyy      := Y * \unitlength
```

```

\@tempdimb := min(\@ovxx,\@ovyy)
\@getcirc{\@tempdimb-2pt} %% "-2pt" added 7 Dec 89
\@ovro := \ht \@tempboxa
\@ovri := \dp \@tempboxa
\@ovdx := \@ovxx - \@tempdima
\@ovdx := \@ovdx/2
\@ovdy := \@ovyy - \@tempdima
\@ovdy := \@ovdy/2
\@circlefnt
\@tempboxa :=
  \hbox{
    if @ovr
      then \@ovvert{3}{2} \kern -\@tempdima
    fi
    if @ovl
      then \kern \@ovxx \@ovvert{0}{1} \kern -\@tempdima
        \kern -\@ovxx
      fi
    if @ovt
      then \@ovhorz \kern -\@ovxx
    fi
    if @ovb
      then \raise \@ovyy \@ovhorz
    fi
  }
\@ovdx := \@ovdx + \@ovro
\@ovdy := \@ovdy + \@ovro
\ht\@tempboxa := \dp\@tempboxa := 0
\@put{-\@ovdx}{-\@ovdy}{\box\@tempboxa}
\endgroup
END

\@ovvert {DELTA1} {DELTA2} ==
BEGIN
  \vbox to \@ovyy {
    if @ovb
      then \@tempcntb := \@tempcnta + DELTA1
        \kern -\@ovro
        \hbox { \char \@tempcntb }
        \nointerlineskip
      else \kern \@ovri \kern \@ovdy
    fi
    \leaders \vrule width \@wholewidth \vfil
    \nointerlineskip
    if @ovt
      then \@tempcntb := \@tempcnta + DELTA2
        \hbox { \char \@tempcntb }
      else \kern \@ovdy \kern \@ovro
    fi
  }

```



```

END

\@ovhorz ==
BEGIN
  \hb@xt@ \ovxxx{
    \kern \@ovro
    if @ovr
      then
        else \kern \@ovdx
      fi
    \leaders \hrule height \@wholewidth \hfil
    if @ovl
      then
        else \kern \@ovdx
      fi
    \kern \@ovri
  }
END

\circle{DIAM} ==
BEGIN
  \begingroup
  \boxmaxdepth := maxdimen
  \@tempdimb := DIAM *\unitlength
  if \@tempdimb > 15.5pt
    then \@getcirc{\@tempdimb}
      \@ovro := \ht \@tempboxa
      \@tempboxa := \hbox{
        \@circlefnt
        \@tempcnta := \@tempcnta + 2
        \char \@tempcnta
        \@tempcnta := \@tempcnta - 1
        \char \@tempcnta
        \kern -2\@tempdima
        \@tempcnta := \@tempcnta + 2
        \raise \@tempdima \hbox { \char \@tempcnta }
        \raise \@tempdima \box\@tempboxa
      }
      \ht\@tempboxa := \dp\@tempboxa := 0
      \@put{-\@ovro}{-\@ovro}{\@tempboxa}
    else
      \@circ{\@tempdimb}{96}
    fi
  \endgroup
END

\circle*{DIAM} == \dot{DIAM} == \@circ{DIAM*\unitlength}{112}

\@circ{DIAM}{CHAR} ==
BEGIN

```

```

\@tempcnta := integer coercion of (DIAM + .5pt)/1pt.
if \@tempcnta > 15 then \@tempcnta := 15 fi
if \@tempcnta > 1 then \@tempcnta := \@tempcnta - 1 fi
\@tempcnta := \@tempcnta + CHAR
\@circlefnt
\char \@tempcnta
END

```

End of historical L^AT_EX 2.09 comments.

```

\if@ovt If producing the Top Bottom Left or Right of an oval.
\if@ovb 427 \newif\if@ovt
\if@ovl 428 \newif\if@ovb
\if@ovr 429 \newif\if@ovl
430 \newif\if@ovr

```

(End definition for \if@ovt and others.)

```

\@ovxx
\@ovyy 431 \newdimen\@ovxx
\@ovdx 432 \newdimen\@ovyy
\@ovdy 433 \newdimen\@ovdx
\@ovro 434 \newdimen\@ovdy
\@ovri 435 \newdimen\@ovro
436 \newdimen\@ovri

```

(End definition for \@ovxx and others.)

\advance\@tempdima 2pt\relax added 1 Nov 88 to fix bug in which size of drawn circle not monotonic function of argument of \circle, caused by different rounding for dimensions of large and small circles.

```

\@getcirc
437 \def\@getcirc#1{\@tempdima #1\relax \advance\@tempdima 2\p@
438 \@tempcnta\@tempdima
439 \@tempdima 4\p@ \divide\@tempcnta\@tempdima
440 \ifnum \@tempcnta >10\relax
441 \@picture@warn
442 \@tempcnta 10\relax
443 \fi
444 \ifnum \@tempcnta >\z@ \advance\@tempcnta\m@ne
Warn if requirements for oval or circle can't be met.
445 \else \@picture@warn \fi
446 \multiply\@tempcnta 4\relax
447 \setbox \@tempboxa \hbox{\@circlefnt
448 \char \@tempcnta}\@tempdima \wd \@tempboxa}

```

(End definition for \@getcirc.)

\@picture@warn Generic warning for lines, vectors (used in \@sline) and oval or circle (used in \@getcirc) are not available at right size.

```

449 \def\@picture@warn{\@latex@warning{%
450 \string\oval, \string\circle, or \string\line\space
451 size unavailable}}

```

```

(End definition for \@picture@warn.)

\@put
452 \def\@put#1#2#3{\raise #2\hb@xt@{z@{\hskip #1#3\hss}}
(End definition for \@put.)

\oval
453 \def\oval(#1,#2){\@ifnextchar[{\@oval(#1,#2)}{\@oval(#1,#2) []}}
(End definition for \oval.)

454 </2ekernel>
455 <latexrelease>\IncludeInRelease{2016/03/31}%
456 <latexrelease>{\@ovhlinetrue}%
457 <latexrelease>{Avoid almost zero length leaders}%
458 <*2ekernel | latexrelease>

\if@ovvline Tests whether horizontal or vertical lines are needed.
\if@ovhline
459 \newif\if@ovvline \@ovvlinetrue
460 \newif\if@ovhline \@ovhlinetrue
461 % \begin{macrocode}
462 </2ekernel | latexrelease>
463 <latexrelease>\EndIncludeInRelease
464 <latexrelease>\IncludeInRelease{0000/00/00}%
465 <latexrelease>{\@ovhlinetrue}%
466 <latexrelease>{Avoid almost zero length leaders}%
467 <latexrelease>\let\if@ovvline\@undefined
468 <latexrelease>\let\if@ovhline\@undefined
469 <latexrelease>\EndIncludeInRelease
470 <*2ekernel>

(End definition for \if@ovvline and \if@ovhline.)

\@oval
471 </2ekernel>
472 <*2ekernel | latexrelease>
473 <latexrelease>\IncludeInRelease{2020/10/01}%
474 <latexrelease>{\@oval}{default units}%
475 \def\@oval(#1,#2)[#3]{\begingroup\boxmaxdepth \maxdimen
476 \ovttrue \ovbtrue \ovltrue \ovrtrue

477 \ovvlinefalse \ovhlinefalse

478 \tfor\reserved@a :=#3\do{%
479 \csname @ov\reserved@a false\endcsname}%
480 \@defaultunitsset\@ovxx{#1}\unitlength
481 \@defaultunitsset\@ovyy{#2}\unitlength

482 \@tempdimb \ifdim \ovyy >\ovxx \ovxx \ovvlinetrue
483 \else \ovyy \ifdim \ovyy =\ovxx \else \ovhlinetrue \fi\fi

484 \advance \@tempdimb -2\p@
485 \@getcirc \@tempdimb
486 \@ovro \ht\@tempboxa \@ovri \dp\@tempboxa
487 \@ovdx\@ovxx \advance\ovdx -\@tempdima \divide\ovdx \tw@
488 \@ovdy\@ovyy \advance\ovdy -\@tempdima \divide\ovdy \tw@

```

```

489 \ifdim \@ovdx >\z@ \@ovhlinetrue \fi
490 \ifdim \@ovdy >\z@ \@ovvlinetrue \fi

491 \@circlefnt \setbox\@tempboxa
492 \hbox{\if@ovr \@ovvert32\kern -\@tempdima \fi
493 \if@ovl \kern \@ovxx \@ovvert01\kern -\@tempdima \kern -\@ovxx \fi
494 \if@ovt \@ovhorz \kern -\@ovxx \fi
495 \if@ovb \raise \@ovyy \@ovhorz \fi}\advance\@ovdx\@ovro
496 \advance\@ovdy\@ovro \ht\@tempboxa\z@ \dp\@tempboxa\z@
497 \put{-\@ovdx}{-\@ovdy}{\box\@tempboxa}%
498 \endgroup}
499 </2ekernel | latexrelease>

500 <latexrelease>\EndIncludeInRelease
501 <latexrelease>\IncludeInRelease{2016/03/31}%
502 <latexrelease> \{\@oval\}{default units}%
503 <latexrelease>\def\@oval(#1,#2)[#3]{\begingroup\boxmaxdepth \maxdimen
504 <latexrelease> \@ovttrue \@ovbtrue \@ovltrue \@ovrtrue
505 <latexrelease> \@ovvlinefalse \@ovhlinefalse
506 <latexrelease> \tfor\reserved@a :=#3\do{%
507 <latexrelease> \csname @ov\reserved@a false\endcsname}%
508 <latexrelease> \@ovxx #1\unitlength
509 <latexrelease> \@ovyy #2\unitlength
510 <latexrelease> \@tempdimb \ifdim \@ovyy >\@ovxx \@ovxx \@ovvlinetrue
511 <latexrelease> \else \@ovyy \ifdim \@ovyy =\@ovxx \else \@ovhlinetrue
512 <latexrelease> \fi\fi
513 <latexrelease> \advance \@tempdimb -2\p@
514 <latexrelease> \@getcirc \@tempdimb
515 <latexrelease> \@ovro \ht\@tempboxa \@ovri \dp\@tempboxa
516 <latexrelease> \@ovdx\@ovxx \advance\@ovdx -\@tempdima \divide\@ovdx \tw@
517 <latexrelease> \@ovdy\@ovyy \advance\@ovdy -\@tempdima \divide\@ovdy \tw@
518 <latexrelease> \ifdim \@ovdx >\z@ \@ovhlinetrue \fi
519 <latexrelease> \ifdim \@ovdy >\z@ \@ovvlinetrue \fi
520 <latexrelease> \@circlefnt \setbox\@tempboxa
521 <latexrelease> \hbox{\if@ovr \@ovvert32\kern -\@tempdima \fi
522 <latexrelease> \if@ovl
523 <latexrelease> \kern \@ovxx \@ovvert01\kern -\@tempdima \kern -\@ovxx
524 <latexrelease> \fi
525 <latexrelease> \if@ovt \@ovhorz \kern -\@ovxx \fi
526 <latexrelease> \if@ovb \raise \@ovyy \@ovhorz \fi}\advance\@ovdx\@ovro
527 <latexrelease> \advance\@ovdy\@ovro \ht\@tempboxa\z@ \dp\@tempboxa\z@
528 <latexrelease> \put{-\@ovdx}{-\@ovdy}{\box\@tempboxa}%
529 <latexrelease> \endgroup}
530 <latexrelease>\EndIncludeInRelease

531 <latexrelease>\IncludeInRelease{0000/00/00}%
532 <latexrelease> \{\@oval\}{default units}%
533 <latexrelease>\def\@oval(#1,#2)[#3]{\begingroup\boxmaxdepth \maxdimen
534 <latexrelease> \@ovttrue \@ovbtrue \@ovltrue \@ovrtrue
535 <latexrelease> \tfor\reserved@a :=#3\do
536 <latexrelease> {\csname @ov\reserved@a false\endcsname}%
537 <latexrelease> \@ovxx #1\unitlength
538 <latexrelease> \@ovyy #2\unitlength
539 <latexrelease> \@tempdimb \ifdim \@ovyy >\@ovxx \@ovxx\else \@ovyy \fi
540 <latexrelease> \advance \@tempdimb -2\p@
541 <latexrelease> \@getcirc \@tempdimb

```

```

542 <latexrelease> \ovro \ht\@tempboxa \ovri \dp\@tempboxa
543 <latexrelease> \ovdx\ovxx \advance\ovdx -\@tempdima \divide\ovdx \tw@
544 <latexrelease> \ovdy\ovyy \advance\ovdy -\@tempdima \divide\ovdy \tw@
545 <latexrelease> \circlefnt \setbox\@tempboxa
546 <latexrelease> \hbox{\if@ovr \ovvert32\kern -\@tempdima \fi
547 <latexrelease> \if@ovl
548 <latexrelease> \kern \ovxx \ovvert01\kern -\@tempdima \kern -\ovxx
549 <latexrelease> \fi
550 <latexrelease> \if@ovt \ovhorz \kern -\ovxx \fi
551 <latexrelease> \if@ovb \raise \ovyy \ovhorz \fi}\advance\ovdx\ovro
552 <latexrelease> \advance\ovdy\ovro \ht\@tempboxa\z@ \dp\@tempboxa\z@
553 <latexrelease> \put{-\ovdx}{-\ovdy}{\box\@tempboxa}%
554 <latexrelease> \endgroup}
555 <latexrelease>\EndIncludeInRelease
556 (*2ekernel)

```

(End definition for \oval.)

\@ovvert

```

557 </2ekernel>
558 <latexrelease>\IncludeInRelease{2016/03/31}%
559 <latexrelease> {\@ovvert}{Avoid almost zero length leaders}%
560 (*2ekernel|latexrelease)
561 \def\@ovvert#1#2{\vbox to\ovyy{%
562 \if@ovb \@tempcntb \@tempcnta \advance \@tempcntb #1\relax
563 \kern -\ovro \hbox{\char \@tempcntb}\nointerlineskip
564 \else \kern \ovri \kern \ovdy \fi
565 \if@ovvline \leaders\vrule \@width \@wholewidth \fi
566 \vfil \nointerlineskip
567 \if@ovt \@tempcntb \@tempcnta \advance \@tempcntb #2\relax
568 \hbox{\char \@tempcntb}%
569 \else \kern \ovdy \kern \ovro \fi}}
570 </2ekernel|latexrelease>
571 <latexrelease>\EndIncludeInRelease
572 <latexrelease>\IncludeInRelease{0000/00/00}%
573 <latexrelease> {\@ovvert}{Avoid almost zero length leaders}%
574 <latexrelease>\def\@ovvert#1#2{\vbox to\ovyy{%
575 <latexrelease> \if@ovb \@tempcntb \@tempcnta \advance \@tempcntb #1\relax
576 <latexrelease> \kern -\ovro \hbox{\char \@tempcntb}\nointerlineskip
577 <latexrelease> \else \kern \ovri \kern \ovdy \fi
578 <latexrelease> \leaders\vrule \@width \@wholewidth\vfil \nointerlineskip
579 <latexrelease> \if@ovt \@tempcntb \@tempcnta \advance \@tempcntb #2\relax
580 <latexrelease> \hbox{\char \@tempcntb}%
581 <latexrelease> \else \kern \ovdy \kern \ovro \fi}}
582 <latexrelease>\EndIncludeInRelease
583 (*2ekernel)

```

(End definition for \@ovvert.)

\@ovhorz

```

584 </2ekernel>
585 <latexrelease>\IncludeInRelease{2016/03/31}%
586 <latexrelease> {\@ovhorz}{Avoid almost zero length leaders}%

```

```

587 <*2ekernel | latexrelease>
588 \def\@ovhorz{\hb@xt@\@ovxx{\kern \@ovro
589   \if@ovr \else \kern \@ovdx \fi

590   \if@ovhline \leaders \hrule \@height \@wholewidth \fi

591   \hfil
592   \if@ovl \else \kern \@ovdx \fi
593   \kern \@ovri}}
594 </2ekernel | latexrelease>

595 <latexrelease>\EndIncludeInRelease
596 <latexrelease>\IncludeInRelease{0000/00/00}%
597 <latexrelease>          {\@ovhorz}{Avoid almost zero length leaders}%
598 <latexrelease>\def\@ovhorz{\hb@xt@\@ovxx{\kern \@ovro
599 <latexrelease>   \if@ovr \else \kern \@ovdx \fi
600 <latexrelease>   \leaders \hrule \@height \@wholewidth \hfil
601 <latexrelease>   \if@ovl \else \kern \@ovdx \fi
602 <latexrelease>   \kern \@ovri}}
603 <latexrelease>\EndIncludeInRelease
604 <*2ekernel>

```

(End definition for \@ovhorz.)

\circle

```

605 \def\circle{\@inmatherr\circle\@ifstar\@dot\@circle}

```

(End definition for \circle.)

\@circle

```

606 </2ekernel>
607 <*2ekernel | latexrelease>
608 <latexrelease>\IncludeInRelease{2020/10/01}%
609 <latexrelease>          {\@circle}{default units}%
610 \def\@circle#1{%
611   \begingroup \boxmaxdepth \maxdimen
612   \@defaultunitsset\@tempdimb{#1}\unitlength
613   \ifdim \@tempdimb >15.5\p@ \getcirc\@tempdimb
614     \@ovro\ht\@tempboxa
615     \setbox\@tempboxa\hbox{\@circlefnt
616       \advance\@tempcnta\tw@ \char \@tempcnta
617       \advance\@tempcnta\m@ne \char \@tempcnta \kern -2\@tempdima
618       \advance\@tempcnta\tw@
619       \raise \@tempdima \hbox{\char\@tempcnta}\raise \@tempdima
620       \box\@tempboxa}\ht\@tempboxa\z@ \dp\@tempboxa\z@
621       \@put{-\@ovro}{-\@ovro}{\box\@tempboxa}%
622     \else \@circ\@tempdimb{96}\fi\endgroup}
623 </2ekernel | latexrelease>

624 <latexrelease>\EndIncludeInRelease
625 <latexrelease>\IncludeInRelease{0000/00/00}%
626 <latexrelease>          {\@circle}{default units}%
627 <latexrelease>\def\@circle#1{%
628 <latexrelease>   \begingroup \boxmaxdepth \maxdimen \@tempdimb #1\unitlength
629 <latexrelease>   \ifdim \@tempdimb >15.5\p@ \getcirc\@tempdimb
630 <latexrelease>   \@ovro\ht\@tempboxa

```

```

631 <latexrelease> \setbox\@tempboxa\hbox{\@circlefnt
632 <latexrelease> \advance\@tempcnta\tw@ \char \@tempcnta
633 <latexrelease> \advance\@tempcnta\m@ne \char \@tempcnta
634 <latexrelease> \kern -2\@tempdima
635 <latexrelease> \advance\@tempcnta\tw@
636 <latexrelease> \raise \@tempdima \hbox{\char\@tempcnta}%
637 <latexrelease> \raise \@tempdima
638 <latexrelease> \box\@tempboxa\ht\@tempboxa\z@ \dp\@tempboxa\z@
639 <latexrelease> \put{-\@ovro}{-\@ovro}{\box\@tempboxa}%
640 <latexrelease> \else \@circ\@tempdimb{96}\fi\endgroup}
641 <latexrelease>\EndIncludeInRelease
642 <*2kernel>

```

(End definition for \@circle.)

\@dot Internal form of \circle*.

```

643 </2kernel>
644 <*2kernel | latexrelease>
645 <latexrelease>\IncludeInRelease{2020/10/01}%
646 <latexrelease> {\@dot}{default units}%
647 \def\@dot#1{%
648 \@defaultunitsset\@tempdimb{#1}\unitlength
649 \@circ\@tempdimb{112}}
650 </2kernel | latexrelease>
651 <latexrelease>\EndIncludeInRelease
652 <latexrelease>\IncludeInRelease{0000/00/00}%
653 <latexrelease> {\@dot}{default units}%
654 <latexrelease>\def\@dot#1{\@tempdimb #1\unitlength \@circ\@tempdimb{112}}
655 <latexrelease>\EndIncludeInRelease
656 <*2kernel>

```

(End definition for \@dot.)

\@circ

```

657 \def\@circ#1#2{\@tempdima #1\relax \advance\@tempdima .5\p@
658 \@tempcnta\@tempdima \@tempdima \p@
659 \divide\@tempcnta\@tempdima
660 \ifnum\@tempcnta >15\relax \@tempcnta 15\relax \fi
661 \ifnum \@tempcnta >\z@ \advance\@tempcnta\m@ne\fi
662 \advance\@tempcnta #2\relax
663 \@circlefnt \char\@tempcnta}

```

(End definition for \@circ.)

\@xarg Counters used for manipulating the ‘slope’ arguments.

```

\@yarg \newcount\@xarg
\@yyarg \newcount\@yarg
\newcount\@yyarg

```

(End definition for \@xarg, \@yarg, and \@yyarg.)

\@multicnt Counter used in \multitup, and also \multicolumn.

```

667 \newcount\@multicnt

```

(End definition for \@multicnt.)

```

\@xdim Length registers.
\@ydim 668 \newdimen\@xdim
        669 \newdimen\@ydim

(End definition for \@xdim and \@ydim.)

\@linechar Box for holding a line segment character, for sloping lines.
        670 \newbox\@linechar

(End definition for \@linechar.)

\@linelen Length of the line currently being built.
        671 \newdimen\@linelen

(End definition for \@linelen.)

\@clnwd Height and width of current line segment.
\@clnht 672 \newdimen\@clnwd
        673 \newdimen\@clnht

(End definition for \@clnwd and \@clnht.)

\@dashdim \dashbox internal registers.
\@dashbox 674 \newdimen\@dashdim
\@dashcnt 675 \newbox\@dashbox
        676 \newcount\@dashcnt

(End definition for \@dashdim, \@dashbox, and \@dashcnt.)
Initialization: “\thinlines”
677 \let\@linefnt\tenln
678 \let\@circlefnt\tencirc
679 \@wholewidth\fontdimen8\tenln
680 \@halfwidth .5\@wholewidth

```

1.1 Curves

The new `\qbezier` command, based on the old `\bezier` defined in `bezier.sty`.
Historical L^AT_EX 2.09 comments (not necessarily accurate any more):

```

\qbezier[N] == \bezier{N}

\bezier{N}(AX,AY)(BX,BY)(CX,CY) ==
BEGIN
  IF N = 0
  THEN \@xdima := |BX - AX|
      \@xb := |CX - BX|
      \@xa := Max(\@xa, \@xb)
      \@ya := |BY - AY|
      \@yb := |CY - BY|
      \@ya := Max(\@ya, \@yb)
      @sc := Max(\@xa, \@ya)
      %% The coefficient .5 below is the degree of overlap of
      %% successive points, where 1 is no overlap and 0 is

```



```

%% complete overlap. A coefficient of C multiplies
%% the number of points plotted by 1/C.
%%
\@xa := .5 * \@halfwidth
@sc := @sc / \@halfwidth
@sc := Max(@sc, qbeziermax)
ELSE @sc := N
@scp := @sc+1
\@xb := 2 * (BX - AX) * \unitlength
\@xa := ((CX-AX)*\unitlength - \@xb)/@sc
\@yb := 2 * (BY - AY) * \unitlength
\@ya := ((CY-AY)*\unitlength - \@yb)/@sc
\@pictdot := square rule of width \@wholewidth
\count@ := 0
WHILE \count@ < @scp
DO \@xdim := ((\count@*\@xa + @xb) / @sc) * \count@
\@ydim := ((\count@*\@ya + @yb) / @sc) * \count@
plot pt with relative coords (\@xdim,\@ydim)
\count@ := \count@+1
OD

```

End of historical L^AT_EX 2.09 comments.

\qbeziermax The maximum number of points to plot.

```
681 \def\qbeziermax{500}
```

(End definition for \qbeziermax.)

In the code below, to save registers \@a ... are not used. Instead other registers are reused.

```

\newcounter{@sc} -> \c@multicnt
\newcounter{@scp} -> \@tempcnta
\newdimen\@xa -> \@ovxx
\newdimen\@xb -> \@ovdx
\newdimen\@ya -> \@ovyy
\newdimen\@yb -> \@ovdy
\newsavebox{\@pictdot} -> \@tempboxa

```

\qbezier Main user-level command to plot quadratic bezier curves. #2 should be (.

```
682 \newcommand\qbezier[2][0]{\bezier{#1}#2}
```

(End definition for \qbezier.)

\bezier Form of \bezier compatible with 2.09 bezier.sty, but modified to ignore spaces between its arguments. #2 should be white space, and #4 should be (.

```
683 \def\bezier#1)#2(#3)#4({\@bezier#1)(#3){}
```

```

\@bezier 684 {/2ekernel}
685 {*2ekernel | latexrelease}
686 {latexrelease}\IncludeInRelease{2020/10/01}%
687 {latexrelease} {\@bezier}{default units}%
688 \def\@bezier#1(#2,#3)(#4,#5)(#6,#7){%
689 \ifnum #1=\z@
690 \@defaultunitsset\@ovxx{#4}\unitlength
691 \@defaultunitsset{\advance\@ovxx}{-#2}\unitlength
692 \ifdim \@ovxx<\z@ \@ovxx -\@ovxx \fi
693 \@defaultunitsset\@ovdx{#6}\unitlength
694 \@defaultunitsset{\advance\@ovdx}{-#4}\unitlength
695 \ifdim \@ovdx<\z@ \@ovdx -\@ovdx \fi
696 \ifdim \@ovxx<\@ovdx \@ovxx \@ovdx \fi
697 \@defaultunitsset\@ovyy{#5}\unitlength
698 \@defaultunitsset{\advance\@ovyy}{-#3}\unitlength
699 \ifdim \@ovyy<\z@ \@ovyy -\@ovyy \fi
700 \@defaultunitsset\@ovdy{#7}\unitlength
701 \@defaultunitsset{\advance\@ovdy}{-#5}\unitlength
702 \ifdim \@ovdy<\z@ \@ovdy -\@ovdy \fi
703 \ifdim \@ovyy<\@ovdy \@ovyy \@ovdy \fi
704 \@multicnt
705 \ifdim \@ovxx>\@ovyy \@ovxx \else \@ovyy \fi
706 \@ovxx .5\@halfwidth \divide\@multicnt\@ovxx
707 \ifnum \qbeziermax<\@multicnt
708 \@multicnt\qbeziermax\relax
709 \fi
710 \else \@multicnt#1\relax \fi
711 \@tempcnta\@multicnt \advance\@tempcnta\@ne
712 \@defaultunitsset\@ovdx{#4}\unitlength
713 \@defaultunitsset{\advance\@ovdx}{-#2}\unitlength
714 \multiply\@ovdx \tw@
715 \@defaultunitsset\@ovxx{#6}\unitlength
716 \@defaultunitsset{\advance\@ovxx}{-#2}\unitlength
717 \advance\@ovxx -\@ovdx \divide\@ovxx\@multicnt
718 \@defaultunitsset\@ovdy{#5}\unitlength
719 \@defaultunitsset{\advance\@ovdy}{-#3}\unitlength
720 \multiply\@ovdy \tw@
721 \@defaultunitsset\@ovyy{#7}\unitlength
722 \@defaultunitsset{\advance\@ovyy}{-#3}\unitlength
723 \advance\@ovyy -\@ovdy \divide\@ovyy\@multicnt
724 \setbox\@tempboxa\hbox{%
725 \hskip -\@halfwidth
726 \vrule \@height\@halfwidth
727 \@depth \@halfwidth
728 \@width \@wholewidth}%
729 \put(#2,#3){%
730 \count@\z@
731 \@whilenum{\count@<\@tempcnta}\do
732 {\@xdim\count@\@ovxx
733 \advance\@xdim\@ovdx
734 \divide\@xdim\@multicnt
735 \multiply\@xdim\count@

```

```

736      \ydim\count@\@ovvy
737      \advance\ydim\@ovdy
738      \divide\ydim\@multicnt
739      \multiply\ydim\count@
740      \raise \ydim
741      \hb@xt@\z@{\kern\@xdim
742              \unhcopy\@tempboxa\hss}%
743      \advance\count@\@ne}}
744 (/2ekernel | latexrelease)

745 <latexrelease>\EndIncludeInRelease
746 <latexrelease>\IncludeInRelease{0000/00/00}%
747 <latexrelease>      {\@bezier}{default units}%
748 <latexrelease>\def\@bezier#1(#2,#3)(#4,#5)(#6,#7){%
749 <latexrelease>  \ifnum #1=\z@
750 <latexrelease>      \@ovxx #4\unitlength
751 <latexrelease>      \advance\@ovxx -#2\unitlength
752 <latexrelease>      \ifdim \@ovxx<\z@ \@ovxx -\@ovxx \fi
753 <latexrelease>      \@ovdx #6\unitlength
754 <latexrelease>      \advance\@ovdx -#4\unitlength
755 <latexrelease>      \ifdim \@ovdx<\z@ \@ovdx -\@ovdx \fi
756 <latexrelease>      \ifdim \@ovxx<\@ovdx \@ovxx \@ovdx \fi
757 <latexrelease>      \@ovyy #5\unitlength
758 <latexrelease>      \advance\@ovyy -#3\unitlength
759 <latexrelease>      \ifdim \@ovyy<\z@ \@ovyy -\@ovyy \fi
760 <latexrelease>      \@ovdy #7\unitlength
761 <latexrelease>      \advance\@ovdy -#5\unitlength
762 <latexrelease>      \ifdim \@ovdy<\z@ \@ovdy -\@ovdy \fi
763 <latexrelease>      \ifdim \@ovyy<\@ovdy \@ovyy \@ovdy \fi
764 <latexrelease>      \@multicnt
765 <latexrelease>      \ifdim \@ovxx>\@ovyy \@ovxx \else \@ovyy \fi
766 <latexrelease>      \@ovxx .5\@halfwidth \divide\@multicnt\@ovxx
767 <latexrelease>      \ifnum
768 <latexrelease>      \qbeziermax<\@multicnt \@multicnt\qbeziermax\relax
769 <latexrelease>      \fi
770 <latexrelease> \else \@multicnt#1\relax \fi
771 <latexrelease> \@tempcnta\@multicnt \advance\@tempcnta\@ne
772 <latexrelease> \@ovdx #4\unitlength \advance\@ovdx -#2\unitlength
773 <latexrelease>      \multiply\@ovdx \tw@
774 <latexrelease> \@ovxx #6\unitlength \advance\@ovxx -#2\unitlength
775 <latexrelease>      \advance\@ovxx -\@ovdx \divide\@ovxx\@multicnt
776 <latexrelease> \@ovdy #5\unitlength \advance\@ovdy -#3\unitlength
777 <latexrelease>      \multiply\@ovdy \tw@
778 <latexrelease> \@ovyy #7\unitlength \advance\@ovyy -#3\unitlength
779 <latexrelease>      \advance\@ovyy -\@ovdy \divide\@ovyy\@multicnt
780 <latexrelease> \setbox\@tempboxa\hbox{%
781 <latexrelease>      \hskip -\@halfwidth
782 <latexrelease>      \vrule \@height\@halfwidth
783 <latexrelease>      \@depth \@halfwidth
784 <latexrelease>      \@width \@wholewidth}%
785 <latexrelease> \put(#2,#3){%
786 <latexrelease>      \count@\z@
787 <latexrelease>      \@whilenum{\count@<\@tempcnta}\do
788 <latexrelease>          {\@xdim\count@\@ovxx
789 <latexrelease>          \advance\@xdim\@ovdx

```

```

790 <latexrelease>          \divide\@xdim\@multicnt
791 <latexrelease>          \multiply\@xdim\count@
792 <latexrelease>          \@ydim\count@\@ovvy
793 <latexrelease>          \advance\@ydim\@ovdy
794 <latexrelease>          \divide\@ydim\@multicnt
795 <latexrelease>          \multiply\@ydim\count@
796 <latexrelease>          \raise \@ydim
797 <latexrelease>          \hb@xt@\z@{\kern\@xdim
798 <latexrelease>          \unhcopy\@tempboxa\hss}%
799 <latexrelease>          \advance\count@\@ne}}
800 <latexrelease>\EndIncludeInRelease
801 <*2ekernel>

```

(End definition for \bezier and \@bezier.)

As the commands above all use “picture” interface we couldn’t define them with \DeclareRobustCommand so we do that now.

```

802 </2ekernel>
803 <*2ekernel | latexrelease>
804 <latexrelease>\IncludeInRelease{2019/10/01}%
805 <latexrelease>          {\bezier}{Make commands robust}%
806 \MakeRobust\bezier
807 \MakeRobust\circle
808 \MakeRobust\dashbox
809 \MakeRobust\line
810 \MakeRobust\linethickness
811 \MakeRobust\multiput
812 \MakeRobust\oval
813 \MakeRobust\put
814 \MakeRobust\qbezier
815 \MakeRobust\shortstack
816 \MakeRobust\thinline
817 \MakeRobust\vector
818 </2ekernel | latexrelease>
819 <latexrelease>\EndIncludeInRelease
820 <latexrelease>\IncludeInRelease{0000/00/00}%
821 <latexrelease>          {\bezier}{Make commands robust}%
822 <latexrelease>
823 <latexrelease>\kernel@make@fragile\bezier
824 <latexrelease>\kernel@make@fragile\circle
825 <latexrelease>\kernel@make@fragile\dashbox
826 <latexrelease>\kernel@make@fragile\line
827 <latexrelease>\kernel@make@fragile\linethickness
828 <latexrelease>\kernel@make@fragile\multiput
829 <latexrelease>\kernel@make@fragile\oval
830 <latexrelease>\kernel@make@fragile\put
831 <latexrelease>\kernel@make@fragile\qbezier
832 <latexrelease>\kernel@make@fragile\shortstack
833 <latexrelease>\kernel@make@fragile\thinline
834 <latexrelease>\kernel@make@fragile\vector
835 <latexrelease>
836 <latexrelease>\EndIncludeInRelease
837 <*2ekernel>
838 </2ekernel>

```

File M

ltthm.dtx

1 Theorem Environments

The user creates his own theorem-like environments with the command

```
\newtheorem{<name>}{<text>}[<counter>] or
\newtheorem{<name>}[<oldname>]{<text>}
```

This defines the environment $\langle name \rangle$ to be just as one would expect a theorem environment to be, except that it prints $\langle text \rangle$ instead of “Theorem”.

If $\langle oldname \rangle$ is given, then environments $\langle name \rangle$ and $\langle oldname \rangle$ use the same counter, so using a $\langle name \rangle$ environment advances the number of the next $\langle name \rangle$ environment, and vice-versa.

If $\langle counter \rangle$ is given, then environment $\langle name \rangle$ is numbered within $\langle counter \rangle$.

E.g., if $\langle counter \rangle = \text{subsection}$, then the first $\langle name \rangle$ in subsection 7.2 is numbered $\langle text \rangle$ 7.2.1.

The way $\langle name \rangle$ environments are numbered can be changed by redefining $\backslash\text{the}\langle name \rangle$.
Historical L^AT_EX 2.09 comments (not necessarily accurate any more):

DOCUMENT STYLE PARAMETERS

$\backslash\text{thmcounter}\{\text{COUNTER}\}$: A command such that

```
\edef\theCOUNTER{\@thmcounter{COUNTER}}
```

defines $\backslash\text{theCOUNTER}$ to produce a number for a theorem environment.

The default is:

```
BEGIN \noexpand\arabic{COUNTER} END
```

$\backslash\text{thmcountersep}$: A separator placed between a theorem number and the number of the counter within which it is numbered.

E.g., to make the third theorem of section 7.2 be numbered 7.2-3, $\backslash\text{thmcountersep}$ should be $\backslash\text{def}$ 'ed to ‘-’. Its default is ‘.’.

$\backslash\text{begintheorem}\{\text{NAME}\}\{\text{NUMBER}\}$: A command that begins a theorem environment for a ‘theorem’ named ‘NAME NUMBER’ –

e.g., $\backslash\text{begintheorem}\{\text{Lemma}\}\{3.7\}$ starts Lemma 3.7.

$\backslash\text{opargbegintheorem}\{\text{NAME}\}\{\text{NUMBER}\}\{\text{OPARG}\}$:

A command that begins a theorem environment for a ‘theorem’ named ‘NAME NUMBER’ with optional argument OPARG – e.g., $\backslash\text{begintheorem}\{\text{Lemma}\}\{3.7\}\{\text{Jones}\}$ starts ‘Lemma 3.7 (Jones)’.

$\backslash\text{endtheorem}$: A command that ends a theorem environment.

$\backslash\text{newtheorem}\{\text{NAME}\}\{\text{TEXT}\}[\text{COUNTER}] ==$

```
BEGIN
```

```
if \NAME is definable
```

```

then \@definecounter{NAME}
  if COUNTER present
    then \@newctr{NAME}[COUNTER] fi
    \theNAME == BEGIN \theCOUNTER \@thmcountersep
                                eval\@thmcounter{NAME} END
    else \theNAME == BEGIN eval\@thmcounter{NAME} END
    \NAME == \@thm{NAME}{TEXT}
    \endNAME == \@endtheorem
  else error
fi
END

\newtheorem{NAME}[OLDNAME]{TEXT}==
BEGIN
  if counter OLDNAME nonexistent
  then ERROR
  else
    if \NAME is definable
    then BEGIN
      \theNAME == \theOLDNAME
      \NAME == \@thm{OLDNAME}{TEXT}
      \endNAME == \@endtheorem
      END
    else error
    fi
  fi
END

\@thm{NAME}{TEXT} ==
BEGIN
  \refstepcounter{NAME}
  if next char = [
    then \@ythm{NAME}{TEXT}
    else \@xthm{NAME}{TEXT}
  fi
END

\@xthm{NAME}{TEXT} ==
BEGIN
  \@begintheorem{TEXT}{\theNAME}
  \ignorespaces
END

\@ythm{NAME}{TEXT}[OPARG] ==
BEGIN
  \@opargbegintheorem{TEXT}{\theNAME}{OPARG}
  \ignorespaces
END

```

End of historical L^AT_EX 2.09 comments.

`\newtheorem` `\newtheorem` ought really be allowed only in the preamble Which would be good document style, and allow some main memory to be saved by declaring these commands to be `\onlypreamble`. Unfortunately the L^AT_EX book indicates that `\newtheorem` may be used anywhere in the document...

```

1  \*2ekernel)
2  \def\newtheorem#1{%
3    \ifnextchar[{\@othm{#1}}{\@nthm{#1}}}
```

(End definition for `\newtheorem`.)

`\@nthm`

```

4  \def\@nthm#1#2{%
5    \ifnextchar[{\@xnthm{#1}{#2}}{\@ynthm{#1}{#2}}}
```

(End definition for `\@nthm`.)

`\@xnthm` 92/09/18 RmS: Changed `\@addtoreset` to `\@newctr` to produce error message if counter #3 does not exist (to be consistent with behaviour of `\newcounter`)

```

6  \def\@xnthm#1#2[#3]{%
7    \expandafter\@ifdefinable\csname #1\endcsname
8      {\@definecounter{#1}\@newctr{#1}[#3]%
9      \expandafter\xdef\csname the#1\endcsname{%
10       \expandafter\noexpand\csname the#3\endcsname \@thmcountersep
11       \@thmcounter{#1}}}%
12    \global\@namedef{#1}{\@thm{#1}{#2}}%
13    \global\@namedef{end#1}{\@endtheorem}}}
```

(End definition for `\@xnthm`.)

`\@ynthm`

```

14 \def\@ynthm#1#2{%
15   \expandafter\@ifdefinable\csname #1\endcsname
16     {\@definecounter{#1}%
17     \expandafter\xdef\csname the#1\endcsname{\@thmcounter{#1}}%
18     \global\@namedef{#1}{\@thm{#1}{#2}}%
19     \global\@namedef{end#1}{\@endtheorem}}}
```

(End definition for `\@ynthm`.)

`\@othm`

```

20 \def\@othm#1[#2]#3{%
21   \@ifundefined{c@#2}{\@nocounterr{#2}}%
22   {\expandafter\@ifdefinable\csname #1\endcsname
23     {\global\@namedef{the#1}{\@nameuse{the#2}}%
24     \global\@namedef{#1}{\@thm{#2}{#3}}%
25     \global\@namedef{end#1}{\@endtheorem}}}
```

(End definition for `\@othm`.)

`\@thm`

```

26 \def\@thm#1#2{%
27   \refstepcounter{#1}%
28   \@ifnextchar[{\@ythm{#1}{#2}}{\@xthm{#1}{#2}}}
```

(End definition for `\@thm`.)

```

\@xthm
\@ythm
29 \def\@xthm#1#2{%
30   \@begintheorem{#2}{\csname the#1\endcsname}\ignorespaces}
31 \def\@ythm#1#2[#3]{%
32   \@opargbegintheorem{#2}{\csname the#1\endcsname}{#3}\ignorespaces}

(End definition for \@xthm and \@ythm.)
Default values

\@thmcounter
\@thmcountersep
33 \def\@thmcounter#1{\noexpand\arabic{#1}}
34 \def\@thmcountersep{.}

(End definition for \@thmcounter and \@thmcountersep.)

\@begintheorem Providing theorem defaults.
\@opargbegintheorem
\@endtheorem
35 \def\@begintheorem#1#2{\trivlist
36   \item[\hskip \labelsep{\bfseries #1\ #2}]\itshape}
37 \def\@opargbegintheorem#1#2#3{\trivlist
38   \item[\hskip \labelsep{\bfseries #1\ #2\ (#3)}]\itshape}
39 \def\@endtheorem{\endtrivlist}
40 \endkernel

(End definition for \@begintheorem, \@opargbegintheorem, and \@endtheorem.)

```


File N

ltsect.dtx

1 Sectioning Commands

This file defines the declarations such as `\author` which are used by `\maketitle`. `\maketitle` itself is defined by each class, not in the L^AT_EX kernel.

The second part of the file defines the generic commands used for defining sectioning commands such as `\chapter`. Again the actual document level commands are defined in the class files, in terms of these commands.

```
1 <*2ekernel>
2 \message{title,}
```

1.1 The Title

```
\title The user defines the title and author by the declarations \title{<name>}, \author{<name>}
\author Similarly the date is declared with \date{<date>}.
\date Inside these, the \thanks{<footnote text>} command may be used to make acknowl-
\thanks edgements, notice of address, etc. in a footnote. If there are multiple authors, they have
\and to be separated with the \and command.
\maketitle And finally, the \maketitle command produces the actual title, using the informa-
tion previously saved with the other commands.

3 </2ekernel>
4 <*2ekernel | latexrelease>
5 <latexrelease>\IncludeInRelease{2019/10/01}%
6 <latexrelease> {\title}{Make commands robust}%

\title \title for use in \maketitle. If not given \maketitle will produce an error message.
7 \DeclareRobustCommand\title[1]{\gdef\@title{#1}}

(End definition for \title.)

\author \author for use in \maketitle. If not given \maketitle will produce a warning message.
8 \DeclareRobustCommand*\author[1]{\gdef\@author{#1}}

(End definition for \author.)

\date \date for use in \maketitle. If not given \maketitle will produce \today as the default.
9 \DeclareRobustCommand*\date[1]{\gdef\@date{#1}}

(End definition for \date.)

\thanks
10 \DeclareRobustCommand\thanks[1]{\footnotemark
11 \protected@xdef\@thanks{\@thanks
12 \protect\footnotetext[\the\c@footnote]{#1}}%
13 }

(End definition for \thanks.)
```

`\and`

```
14 \DeclareRobustCommand\and{%    % \begin{tabular}
15   \end{tabular}}%
16   \hskip 1em \@plus.17fil%
17   \begin{tabular}[t]{c}}%      % \end{tabular}
```

(End definition for \and.)

```
18 </2ekernel | latexrelease>
19 <latexrelease>\EndIncludeInRelease
20 <latexrelease>\IncludeInRelease{0000/00/00}%
21 <latexrelease>                {\title}{Make commands robust}%
22 <latexrelease>
23 <latexrelease>\kernel@make@fragile\title
24 <latexrelease>\kernel@make@fragile\author
25 <latexrelease>\kernel@make@fragile\date
26 <latexrelease>\kernel@make@fragile\thanks
27 <latexrelease>\kernel@make@fragile\and
28 <latexrelease>
29 <latexrelease>\EndIncludeInRelease
30 <*2ekernel>
```

`\@title`

```
31 \def\@title{\@latex@error{No \noexpand\title given}\@ehc}
```

(End definition for \@title.)

`\@author`

```
32 \def\@author{\@latex@warning@no@line{No \noexpand\author given}}
```

(End definition for \@author.)

`\@date`

```
33 \gdef\@date{\today}
```

(End definition for \@date.)

`\@thanks`

```
34 \let\@thanks\@empty
```

(End definition for \@thanks.)

```
35 \message{sectioning,}
```

1.2 Sectioning

`\@secpenalty`

```
36 \newcount\@secpenalty
37 \@secpenalty = -300
```

(End definition for \@secpenalty.)

`\if@noskipsec` Way back in 1991 (08/26) FMi & RmS set the `\@noskipsec` switch to true for the
`\@noskipsectrue` preamble and to false in `\document`. This was done to trap lists and related text in the preamble but it does not catch everything.

```
38 \newif\if@noskipsec \@noskipsectrue
```

(End definition for \if@noskipsec and \@noskipsectrue.)

\@startsection The `\@startsection{<name>}{<level>}{<indent>}{<beforeskip>}{<afterskip>}{<style>}`*[`<altheading> <heading>`] command is the mother of all the user level sectioning commands. The part after the *, including the * is optional.

name: e.g., 'subsection'

level: a number, denoting depth of section – e.g., chapter = 0, section = 1, etc.

indent: Indentation of heading from left margin

beforeskip: Absolute value = skip to leave above the heading. If negative, then paragraph indent of text following heading is suppressed.

afterskip: if positive, then skip to leave below heading, else negative of skip to leave to right of run-in heading.

style: Commands to set style. Since June 1996 release the *last* command in this argument may be a command such as `\MakeUppercase` or `\fbox` that takes an argument. The section heading will be supplied as the argument to this command. So setting #6 to, say, `\bfseries\MakeUppercase` would produce bold, uppercase headings.

If '*' is missing, then increment the counter. If it is present, then there should be no [`<altheading>`] argument. The command uses the counter 'secnumdepth'. It contains a pointer to the highest section level that is to be numbered.

Warning: The `\@startsection` command should be at the same or higher grouping level as the text that follows it. For example, you should *not* do something like

```
\def\foo{ \begingroup ...
           \paragraph{...}
           \endgroup}
```

Pseudocode for the `\@startsection` command *Historical L^AT_EX 2.09 comments* (not necessarily accurate any more):

```
\@startsection
{NAME}{LEVEL}{INDENT}{BEFORESKIP}{AFTERSKIP}{STYLE} ==
BEGIN
  IF @noskipsec = T THEN \leavevmode FI
                                % true if previous section had no body.

  \par
  \@tempskipa := BEFORESKIP
  @afterindent := T
  IF \@tempskipa < 0 THEN \@tempskipa := -\@tempskipa
                                @afterindent := F
  FI
  IF @nobreak = true
    THEN \everypar == null
    ELSE \addpenalty{\@secpenalty}
          \addvspace{\@tempskipa}
  FI
  IF * next
```

```

        THEN \@ssect{INDENT}{BEFORESKIP}{AFTERSKIP}{STYLE}
        ELSE \@dblarg{\@sect
                     {NAME}{LEVEL}{INDENT}
                     {BEFORESKIP}{AFTERSKIP}{STYLE}}
    FI
END
End of historical LATEX 2.09 comments.

```

```

39 \def\@startsection#1#2#3#4#5#6{%
40   \if@noskipsec \leavevmode \fi
41   \par
42   \@tempskipa #4\relax
43   \@afterindenttrue
44   \ifdim \@tempskipa <\z@
45     \@tempskipa -\@tempskipa \@afterindentfalse
46   \fi
47   \if@nobreak
48     \everypar{}%
49   \else
50     \addpenalty\@secpenalty\addvspace\@tempskipa
51   \fi
52   \@ifstar
53     {\@ssect{#3}{#4}{#5}{#6}}%
54     {\@dblarg{\@sect{#1}{#2}{#3}{#4}{#5}{#6}}}

```

(End definition for \@startsection.)

\@sect Pseudocode for the \@sect command *Historical L^AT_EX 2.09 comments (not necessarily accurate any more):*

```

\@sect{NAME}{LEVEL}
      {INDENT}{BEFORESKIP}{AFTERSKIP}
      {STYLE}[ARG1]{ARG2}
      ==
BEGIN
  IF LEVEL > \c@secnumdepth
    THEN \@svsec :=L null
    ELSE \refstepcounter{NAME}
         \@svsec :=L BEGIN \@seccntformat{#1}\relax END
  FI
  IF AFTERSKIP > 0
    THEN \begingroup
         STYLE
         \@hangfrom{\hskip INDENT\@svsec}
         {\interlinepenalty 10000 ARG2\par}
         \endgroup
         \NAMEmark{ARG1}
         \addcontentsline{toc}{NAME}
         { IF LEVEL > \c@secnumdepth
           ELSE \protect\numberline{\theNAME} FI
           ARG1 }
    ELSE \@svsechd == BEGIN STYLE
                                   \hskip INDENT\@svsec

```

```

ARG2
\NAMEmark{ARG1}
\addcontentsline{toc}{NAME}
{ IF LEVEL > \c@secnumdepth
ELSE
\protect\numberline{\theNAME}
FI
ARG1 }

END

FI
\@xsect{AFTERSKIP}
END
End of historical LATEX 2.09 comments.

55 \def\@sect#1#2#3#4#5#6[#7]#8{%
56 \ifnum #2>\c@secnumdepth
57 \let\@svsec\@empty
58 \else
59 \refstepcounter{#1}%

Since \@secntformat might end with an improper \hskip which is scanning forward
for plus or minus we end the definition of \@svsec with \relax as a precaution.

60 \protected@edef\@svsec{\@secntformat{#1}\relax}%
61 \fi
62 \@tempskipa #5\relax
63 \ifdim \@tempskipa>\z@
64 \begingroup

This { used to be after the argument to \@hangfrom but was moved here to allow com-
mands such as \MakeUppercase to be used at the end of #6.

65 #6{%
66 \@hangfrom{\hskip #3\relax\@svsec}%
67 \interlinepenalty \@M #8\@par}%
68 \endgroup
69 \csname #1mark\endcsname{#7}%
70 \addcontentsline{toc}{#1}{%
71 \ifnum #2>\c@secnumdepth \else
72 \protect\numberline{\csname the#1\endcsname}%
73 \fi
74 #7}%
75 \else

\relax added 2 May 90

76 \def\@svsechd{%
77 #6{\hskip #3\relax
78 \@svsec #8}%
79 \csname #1mark\endcsname{#7}%
80 \addcontentsline{toc}{#1}{%
81 \ifnum #2>\c@secnumdepth \else
82 \protect\numberline{\csname the#1\endcsname}%
83 \fi
84 #7}}%
85 \fi
86 \@xsect{#5}}

```

(End definition for \@sect.)

`\@xsect` Pseudocode for the `\@xsect` command *Historical L^AT_EX 2.09 comments (not necessarily accurate any more):*

```
\@xsect{AFTERSKIP} ==
BEGIN
  IF AFTERSKIP > 0
    THEN \par \nobreak
         \vskip AFTERSKIP
         \@afterheading
    ELSE @nobreak :=G F
         @noskipsec :=G T
         \everypar{ IF @noskipsec = T
                     THEN @noskipsec :=G F
                        \clubpenalty := 10000 % local
                        \hskip -\parindent
                        \begingroup
                        \@svsechd
                        \endgroup
                        \unskip
                        \hskip -AFTERSKIP \relax
                        %% relax added 14 Jan 91
                     ELSE \clubpenalty := \@clubpenalty % local
                        \everypar := NULL
                     FI
                   }
         FI
  FI
END
```

End of historical L^AT_EX 2.09 comments.

```
87 \def\@xsect#1{%
88   \@tempskipa #1\relax
89   \ifdim \@tempskipa>\z@
```

Why not combine `\@sect` and `\@xsect` and save doing the same test twice? It is not possible to change this now as these have become hooks!

This `\par` seems unnecessary.

```
90   \par \nobreak
91   \vskip \@tempskipa
92   \@afterheading
93 \else
94   \@nobreakfalse
95   \global\@noskipsectrue
96   \everypar{%
97     \if@noskipsec
98       \global\@noskipsecfalse
99       {\setbox\z@\lastbox}%
100       \clubpenalty\@M
101       \begingroup \@svsechd \endgroup
102       \unskip
103       \@tempskipa #1\relax
```

```

104      \hskip -\@tempskipa
105      \else
106        \clubpenalty \@clubpenalty
107        \everypar{}\%
108      \fi}%
109    \fi
110    \ignorespaces}

```

(End definition for \xsect.)

\@secntformat This command formats the section number including the space following it.

```

111 \def\@secntformat#1{\csname the#1\endcsname\quad}

```

(End definition for \@secntformat.)

Pseudocode for the \ssect command *Historical L^AT_EX 2.09 comments (not necessarily accurate any more)*:

```

\ssect{INDENT}{BEFORESKIP}{AFTERSKIP}{STYLE}{ARG} ==
BEGIN
  IF AFTERSKIP > 0
    THEN \begingroup
          STYLE
          \@hangfrom{\hskip INDENT}
                {\interlinepenalty 10000 ARG\par}
          \endgroup
    ELSE \@svsechd == BEGIN STYLE
                      \hskip INDENT
                      ARG
                      END
    FI
    \xsect{AFTERSKIP}
  END

```

End of historical L^AT_EX 2.09 comments.

Pseudocode for the \@afterheading command *Historical L^AT_EX 2.09 comments (not necessarily accurate any more)*:

```

\@afterheading ==
BEGIN
  @nobreak :=G true
  \everypar := BEGIN IF @nobreak = T
                  THEN @nobreak :=G false
                  \clubpenalty := 10000 % local
                  IF @afterindent = F
                    THEN remove \lastbox
                  FI
                  ELSE \clubpenalty := \@clubpenalty % local
                  \everypar := NULL
                FI
  END

```

END

End of historical L^AT_EX 2.09 comments.

`\@ssect`

```

112 \def\@ssect#1#2#3#4#5{%
113   \@tempkipa #3\relax
114   \ifdim \@tempkipa>\z@
115     \begingroup
This { used to be after the argument to \@hangfrom but was moved here to allow com-
mands such as \MakeUppercase to be used at the end of #4.
116     #4{%
117       \@hangfrom{\hskip #1}%
118       \interlinepenalty \@M #5\@par}%
119     \endgroup
120   \else
121     \def\@svsechd{#4{\hskip #1\relax #5}}%
122   \fi
123   \@xsect{#3}}

```

(End definition for \@ssect.)

`\if@afterindent`
`\@afterindenttrue`

```

124 \newif\if@afterindent \@afterindenttrue

```

(End definition for \if@afterindent and \@afterindenttrue.)

`\@afterheading`

This hook is used in setting up custom-built headings in classes.dtx.

```

125 \def\@afterheading{%
126   \@nbreaktrue
127   \everypar{%
128     \if@nbreak
129       \@nbreakfalse
130       \clubpenalty \@M
131       \if@afterindent \else
132         {\setbox\z@\lastbox}%
133       \fi
134     \else
135       \clubpenalty \@clubpenalty
136     \everypar{}%
137   \fi}}

```

(End definition for \@afterheading.)

`\@hangfrom` `\@hangfrom{<text>}` : Puts *<text>* in a box, and makes a hanging indentation of the following material up to the first `\par`. Should be used in vertical mode.

```

138 \def\@hangfrom#1{\setbox\@tempboxa\hbox{#1}}%
139   \hangindent \wd\@tempboxa\noindent\box\@tempboxa}

```

(End definition for \@hangfrom.)

`\c@secnumdepth`
`\c@tocdepth`

```

140 \newcount\c@secnumdepth
141 \newcount\c@tocdepth

```

(End definition for \c@secnumdepth and \c@tocdepth.)

`\secdef \secdef{<unstarcmds>}{<unstarcmds>}{<starcmds>}`

When defining a `\chapter` or `\section` command without using `\@startsection`, you can use `\secdef` as follows:

1. `\def\chapter{ ... \secdef \<starcmd> \<unstarcmd> }`
2. `\def\<starcmd>[#1]#2{ ... } % Command to define \chapter[...]{...}`
3. `\def\<unstarcmd>#1{ ... } % Command to define \chapter*{...}`

142 `\def\secdef#1#2{\@ifstar{#2}{\@dblarg{#1}}}`

(End definition for `\secdef`.)

1.2.1 Initializations

```
\sectionmark
\subsectionmark
\subsubsectionmark
\paragraphmark
\subparagraphmark
```

143 `\let\sectionmark\@gobble`

144 `\let\subsectionmark\@gobble`

145 `\let\subsubsectionmark\@gobble`

146 `\let\paragraphmark\@gobble`

147 `\let\subparagraphmark\@gobble`

(End definition for `\sectionmark` and others.)

148 `\message{contents,}`

1.3 Table of Contents etc.

1.3.1 Convention

`\tf@<foo>` = file number for output for table foo. The file is opened only if `@files` = true.

1.3.2 Commands

A `\l@<type>{<entry>}{<page>}` Macro needs to be defined by document style for making an entry of type `<type>` in a table of contents, etc. E.g., the document style should define `\l@chapter`, `\l@section`, etc.

Note: When the `\protect` command is used in the `<entry>` or `<text>` of one of the commands below, it causes the following control sequence to be written on the file without being expanded. The sequence will be expanded when the table of contents entry is processed.

Surprise: Inside an `\addcontentsline` or `\addtocontents` command argument, the commands: `\index`, `\glossary`, and `\label` are no-ops. This could cause a problem if the user puts an `\index` or `\label` into one of the commands he writes, or into the optional ‘short version’ argument of a `\section` or `\caption` command.

`\@starttoc` The `\@starttoc{<ext>}` command is used to define the commands:

`\tableofcontents`, `\listoffigures`, etc.

For example: `\@starttoc{lof}` is used in `\listoffigures`. This command reads the `.<ext>` file and sets up to write the new `.<ext>` file.

Historical L^AT_EX 2.09 comments (not necessarily accurate any more):

`\@starttoc{EXT} ==`

```

BEGIN
  \begingroup
  \makeatletter
  read file \jobname.EXT
  IF @filesw = true
    THEN open \jobname.EXT as file \tf@EXT
  FI
  @nobreak :=G FALSE %% added 24 May 89
  \endgroup
END

```

End of historical L^AT_EX 2.09 comments.

```

149 \def\@starttoc#1{%
150   \begingroup
151   \makeatletter
152   \@input{\jobname.#1}%
153   \if@filesw
154     \expandafter\newwrite\csname tf@#1\endcsname
155     \immediate\openout \csname tf@#1\endcsname \jobname.#1\relax
156   \fi
157   \@nobreakfalse
158   \endgroup}

```

(End definition for \@starttoc.)

\addcontentsline The `\addcontentsline{<table>}{<type>}{<entry>}` command allows the user to add his/her own entry to a table of contents, etc. The command adds the entry `\contentsline{<type>}{<entry>}{<page>}` to the `.<table>` file.

This macro is implemented as an application of `\addtocontents`. Note that `\thepage` is not expandable during `\protected@write` therefore one gets the page number at the time of the `\shipout`.

```

159 </2kernel>
160 < *2kernel | latexrelease>
161 < latexrelease> \IncludeInRelease{2020/10/01}%
162 < latexrelease> { \addcontentsline}{fourth argument}%
163 \def\addcontentsline#1#2#3{%

```

We add an empty brace pair at the end of `\contentsline` so that the number of argument is identical in documents with and without hyperref.

```

164   \addtocontents{#1}{\protect\contentsline{#2}{#3}{\thepage}}}%
165   \protected@file@percent}}
166 </2kernel | latexrelease>
167 < latexrelease> \EndIncludeInRelease
168 < latexrelease> \IncludeInRelease{2018/12/01}%
169 < latexrelease> { \addcontentsline}{Mask line endings}%
170 < latexrelease> \def\addcontentsline#1#2#3{%
171 < latexrelease> \addtocontents{#1}{\protect\contentsline{#2}{#3}{\thepage}}%

```

We add `\protected@file@percent` at the end which is turned inside `\@writefile` into a percent character to mask the newline after the closing argument brace.

```

172 < latexrelease> \protected@file@percent}}
173 < latexrelease> \EndIncludeInRelease
174 < latexrelease> \IncludeInRelease{0000/00/00}%
175 < latexrelease> { \addcontentsline}{Mask line endings}%

```

```

176 <latexrelease>\def\addcontentsline#1#2#3{%
177 <latexrelease> \addtocontents{#1}{\protect\contentsline{#2}{#3}{\thepage}}}
178 <latexrelease>\EndIncludeInRelease
179 <*2ekernel>

```

(End definition for \addcontentsline.)

\addtocontents The `\addtocontents{<table>}{<text>}` command adds `<text>` to the `.<table>` file, with no page number.

```

180 \long\def\addtocontents#1#2{%
181   \protected@write\@auxout
182     {\let\label\@gobble \let\index\@gobble \let\glossary\@gobble}%
183     {\string\@writefile{#1}{#2}}}

```

(End definition for \addtocontents.)

\contentsline The `\contentsline{<type>}{<entry>}{<page>}` macro produces a `<type>` entry in a table of contents, etc. It will appear in the `.toc` or other file. For example, The entry for subsection 1.4.3 in the table of contents for example, might be produced by:

```

\contentsline{subsection}
  {\makebox{30pt}[r]{1.4.3} Gnats and Gnus}{22}

```

The `\protect` command causes command sequences to be written without expanding them.

```

184 \def\contentsline#1{\csname l@#1\endcsname}

```

(End definition for \contentsline.)

`\@dottedtocline{<level>}{<indent>}{<numwidth>}{<title>}{<page>}`: Macro to produce a table of contents line with the following parameters:

level If `<level> > \c@tocdepth`, then no line produced.

indent Total indentation from the left margin.

numwidth Width of box for number if the `<title>` has a `\numberline` command. As of 25 Jan 1988, this is also the amount of extra indentation added to second and later lines of a multiple line entry.

title Contents of entry.

page Page number.

Uses the following parameters, which must be set by the document style. They should be defined with `\def`'s.

pnumwidth Width of box in which page number is set.

tocrmarg Right margin indentation for all but last line of multiple-line entries.

dotsep Separation between dots, in mu units. Should be `\def`'d to a number like 2 or 1.7

`\@dottedtocline`

```

185 </2ekernel>
186 <*2ekernel | latexrelease>
187 <latexrelease>\IncludeInRelease{2018/12/01}%
188 <latexrelease>          {\@dottedtocline}{Prevent protrusion}%
189 \def\@dottedtocline#1#2#3#4#5{%
190   \ifnum #1>\c@tocdepth \else
191     \vskip \z@ \@plus.2\p@
192     {\leftskip #2\relax \rightskip \@tocrmarg \parfillskip -\rightskip
193     \parindent #2\relax\@afterindenttrue
194     \interlinepenalty\@M
195     \leavevmode
196     \@tempdima #3\relax

197     \advance\leftskip \@tempdima \null\nobreak\hskip -\leftskip
198     {#4}\nobreak
199     \leaders\hbox{$\m@th

```

If a document uses fonts other than computer modern, the use of a dot from math can be very disturbing despite the fact that this might be the only place in a document that then uses computer modern. Therefore we surround the dot with an `\hbox` to escape to the surrounding text font.

```

200       \mkern \@dotsep mu\hbox{.}\mkern \@dotsep
201       mu$}\hfill
202       \nobreak
203       \hb@xt@\@pnumwidth{\hfil\normalfont \normalcolor #5%

```

We finish off by preventing any protrusion if that is enabled. If protrusion happens the number may shift to the right and as a result you may end up with an additional dot in the toc line in some situations.

```

204               \kern-\p@\kern\p@}%
205   \par}%
206 \fi}

```

(End definition for \@dottedtocline.)

\noprotrusion This command, if placed directly to the right (or left) of a word, will prevent protrusion of that word into the margin. It is used in the toc entry lines as they shouldn't protrude. It is implemented as to kerns that cancel each other but being there hide the word so that protrusion is not added. Note that a zero kern or an empty box would not work as the protrusion mechanism will skip over those.

```

207 \DeclareRobustCommand\noprotrusion{\leavevmode\kern-\p@\kern\p@}

```

(End definition for \noprotrusion.)

```

208 </2ekernel | latexrelease>
209 <latexrelease>\EndIncludeInRelease
210 <latexrelease>\IncludeInRelease{0000/00/00}%
211 <latexrelease>          {\@dottedtocline}{Prevent protrusion}%
212 <latexrelease>\def\@dottedtocline#1#2#3#4#5{%
213 <latexrelease>   \ifnum #1>\c@tocdepth \else
214 <latexrelease>     \vskip \z@ \@plus.2\p@
215 <latexrelease>     {\leftskip #2\relax \rightskip \@tocrmarg \parfillskip -\rightskip
216 <latexrelease>     \parindent #2\relax\@afterindenttrue
217 <latexrelease>     \interlinepenalty\@M

```

```

218 <latexrelease> \leavevmode
219 <latexrelease> \@tempdima #3\relax
220 <latexrelease> \advance\leftskip \@tempdima \null\nobreak\hskip -\leftskip
221 <latexrelease> {#4}\nobreak
222 <latexrelease> \leaders\hbox{$\m@th
223 <latexrelease> \mkern \@dotsep mu\hbox{.}\mkern \@dotsep
224 <latexrelease> mu$}\hfill
225 <latexrelease> \nobreak
226 <latexrelease> \hb@xt@\@pnumwidth{\hfil\normalfont \normalcolor #5}%
227 <latexrelease> \par}%
228 <latexrelease> \fi}
229 <latexrelease>
230 <latexrelease>\let\noprotrusion\@undefined
231 <latexrelease>\EndIncludeInRelease
232 <*2kernel>

```

Note: `\nobreak`'s added 7 Jan 86 to prevent bad line break that left the page number dangling by itself at left edge of a new line.

Changed 25 Jan 88 to use `\leftskip` instead of `\hangindent` so leaders of multiple-line contents entries would line up properly.

`\numberline` `\numberline{<number>}`: For use in a `\contentsline` command. It puts `<number>` flush-left in a box of width `\@tempdima` (Before 25 Jan 88 change, it also added `\@tempdima` to the hanging indentation.)

```

233 \def\numberline#1{\hb@xt@\@tempdima{#1\hfil}}
234 </2kernel>

```

(End definition for `\numberline`.)

File O

ltfloat.dtx

1 Floats

The different types of floats are identified by a $\langle type \rangle$ name, which is the name of the counter for that kind of float. For example, figures are of type ‘figure’ and tables are of type ‘table’. Each $\langle type \rangle$ has associated a positive $\langle type\ number \rangle$, which is a power of two. E.g., figures might have type number 1, tables type number 2, programs type number 4, etc.

The locations where a float can go are specified by a $\langle placement\ specifier \rangle$, which is a list of the possible locations, each denoted by a letter as follows:

h : here	— at the current location in the text.
t : top	— at the top of a text page.
b : bottom	— at the bottom of a text page.
p : page	— on a separate float page

In addition, in conjunction with these, you can use ‘!’ which means that the current values of the float positioning parameters are ignored for this float. (Has no effect on ‘p’, float page positioning.) For example, ‘pht’ specifies that the float can appear in any of three locations: page, here or top.

1.1 Floating Environments

```
1 \*2ekernel
2 \message{floats,}
```

Historical L^AT_EX 2.09 comments (not necessarily accurate any more):

Where floats may appear on a page, and how many may appear there are specified by the following float placement parameters. The numbers are named like counters so the user can set them with the ordinary counter-setting commands.

<code>\c@topnumber</code>	: Number of floats allowed at the top of a column.
<code>\topfraction</code>	: Fraction of column that can be devoted to floats.
<code>\c@dbltopnumber, \dbltopfraction</code>	: Same as above, but for double-column floats.
<code>\c@bottomnumber, \bottomfraction</code>	: Same as above for bottom of page.
<code>\c@totalnumber</code>	: Number of floats allowed in a single column, including in-text floats.
<code>\textfraction</code>	: Minimum fraction of column that must contain text.
<code>\floatpagefraction</code>	: Minimum fraction of page that must be taken up by float page.
<code>\dblfloatpagefraction</code>	: Same as above, for double-column floats.

The document style must define the following.

`\fps@TYPE` : The default placement specifier for floats of type TYPE.

`\ftype@TYPE` : The type number for floats of type TYPE.

`\ext@TYPE` : The file extension indicating the file on which the contents list for float type TYPE is stored.
For example, `\ext@figure = 'lof'`.

`\fnum@TYPE` : A macro to generate the figure number for a caption.
For example, `\fnum@TYPE == Figure \thefigure`.

`\@makecaption{NUM}{TEXT}` :
A macro to make a caption, with NUM the value produced by `\fnum@...` and TEXT the text of the caption. It can assume it's in a `\parbox` of the appropriate width.

`\@float{TYPE}[PLACEMENT]` : This macro begins a float environment for a single-column float of type TYPE with PLACEMENT as the placement specifier. The default value of PLACEMENT is defined by `\fps@TYPE`. The environment is ended by `\end@float`.
E.g., `\figure == \@float{figure}, \endfigure == \end@float`.

```
\@float{TYPE}[PLACEMENT] ==
BEGIN
  if hmode then \@bsphack
    \@floatpenalty := -10002
  else \@floatpenalty := -10003
  fi
  \@captype ==L TYPE
  \@dblflset
  \@fps ==L PLACEMENT
  \@onelevel@sanitize \@fps
  add default PLACEMENT if at most ! in PLACEMENT ==
\@fpsadddefault
  if inner
    then LaTeX Error: 'Not in outer paragraph mode.'
    \@floatpenalty := 0
  else if \@freelist nonempty
    then \@currbox :=L head of \@freelist
    \@freelist :=G tail of \@freelist
    \count\@currbox :=G 32*\ftype@TYPE +
                                bits determined by PLACEMENT
    else \@floatpenalty := 0
    LaTeX Error: 'Too many unprocessed floats'
  fi
fi
\@currbox :=G \color@vbox
```

```

\normalcolor
\ vbox{
  %% 15 Dec 87 -
  %% removed \boxmaxdepth :=L 0pt
  %% that made box 0 depth because it screwed
  %% things up. Instead, added \vskip0pt at end
  \hsize = \columnwidth
  \@parboxrestore
  \@floatboxreset
END

```

```

\caption ==
BEGIN
  \refstepcounter{\@capttype}
  \@dblarg{\@caption{\@capttype}}
END

```

In following definition, `\par` moved from after `\addcontentsline` to before `\addcontentsline` because the `\write` could cause an extra blank line to be added to the paragraph above the caption. (Change made 12 Jun 87)

```

\@caption{TYPE}[STEXT]{TEXT} ==
BEGIN
  \par
  \addcontentsline{\ext@TYPE}{TYPE}{\numberline{\theTYPE}{STEXT}}
  \begingroup
    \@parboxrestore
    \@normalsize
    \makecaption{\fnum@TYPE}{TEXT}
  \par
  \endgroup
END

```

`\@dblfloat{TYPE}[PLACEMENT]` : Macro to begin a float environment for a double-column float of type TYPE with PLACEMENT as the placement specifier. The default value of PLACEMENT is 'tp'

The environment is ended by `\end@dblfloat`.

E.g., `\figure* == \@dblfloat{figure},`
`\endfigure* == \end@dblfloat.`

```

\@dblfloat{TYPE}[PLACEMENT] ==
  Identical to \@float{TYPE}[PLACEMENT] except \hsize and \linewidth
  are set to \textwidth.

```

End of historical L^AT_EX 2.09 comments.

`\@floatpenalty`

```

3 \newcount\@floatpenalty

```

(End definition for \@floatpenalty.)

`\caption` This is set to be an error message outside a float since no `captype` is defined there; this may need to be changed by some classes.

```

4 \def\caption{%
5   \ifx\@captype\@undefined
6     \latexerror{\noexpand\caption outside float}\@ehd
7     \expandafter\@gobble
8   \else
9     \refstepcounter\@captype
10    \expandafter\@firstofone
11  \fi
12  {\@dblarg{\@caption\@captype}}%
13 }

```

(End definition for `\caption`.)

`\@caption`

```

14 \long\def\@caption#1[#2]#3{%
15   \par
16   \addcontentsline{\csname ext@#1\endcsname}{#1}%
17   {\protect\numberline{\csname the#1\endcsname}{\ignorespaces #2}}%
18   \begingroup

```

The paragraph setting parameters are normalised at this point, however `\@parboxrestore` resets `\everypar` which is not correct in this context so `\@setminipage` is called if needed.

The float mechanism, like `minipage`, sets the flag `@minipage` true before executing the user-supplied text. Many L^AT_EX constructs test for this flag and do not add vertical space when it is true. The intention is that this emulates T_EX's ‘top of page’ behaviour. The flag must be set false at the start of the first paragraph. This is achieved by a redefinition of `\everypar`, but the call to `\@parboxrestore` removes that redefinition, so it is re-inserted if needed. If the flag is already false then the `\caption` was not the first entry in the float, and so some other paragraph has already activated the special `\everypar`. In this case no further action is needed.

```

19   \@parboxrestore
20   \if@minipage
21     \@setminipage
22   \fi
23   \normalsize
24   \@makecaption{\csname fnum@#1\endcsname}{\ignorespaces #3}\par
25   \endgroup}

```

(End definition for `\@caption`.)

`\@float`
`\@dblflset`

```

26 \def\@float#1{%
27   \@ifnextchar[%
28     {\@xfloat{#1}}%
29     {\edef\reserved@a{\noexpand\@xfloat{#1}[\csname fps@#1\endcsname]}}%
30     \reserved@a}

```

(End definition for `\@float` and `\@dblflset`.)

`\@dblfloat`

```

31 \def\@dblfloat{%
32   \if@twocolumn\let\reserved@a\@dblft\else\let\reserved@a\@float\fi
33   \reserved@a}

```

(End definition for \dblfloat.)

\fps@dbl Note that all double floats have default fps ‘tp’.

(End definition for \fps@dbl.)

\@setfps This sets the fps, dealing with error conditions by adding the default.

(End definition for \@setfps.)

\@xfloat The first part of this sets the count register that stores all the information about the type and fps of the float.

We assume here that the default specifiers already contain no active characters.

It may be better to store the defaults as numbers, rather than symbol strings.

```
34 </2ekernel>
35 <latexrelease>\IncludeInRelease{2015/01/01}%
36 <latexrelease>          {\@xfloat}{Check float options}%
37 <*2ekernel | latexrelease>
38 \def\@xfloat #1[#2]{%
39   \@nodocument
40   \def \@capytype {#1}%
41   \def \@fps {#2}%
42   \@onelevel@sanitize \@fps
43   \def \reserved@a {!}%
44   \ifx \reserved@a \@fps
45     \fpsadddefault
46   \else
47     \ifx \@fps \@empty
48       \fpsadddefault
49     \fi
50   \fi
51   \ifhmode
52     \bsphack
53     \@floatpenalty -\@Mii
54   \else
55     \@floatpenalty-\@Miii
56   \fi
57   \ifinner
58     \@parmoderr\@floatpenalty\z@
59   \else
60     \@next\@currbox\@freelist
61     {%
62       \@tempcnta \sixt@@n
63       \expandafter \@tfor \expandafter \reserved@a
64       \expandafter : \expandafter =\@fps
65       \do
```

Start of changes, use a nested if structure, ending in an error.

```
66       {%
67         \if \reserved@a h%
68           \ifodd \@tempcnta
69             \else
70               \advance \@tempcnta \@ne
71             \fi
```

```

72         \else\if \reserved@a t%
73             \@setfpsbit \tw@
74         \else\if \reserved@a b%
75             \@setfpsbit 4%
76         \else\if \reserved@a p%
77             \@setfpsbit 8%
78         \else\if \reserved@a !%
79             \ifnum \@tempcnta>15
80                 \advance\@tempcnta -\sixt@@n\relax
81             \fi
82         \else
83             \@latex@error{Unknown float option ‘\reserved@a’}%
84             {Option ‘\reserved@a’ ignored and ‘p’ used.}%
85             \@setfpsbit 8%
86         \fi\fi\fi\fi\fi
87     }%

```

End of changes

```

88         \@tempcntb \csname ftype@\@capttype \endcsname
89         \multiply \@tempcntb \@xxxii
90         \advance \@tempcnta \@tempcntb
91         \global \count\@currbox \@tempcnta
92     }%
93     \@fltovf
94 \fi

```

The remainder sets up the box in which the float is typeset, and the typesetting environment to be used. It is essential to have the extra box to avoid the unwanted space that would otherwise often be put at the top of the float.

It ends with a hook; not sure how useful this is but it is needed at present to deal with double-column floats.

```

95     \global \setbox\@currbox
96         \color@vbox
97         \normalcolor
98         \vbox \bgroup
99             \hsize\columnwidth
100             \@parboxrestore
101             \@floatboxreset
102     }%
103 </2ekernel | latexrelease>
104 <latexrelease>\EndIncludeInRelease
105 <latexrelease>\IncludeInRelease{0000/00/00}%
106 <latexrelease>          {\@xfloat}{Check float options}%
107 <latexrelease>\def\@xfloat #1[#2]{%
108 <latexrelease>    \@nodocument
109 <latexrelease>    \def \@capttype {#1}%
110 <latexrelease>    \def \@fps {#2}%
111 <latexrelease>    \@onelevel@sanitize \@fps
112 <latexrelease>    \def \reserved@b {!}%
113 <latexrelease>    \ifx \reserved@b \@fps
114 <latexrelease>        \@fpsadddefault
115 <latexrelease>    \else
116 <latexrelease>        \ifx \@fps \@empty
117 <latexrelease>            \@fpsadddefault

```

```

118 <latexrelease> \fi
119 <latexrelease> \fi
120 <latexrelease> \ifhmode
121 <latexrelease> \@bsphack
122 <latexrelease> \@floatpenalty -\@Mii
123 <latexrelease> \else
124 <latexrelease> \@floatpenalty-\@Miii
125 <latexrelease> \fi
126 <latexrelease> \ifinner
127 <latexrelease> \@parmoderr\@floatpenalty\z@
128 <latexrelease> \else
129 <latexrelease> \@next\@currbox\@freelist
130 <latexrelease> {%
131 <latexrelease> \@tempcnta \sixt@@n
132 <latexrelease> \expandafter \@tfor \expandafter \reserved@a
133 <latexrelease> \expandafter :\expandafter =\@fps
134 <latexrelease> \do
135 <latexrelease> {%
136 <latexrelease> \if \reserved@a h%
137 <latexrelease> \ifodd \@tempcnta
138 <latexrelease> \else
139 <latexrelease> \advance \@tempcnta \@ne
140 <latexrelease> \fi
141 <latexrelease> \fi
142 <latexrelease> \if \reserved@a t%
143 <latexrelease> \@setfpsbit \tw@
144 <latexrelease> \fi
145 <latexrelease> \if \reserved@a b%
146 <latexrelease> \@setfpsbit 4%
147 <latexrelease> \fi
148 <latexrelease> \if \reserved@a p%
149 <latexrelease> \@setfpsbit 8%
150 <latexrelease> \fi
151 <latexrelease> \if \reserved@a !%
152 <latexrelease> \ifnum \@tempcnta>15
153 <latexrelease> \advance\@tempcnta -\sixt@@n\relax
154 <latexrelease> \fi
155 <latexrelease> \fi
156 <latexrelease> }%
157 <latexrelease> \@tempcntb \csname ftype@\@capttype \endcsname
158 <latexrelease> \multiply \@tempcntb \@xxxii
159 <latexrelease> \advance \@tempcnta \@tempcntb
160 <latexrelease> \global \count\@currbox \@tempcnta
161 <latexrelease> }%
162 <latexrelease> \fltovf
163 <latexrelease> \fi
164 <latexrelease> \global \setbox\@currbox
165 <latexrelease> \color@vbox
166 <latexrelease> \normalcolor
167 <latexrelease> \vbox \bgroup
168 <latexrelease> \hsize\columnwidth
169 <latexrelease> \@parboxrestore
170 <latexrelease> \@floatboxreset
171 <latexrelease> }%

```

```

172 \latexrelease\EndIncludeInRelease
173 \*2ekernel)

```

(End definition for \@xfloat.)

\@floatboxreset The rationale for allowing these normally global flags to be set locally here, via **\@parboxrestore**, was stated originally by Donald Arseneau and extended by Chris Rowley. It is because these flags are only set globally to true by section commands, and these should never appear within marginals or floats or, indeed, in any group; and they are only ever set globally to false when they are definitely true.

If anyone is unhappy with this argument then both flags should be treated as in **\set@nobreak**; otherwise this command will be redundant.

```

174 \def \@floatboxreset {%
175     \reset@font
176     \normalsize
177     \@setminipage
178 }

```

(End definition for \@floatboxreset.)

\@setnobreak

```

179 \def \@setnobreak{%
180     \if@nobreak
181         \let\outer@nobreak\@nobreaktrue
182         \@nobreakfalse
183     \fi
184 }

```

(End definition for \@setnobreak.)

\@setminipage

```

185 \def \@setminipage{%
186     \@minipagetrue
187     \everypar{\@minipagefalse\everypar{}}%
188 }

```

(End definition for \@setminipage.)

\end@float

```

189 \def\end@float{%
190     \@endfloatbox
191     \ifnum\@floatpenalty <\z@

```

We make sure that we never exceed **\textheight**, otherwise float will never get typeset (91/03/15 FMi).

```

192     \@largefloatcheck
193     \@cons\@currlist\@currbox
194     \ifnum\@floatpenalty <-\@Mii
195         \penalty -\@Miv

```

Saving and restoring **\prevdepth** added 26 May 87 to prevent extra vertical space when used in vertical mode.

```

196     \@tempdima\prevdepth
197     \vbox{}%
198     \prevdepth\@tempdima

```

```

199     \penalty\@floatpenalty

200     \else
201     \vadjust{\penalty -\@Miv \vbox{}\penalty\@floatpenalty}\@Esphack
202     \fi
203     \fi
204 }

```

(End definition for \end@float.)

\end@dblfloat

```

205 </2ekernel>
206 <latexrelease>\IncludeInRelease{2015/01/01}%
207 <latexrelease>           {\end@dblfloat}{float order in 2-column}%
208 <*2ekernel | latexrelease>
209 \def\end@dblfloat{%
210     \if@twocolumn
211     \endfloatbox
212     \ifnum\@floatpenalty <\z@
213     \@largefloatcheck
214
215     Force the depth of two column float boxes.
216     \global\dp\@currbox1sp %
217
218     What follows is essentially \end@float without a starting \endfloatbox.
219
220     \@cons\@currlist\@currbox
221     \ifnum\@floatpenalty <-\@Mii
222     \penalty -\@Miv
223     \@tempdima\prevdepth
224     \vbox{}\%
225     \prevdepth\@tempdima
226     \penalty\@floatpenalty
227     \else
228     \vadjust{\penalty -\@Miv \vbox{}\penalty\@floatpenalty}\@Esphack
229     \fi
230
231     \fi
232     \else
233     \end@float
234     \fi
235 }%
236 </2ekernel | latexrelease>
237 <latexrelease>\EndIncludeInRelease
238 <latexrelease>\IncludeInRelease{0000/00/00}%
239 <latexrelease>           {\end@dblfloat}{float order in 2-column}%
240 <latexrelease>\def\end@dblfloat{%
241     \if@twocolumn
242     \endfloatbox
243     \ifnum\@floatpenalty <\z@

```

We make sure that we never exceed \textheight, otherwise float will never get typeset (91/03/15 FMi).

```

238 <latexrelease>     \@largefloatcheck
239 <latexrelease>     \@cons\@dbldeferlist\@currbox
240 <latexrelease>     \fi

```

RmS 92/03/18 changed \@esphack to \@Esphack.

```
241 \<latexrelease> \ifnum \@floatpenalty =-\@Mii \@Esphack\fi
242 \<latexrelease>\else
243 \<latexrelease> \end@float
244 \<latexrelease>\fi
245 \<latexrelease>\}%
246 \<latexrelease>\EndIncludeInRelease
247 \*2ekernel)
```

(End definition for \enddblfloat.)

\@endfloatbox This macro is not intended to be a hook; it is designed to help maintain the integrity of this code, which is used twice and, as can be seen, is subject to frequent changes.

```
248 \def \@endfloatbox{%
249     \par\vskip\z@skip      %% \par\vskip\z@ added 15 Dec 87

250     \@minipagefalse
251     \outer@nobreak
252     \egroup                %% end of vbox
253     \color@endbox
254 }
```

(End definition for \@endfloatbox.)

\outer@nobreak

```
255 \let\outer@nobreak\@empty
```

(End definition for \outer@nobreak.)

\@largefloatcheck This calculates by how much a float is oversize for the page and prints this in a warning message.

```
256 \def \@largefloatcheck{%
257     \ifdim \ht\@currbox>\textheight
258         \@tempdima -\textheight
259         \advance \@tempdima \ht\@currbox

260         \@latex@warning {Float too large for page by \the\@tempdima}%
261         \ht\@currbox \textheight
262     \fi
263 }
```

(End definition for \@largefloatcheck.)

\@dbflt

\@xdblfloat

```
264 \def\@dbflt#1{\@ifnextchar[{\@xdblfloat{#1}}{\@xdblfloat{#1}[tp]}}
265 \def\@xdblfloat#1[#2]{%
266     \@xfloat{#1}[#2]\hsize\textwidth\linewidth\textwidth}
```

(End definition for \@dbflt and \@xdblfloat.)

Moved to ltoutput 93/12/16

```
267 %\newcount\c@topnumber
268 %\newcount\c@dbltopnumber
269 %\newcount\c@bottomnumber
270 %\newcount\c@totalnumber
```

`\@floatplacement` An analysis of `\@floatplacement`:
This should be called whenever `\@colht` has been set.

```

271 \def\@floatplacement{\global\@topnum\c@topnumber
272   % Textpage bit, global:
273   \global\@toproom \topfraction\@colht
274   \global\@botnum \c@bottomnumber
275   \global\@botroom \bottomfraction\@colht
276   \global\@colnum \c@totalnumber
277   % Floatpage bit, local:
278   \@fpmin \floatpagefraction\@colht}
279 \endkernel

```

(End definition for `\@floatplacement`.)

`\@dblfloatplacement` This should be called only within a group. Now changed to provide extra checks in `\@addtodblcol`, needed when processing a BANG float.

```

280 \latexrelease\IncludeInRelease{2015/01/01}%
281 \latexrelease \@dblfloatplacement{float order in 2-column}%
282 \endkernel | latexrelease

```

When making two column float area, look for floats with 1sp depth.

```

283 \def\@dblfloatplacement{\global\@dbltopnum\c@dbltopnumber
284   \global\@dbltoproom \dbltopfraction\@colht
285   \@textmin \@colht
286   \advance \@textmin -\@dbltoproom
287   \@fpmin \dblfloatpagefraction\textheight
288   \fptop \@dblftop
289   \fpsep \@dblfpsep
290   \fpbot \@dblfpbot

```

`\f@depth` is used in `\@testwrongwidth` to look for either column or dbl-column floats. A value of 1sp signals the latter. Because of this setting here, `\@dblfloatplacement` needs to be called inside a group which is a questionable design.

```

291 \def\f@depth{1sp}%
292 \endkernel | latexrelease
293 \latexrelease\EndIncludeInRelease
294 \latexrelease\IncludeInRelease{0000/00/00}%
295 \latexrelease \@dblfloatplacement{float order in 2-column}%
296 \latexrelease\def \@dblfloatplacement {%

```

Textpage bit: global, but need not be.

```

297 \latexrelease \global \@dbltopnum \c@dbltopnumber
298 \latexrelease \global \@dbltoproom \dbltopfraction\@colht

```

This new bit uses `\@textmin` to locally store the amount of extra room in the column.

```

299 \latexrelease \@textmin \@colht
300 \latexrelease \advance \@textmin -\@dbltoproom

```

Floatpage bit: must be local.

```

301 \latexrelease \@fpmin \dblfloatpagefraction\textheight
302 \latexrelease \fptop \@dblftop
303 \latexrelease \fpsep \@dblfpsep
304 \latexrelease \fpbot \@dblfpbot
305 \latexrelease}%
306 \latexrelease\EndIncludeInRelease
307 \endkernel

```


(End definition for `\dblfloatplacement`.)

Historical *LaTeX* 2.09 comments (not necessarily accurate any more):

MARGINAL NOTES:

Marginal notes use the same mechanism as floats to communicate with the `\output` routine. Marginal notes are distinguished from floats by having a negative placement specification. The command `\marginpar [LTEXT]{RTEXT}` generates a marginal note in a parbox, using LTEXT if it's on the left and RTEXT if it's on the right. (Default is RTEXT = LTEXT.) It uses the following parameters.

`\marginparwidth` : Width of marginal notes.
`\marginparsep` : Distance between marginal note and text.
the page layout to determine how to move the marginal
note into the margin. E.g., `\@leftmargin skip ==`
`\hskip -\marginparwidth \hskip -\marginparsep` .
`\marginparpush` : Minimum vertical separation between `\marginpar`'s

Marginal notes are normally put on the outside of the page if `@mparswitch = true`, and on the right if `@mparswitch = false`. The command `\reversemarginpar` reverses the side where they are put. `\normalmarginpar` undoes `\reversemarginpar`. These commands have no effect for two-column output.

SURPRISE: if two marginal notes appear on the same line of text, then the second one could appear on the next page, in a funny position.

```
\marginpar [LTEXT]{RTEXT} ==
BEGIN
  if hmode then \bsphack
    \@floatpenalty := -10002
  else \@floatpenalty := -10003
  fi
  if inner
    then LaTeX Error: 'Not in outer paragraph mode.'
    \@floatpenalty := 0
  else if \@freelist has two elements:
    then get \@marbox, \@currbox from \@freelist
    \count\@marbox :=G -1
  else \@floatpenalty := 0
    LaTeX Error: 'Too many unprocessed floats'
    \@currbox, \@marbox := \@tempboxa %%use \def
  fi
fi
if optional argument
then %% \@xmpar ==
  \@savemarbox\@marbox{LTEXT}
  \@savemarbox\@currbox{RTEXT}
```

```

        else %% \@ympar ==
            \@savemarbox\@marbox{RTEXT}
            \box\@currbox :=G \box\@marbox
        fi
        \@xympar
    END

\reversemarginpar == BEGIN \@mparbottom :=G 0
                        @reversemargin :=G true
                        END

\normalmarginpar == BEGIN \@mparbottom :=G 0
                        @reversemargin :=G false
                        END

```

End of historical L^AT_EX 2.09 comments.

\marginpar

```

308 \def\marginpar{%
309   \ifhmode
310     \@bsphack
311     \@floatpenalty -\@Mii
312   \else
313     \@floatpenalty-\@Miii
314   \fi
315   \ifinner
316     \@parmoderr
317     \@floatpenalty\z@
318   \else
319     \@next\@currbox\@freelist{}\{}%
320     \@next\@marbox\@freelist{\global\count\@marbox\m@ne}%
321     {\@floatpenalty\z@
322       \@fltovf\def\@currbox{\@tempboxa}\def\@marbox{\@tempboxa}}%
323   \fi
324   \@ifnextchar [\@xmpar\@ympar}

```

(End definition for \marginpar.)

\@xmpar

```

325 \long\def\@xmpar[#1]#2{%
326   \@savemarbox\@marbox{#1}%
327   \@savemarbox\@currbox{#2}%
328   \@xympar}

```

(End definition for \@xmpar.)

\@ympar

```

329 \long\def\@ympar#1{%
330   \@savemarbox\@marbox{#1}%
331   \global\setbox\@currbox\copy\@marbox
332   \@xympar}

```

(End definition for \@ympar.)

`\@savemarbox`

```

333 </2ekernel>
334 <*2ekernel | latexrelease>
335 <latexrelease>\IncludeInRelease{2021/06/01}%
336 <latexrelease>          {\@savemarbox}{Explicit par for marginpar}%
337 \long\def \@savemarbox #1#2{%
338   \global\setbox #1%
339   \color@vbox
340   \vtop{%
341     \hsize\marginparwidth
342     \@parboxrestore
343     \@marginparreset
344     #2\par
345     \@minipagefalse
346     \outer@nobreak
347   }%
348   \color@endbox
349 }
350 </2ekernel | latexrelease>
351 <latexrelease>\EndIncludeInRelease
352 <latexrelease>\IncludeInRelease{0000/00/00}%
353 <latexrelease>          {\@savemarbox}{Explicit par for marginpar}%
354 <latexrelease>
355 <latexrelease>\long\def \@savemarbox #1#2{%
356 <latexrelease>  \global\setbox #1%
357 <latexrelease>    \color@vbox
358 <latexrelease>    \vtop{%
359 <latexrelease>      \hsize\marginparwidth
360 <latexrelease>      \@parboxrestore
361 <latexrelease>      \@marginparreset
362 <latexrelease>      #2%
363 <latexrelease>      \@minipagefalse
364 <latexrelease>      \outer@nobreak
365 <latexrelease>      }%
366 <latexrelease>    \color@endbox
367 <latexrelease>}
368 <latexrelease>\EndIncludeInRelease
369 <*2ekernel>

```

(End definition for \@savemarbox.)

`\@marginparreset` The rationale for allowing these normally global flags to be set locally here, via `\@parboxrestore` was stated originally by Donald Arseneau and extended by Chris Rowley. It is because these flags are only set globally to true by section commands, and these should never appear within marginals or floats or, indeed, in any group; and they are only ever set globally to false when they are definitely true.

If anyone is unhappy with this argument then both flags should be treated as in `\set@nobreak`; otherwise this command will be redundant.

```

370 \def \@marginparreset {%
371   \reset@font
372   \normalsize
373 %   \let@if@nobreak\iffalse
374 %   \let@if@noskipsec\iffalse

```

```

375 %      \@setnobreak
376      \@setminipage
377 }

(End definition for \@marginparreset.)

```

`\@xympar`

Setting the box here is done only because the code uses `\end@float`; it will be empty and gets discarded.

```

378 \def \@xympar{%
379   \ifnum \@floatpenalty < \z@ \@cons \@currlist \@marbox \fi
380   \setbox \@tempboxa
381     \color@vbox
382     \vbox \bgroup
383   \end@float
384   \@ignorefalse
385   \@esphack
386 }

(End definition for \@xympar.)

```

`\reversemarginpar`

`\normalmarginpar`

```

387 \def \reversemarginpar {\global \@parbottom \z@ \@reversemargintrue}
388 \def \normalmarginpar {\global \@parbottom \z@ \@reversemarginfalse}

(End definition for \reversemarginpar and \normalmarginpar.)

389 \message{footnotes,}

```

1.2 Footnotes

Historical L^AT_EX 2.09 comments (not necessarily accurate any more):

`\footnote{NOTE}` : User command to insert a footnote.

`\footnote[NUM]{NOTE}`: User command to insert a footnote numbered *NUM*, where *NUM* is a number – 1, 2, etc. For example, if footnotes are numbered *, **, etc. within pages, then `\footnote[2]{...}` produces footnote ‘**’. This command does not step the footnote counter.

`\footnotemark[NUM]` : Command to produce just the footnote mark in the text, but no footnote. With no argument, it steps the footnote counter before generating the mark.

`\footnotetext[NUM]{TEXT}` : Command to produce the footnote but no mark. `\footnote` is equivalent to `\footnotemark \footnotetext` .

As in PLAIN, footnotes use `\insert\footins`, and the following parameters:

`\footnotesize` : Size-changing command for footnotes.

`\footnotesep` : The height of a strut placed at the beginning of every footnote.

`\skip\footins` : Space between main text and footnotes. The rule separating footnotes from text occurs in this space. This space lies above the strut of height `\footnotesep` which is at the beginning of the first footnote.

`\footnoterule` : Macro to draw the rule separating footnotes from text. It is executed right after a `\vspace` of `\skip\footins`. It should take zero vertical space—i.e., it should to a negative skip to compensate for any positive space it occupies. (See PLAIN.TEX.)

`\interfootnotelinepenalty` : Interline penalty for footnotes.

`\thefootnote` : In usual LaTeX style, produces the footnote number. If footnotes are to be numbered within pages, then the document style file must include an `\@addtoreset` command to cause the footnote counter to be reset when the page counter is stepped. This is not a good idea, though, because the counter will not always be reset in time to ensure that the first footnote on a page is footnote number one.

`\@thefnmark` : Holds the current footnote's mark—e.g., `\dag` or `'1'` or `'a'`.

`\@mpfnnumber` : A macro that generates the numbers for `\footnote` and `\footnotemark` commands. It == `\thefootnote` outside a minipage environment, but can be changed inside to generate numbers for `\footnote`'s.

`\@makefnmark` : A macro to generate the footnote marker from `\@thefnmark`. The default definition was `\hbox{$\@thefnmark$}`.

This is now replaced by
`\@thefnmark`

`\@makefntext{NOTE}` :
 Must produce the actual footnote, using `\@thefnmark` as the mark of the footnote and `NOTE` as the text. It is called when effectively inside a `\parbox`, with `\hsize = \columnwidth`.
 For example, it might be as simple as
`$\@thefnmark$ NOTE`

In a minipage environment, `\footnote` and `\footnotetext` are redefined so that

- (a) they use the counter `mpfootnote`
- (b) the footnotes they produce go at the bottom of the minipage.

The switch is accomplished by letting `\@mpfn == footnote` or `mpfootnote` and `\thempfn == \thefootnote` or `\thempfootnote`, and by redefining `\@footnotetext` to be `\@mpfootnotetext` in the minipage.

```

\footnote{NOTE} ==
BEGIN
  \stepcounter{\@mpfn}
  begingroup
    \protect == \noexpand
    \@thefnmark :=G eval (\thempfn)
  endgroup
  \@footnotemark
  \@footnotetext{NOTE}
END

\footnote[NUM]{NOTE} ==
BEGIN
  begingroup
    \protect == \noexpand
    counter \@mpfn :=L NUM
    \@thefnmark :=G eval (\thempfn)
  endgroup
  \@footnotemark
  \@footnotetext{NOTE}
END

\footnotemark ==
BEGIN \stepcounter{footnote}
  begingroup
    \protect == \noexpand
    \@thefnmark :=G eval(\thefootnote)
  endgroup
  \@footnotemark
END

\footnotemark[NUM] ==
BEGIN
  begingroup
    footnote counter :=L NUM
    \protect == \noexpand
    \@thefnmark :=G eval(\thefootnote)
  endgroup
  \@footnotemark
END

```

```

\@footnotemark ==
  BEGIN
    \leavevmode
    IF hmode THEN \@x@sf := \the\spacefactor FI
    \@makefnmark          % put number in main text
    IF hmode THEN \spacefactor := \@x@sf FI
  END

\footnotetext      ==
  BEGIN begingroup \protect == \noexpand
    \@thefnmark :=G eval (\thempfn)
  endgroup
  \@footnotetext
  END

\footnotetext[ NUM ] ==
  BEGIN begingroup counter \@mpfn :=L NUM
    \protect == \noexpand
    \@thefnmark :=G eval (\thempfn)
  endgroup
  \@footnotetext
  END

```

End of historical L^AT_EX 2.09 comments.

`\footins` L^AT_EX does use the same insert for footnotes as PLAIN.

```

390 \newinsert\footins

```

L^AT_EX leaves these initializations for the `\footins` insert.

```

391 \skip\footins=\bigskipamount % space added when footnote is present
392 \count\footins=1000 % footnote magnification factor (1 to 1)
393 \dimen\footins=8in % maximum footnotes per page

```

(End definition for \footins.)

`\footnoterule` L^AT_EX keeps PLAIN T_EX's `\footnoterule` as the default.

```

394 \def\footnoterule{\kern-3\p@
395   \hrule \@width 2in \kern 2.6\p@} % the \hrule is .4pt high

```

(End definition for \footnoterule.)

`\thefootnote`

```

396 \@definecounter{footnote}
397 \def\thefootnote{\@arabic\c@footnote}

```

(End definition for \thefootnote.)

`\thempfootnote` The default display for the footnote counter in minipages is to use italic letters. We use `\itshape` not `\textit` as the latter would add an italic correction.

```

398 \@definecounter{mpfootnote}
399 \def\thempfootnote{{\itshape\@alph\c@mpfootnote}}

```

(End definition for \thempfootnote.)

`\@makefnmark` Default definition.

```
400 %\def\@makefnmark{\hbox{$\sim\@thefnmark}\m@th$}%
401 \def\@makefnmark{\hbox{\@textsuperscript{\normalfont\@thefnmark}}}
```

(End definition for \@makefnmark.)

`\textsuperscript` This command provides superscript characters in the current text font. It's implementation might change!!!

```
402 \DeclareRobustCommand*\textsuperscript[1]{%
403   \@textsuperscript{\selectfont#1}}
```

(End definition for \textsuperscript.)

`\@textsuperscript` This command should not be used directly, but may be used to define other commands `\textsuperscript`, `\@makefnmark`. #1 should always start with a font selection command, to activate the font size switch.

```
404 \</2ekernel>
405 \<*2ekernel | latexrelease>
406 \<latexrelease>\IncludeInRelease{2020/10/01}%
407 \<latexrelease>           {\@textsuperscript}{superscript baseline}%
408 \def\@textsuperscript#1{%
409   {\m@th\ensuremath{\sim\hbox{\fontsize\sfontsize\sfontsize#1}}}}
410 \</2ekernel | latexrelease>
411 \<latexrelease>\EndIncludeInRelease
412 \<latexrelease>\IncludeInRelease{0000/00/00}%
413 \<latexrelease>           {\@textsuperscript}{superscript baseline}%
414 \<latexrelease>
415 \<latexrelease>\def\@textsuperscript#1{%
416 \<latexrelease>   {\m@th\ensuremath{\sim\hbox{\fontsize\sfontsize\z@#1}}}}
417 \<latexrelease>\EndIncludeInRelease
418 \<*2ekernel>
```

(End definition for \@textsuperscript.)

`\textsubscript`

```
419 \</2ekernel>
420 \<latexrelease>\IncludeInRelease{2015/01/01}%
421 \<latexrelease>           {\textsubscript}{\textsubscript}%
422 \<*2ekernel | latexrelease>
423 \DeclareRobustCommand*\textsubscript[1]{%
424   \@textsubscript{\selectfont#1}}%
425 \</2ekernel | latexrelease>
426 \<latexrelease>\EndIncludeInRelease
427 \<latexrelease>\IncludeInRelease{0000/00/00}%
428 \<latexrelease>           {\textsubscript}{\textsubscript}%
429 \<latexrelease>\let\textsubscript\undefined
430 \<latexrelease>\EndIncludeInRelease
431 \<*2ekernel>
```

(End definition for \textsubscript.)

`\@textsubscript`

```
432 </2ekernel>
433 <*2ekernel | latexrelease>
434 <latexrelease>\IncludeInRelease{2020/10/01}%
435 <latexrelease>          {\@textsubscript}{subscript baseline}%
436 \def\@textsubscript#1{%
437   {\m@th\ensuremath{_{\mbox{\fontsize\sf@size\sf@size#1}}}}
438 </2ekernel | latexrelease>
439 <latexrelease>\EndIncludeInRelease

440 <latexrelease>\IncludeInRelease{2015/01/01}%
441 <latexrelease>          {\@textsubscript}{subscript baseline}%
442 <latexrelease>
443 <latexrelease>\def\@textsubscript#1{%
444 <latexrelease>   {\m@th\ensuremath{_{\mbox{\fontsize\sf@size\z@#1}}}}
445 <latexrelease>\EndIncludeInRelease
446 <latexrelease>\IncludeInRelease{0000/00/00}%
447 <latexrelease>          {\@textsubscript}{subscript baseline}%
448 <latexrelease>\let\@textsubscript\undefined
449 <latexrelease>\EndIncludeInRelease
450 <*2ekernel>

(End definition for \@textsubscript.)
```

`\footnotesep`

```
451 \newdimen\footnotesep

(End definition for \footnotesep.)
```

`\footnote`

```
452 \def\footnote{\@ifnextchar[\@xfootnote{\stepcounter\@mpfn
453   \protected@xdef\@thefnmark{\thempfn}%
454   \@footnotemark\@footnotetext}}

(End definition for \footnote.)
```

`\@xfootnote`

```
455 \def\@xfootnote[#1]{%
456   \begingroup
457   \csname c@\@mpfn\endcsname #1\relax
458   \unrestored@protected@xdef\@thefnmark{\thempfn}%
459   \endgroup
460   \@footnotemark\@footnotetext}

(End definition for \@xfootnote.)
```

`\@footnotetext`

```
461 </2ekernel>
462 <*2ekernel | latexrelease>
463 <latexrelease>\IncludeInRelease{2021/06/01}%
464 <latexrelease>          {\@footnotetext}{footnotetext tagging}%
465 \long\def\@footnotetext#1{\insert\footins{%
466   \reset@font\footnotesize
467   \interlinepenalty\interfootnotelinepenalty
468   \splittopskip\footnotesep
```

```

469 \splitmaxdepth \dp\strutbox \floatingpenalty \@MM
470 \hsize\columnwidth \@parboxrestore
471 \protected@edef\@currentlabel{%
472 \csname p@footnote\endcsname\@thefnmark
473 }%
474 \color@begingroup
475 \makefnintext{%
476 \rule\z@\footnotesep\ignorespaces#1\@finalstrut\strutbox}%
477 \par
478 \color@endgroup}}%
479 \endkernel\latexrelease)
480 \EndIncludeInRelease
481 \IncludeInRelease{0000/00/00}%
482 \latexrelease\@footnotetext\footnotetext tagging}%
483 \latexrelease
484 \long\def\@footnotetext#1{\insert\footins{%
485 \latexrelease\reset@font\footnotesize
486 \latexrelease\interlinepenalty\interfootnotelinepenalty
487 \latexrelease\splittopskip\footnotesep
488 \latexrelease\splitmaxdepth \dp\strutbox \floatingpenalty \@MM
489 \latexrelease\hsize\columnwidth \@parboxrestore
490 \latexrelease\protected@edef\@currentlabel{%
491 \latexrelease\csname p@footnote\endcsname\@thefnmark
492 \latexrelease}%
493 \latexrelease\color@begingroup
494 \latexrelease\makefnintext{%
495 \latexrelease\rule\z@\footnotesep\ignorespaces#1\@finalstrut\strutbox}%
496 \latexrelease\color@endgroup}}%
497 \latexrelease
498 \EndIncludeInRelease
499 \endkernel)

```

(End definition for \@footnotetext.)

\footnotemark

```

500 \def\footnotemark{%
501 \ifnextchar[\@xfootnotemark
502 {\stepcounter{footnote}%
503 \protected@xdef\@thefnmark{\thefootnote}%
504 \@footnotemark}}

```

(End definition for \footnotemark.)

\@xfootnotemark

```

505 \def\@xfootnotemark[#1]{%
506 \begingroup
507 \c@footnote #1\relax
508 \unrestored@protected@xdef\@thefnmark{\thefootnote}%
509 \endgroup
510 \@footnotemark}

```

(End definition for \@xfootnotemark.)

`\@footnotemark`

```
511 \def\@footnotemark{%
512   \leavevmode
513   \ifhmode\edef\x@sf{\the\spacefactor}\nobreak\fi
514   \@makefnmark
515   \ifhmode\spacefactor\x@sf\fi
516   \relax}
```

(End definition for \@footnotemark.)

`\footnotetext`

```
517 \def\footnotetext{%
518   \ifnextchar [\xfootnotenext
519   {\protected@xdef\@thefnmark{\thempfn}%
520   \@footnotetext}}
```

(End definition for \footnotetext.)

`\xfootnotenext`

```
521 \def\xfootnotenext[#1]{%
522   \begingroup
523   \csname c@\mpfn\endcsname #1\relax
524   \unrestored@protected@xdef\@thefnmark{\thempfn}%
525   \endgroup
526   \@footnotetext}
```

(End definition for \xfootnotenext.)

`\thempfn`

`\@mpfn`

```
527 \def\@mpfn{footnote}
528 \def\thempfn{\thefootnote}
```

(End definition for \thempfn and \@mpfn.)

`\footref` This command generates a footnote mark. The value is produced by referencing a `\label` placed into a `\footnote` elsewhere (can be one in the main galley or in a minipage).

```
529 </2ekernel>
530 <*2ekernel | latexrelease>
531 <latexrelease>\IncludeInRelease{2021/06/01}%
532 <latexrelease>          {\footref}{Add footref}%
533 \def\footref#1{%
534   \begingroup
535   \unrestored@protected@xdef\@thefnmark{\ref{#1}}%
536   \endgroup
537   \@footnotemark
538 }
539 </2ekernel | latexrelease>
540 <latexrelease>\EndIncludeInRelease
```

We don't remove it when rolling back so that packages offered it in the past do not need to alter their behavior in a rollback situation.

```
541 <latexrelease>\IncludeInRelease{0000/00/00}%
542 <latexrelease>          {\footref}{Add footref}%
543 <latexrelease>
544 <latexrelease> % \let\footref\@undefined
```

```

545 <latexrelease>
546 <latexrelease>\EndIncludeInRelease
547 <*2ekernel>

(End definition for \footref.)

548 </2ekernel>

```

File P

ltidxglo.dtx

1 Index and Glossary Generation

Index and Glossary commands.

<code>\makeindex</code>	A preamble command to turn on indexing.
<code>\makeglossary</code>	A preamble command to turn on making glossary entries.
<code>\index</code>	Make an index entry for #1.
<code>\glossary</code>	Make a glossary entry for #1.

Historical L^AT_EX 2.09 comments (not necessarily accurate any more):

```

\makeindex ==
  BEGIN
    \index == BEGIN \@bsphack
              \begingroup
              \protect{X} == \string X\space
              %% added 3 Feb 87 for \index commands
              %% in \footnotes
              re-\catcode special characters
              to 'other'
              \@wrindex

  END

\@wrindex{ITEM} ==
  BEGIN
    write of {\indexentry{ITEM}{page number}}
  \endgroup
  \@esphack
  END

INITIALIZATION:

\index == BEGIN \@bsphack
          \begingroup
          re-\catcode special characters (in case '%' there)
          \@index

  END

\@index{ITEM} == BEGIN \endgroup \@esphack END

```

Changes made 14 Apr 89 to write `\glossaryentry`'s instead of
`\indexentry`'s on the .glo file.

End of historical L^AT_EX 2.09 comments.

```

1 \<*2ekernel)
2 \message{index,}

```

`\makeindex`

```
3 \def\makeindex{%
4   \newwrite\@indexfile
5   \immediate\openout\@indexfile=\jobname.idx
6   \def\index{\@bsphack\begingroup
7     \@sanitize
8     \@wrindex}\typeout
9     {Writing index file \jobname.idx}}%
```

Opening the write channel should be done only once since on some OS multiple opens are forbidden and in any case it is useless. So we turn this into a no-op after use.

```
10   \let\makeindex\@empty
11 }
12 \@onlypreamble\makeindex
```

(End definition for \makeindex.)

`\@wrindex`

```
13 \def\@wrindex#1{%
14   \protected@write\@indexfile{%
15     {\string\indexentry{#1}{\thepage}}%
16   \endgroup
17   \@esphack}
```

(End definition for \@wrindex.)

`\index`

```
18 \def\index{\@bsphack\begingroup \@sanitize\@index}
```

(End definition for \index.)

`\@index`

```
19 \def\@index#1{\endgroup\@esphack}
```

(End definition for \@index.)

`\makeglossary`

```
20 \def\makeglossary{%
21   \newwrite\@glossaryfile
22   \immediate\openout\@glossaryfile=\jobname.glo
23   \def\glossary{\@bsphack\begingroup
24     \@sanitize
25     \@wrglossary}\typeout
26     {Writing glossary file \jobname.glo }}%
```

Opening the write channel should be done only once since on some OS multiple opens are forbidden and in any case it is useless. So we turn this into a no-op after use.

```
27   \let\makeglossary\@empty
28 }
29 \@onlypreamble\makeglossary
```

(End definition for \makeglossary.)

`\@wrglossary`

```
30 \def\@wrglossary#1{%  
31   \protected@write\@glossaryfile{%  
32     {\string\glossaryentry{#1}{\thepage}}}%  
33   \endgroup  
34   \@esphack}
```

(End definition for \@wrglossary.)

`\glossary`

```
35 \def\glossary{\@bsphack\begin@group\@sanitizel\@index}
```

(End definition for \glossary.)

```
36 \endkernel
```

File Q

ltbibl.dtx

1 Bibliography Generation

A bibliography is created by the `thebibliography` environment, which generates a title such as “References”, and a list of entries. The `BIBTEX` program will create a file containing such an environment, which will be read in by the `\bibliography` command. With `BIBTEX`, the following commands will be used.

<code>\bibliography</code>	<code>\bibliography{<file1,file2, ...,filen>}</code> : specifies the bibdata files. Writes a <code>\bibdata</code> entry on the <code>.aux</code> file and tries to read in <code>mainfile.bbl</code> .
<code>\bibliographystyle</code>	<code>\bibliographystyle{<style>}</code> : Writes a <code>\bibstyle</code> entry on the <code>.aux</code> file.
<code>thebibliography</code>	The <code>thebibliography</code> environment is a list environment. To save the use of an extra counter, it should use <code>enumiv</code> as the item counter. Instead of using <code>\item</code> , items in the bibliography are produced by the following commands:

`\bibitem{<name>}` : Produces a numbered entry cited as `<name>`.
`\bibitem[<label>]{<name>}` : Produces an entry labeled by `<Label>` and cited by `<name>`.
The former is used for bibliographies with citations like [1], [2], etc.; the latter is used for citations like [Knuth82].

The document class must define the `thebibliography` environment. This environment has a single argument, which is the widest bibliography label— e.g., if the [Knuth67] is the widest entry, then this argument will be Knuth67. The `\thebibliography` command must begin a list environment, which the `\endthebibliography` command ends.

<code>\cite</code>	Entries are cited by the command <code>\cite{<name>}</code> .
<code>\nocite</code>	<code>\nocite{< citations>}</code> puts information on the <code>.aux</code> file that causes <code>BIBTEX</code> to include the <code>{< citations>}</code> list in the bibliography, but puts nothing in the text. <code>\nocite{*}</code> is special: it tells <code>BIBTEX</code> to put the whole of a collection of references into the bibliography.

```

1 <*2ekernel>
2 \message{bibliography,}

```

Historical L^AT_EX 2.09 comments (not necessarily accurate any more):

PARAMETERS

`\@cite` : A macro such that `\@cite{LABEL1,LABEL2}{NOTE}` produces the output for a `\cite[NOTE]{FOO1,FOO2}` command, where entry `FOOi` is defined by `\bibitem[LABELi]{FOOi}`. The switch `@tempswa` is true if the optional `NOTE` argument is present.
The default definition is :

```

\@cite{LABELS}{NOTE} ==
  BEGIN [LABELS
    IF @tempswa = T THEN , NOTE FI
  ]
END

```

`\@biblabel` : A macro to produce the label in the bibliography entry. For `\bibitem[LABEL]{NAME}`, the label is

generated by \@biblabel{LABEL}. It has the default definition \@biblabel{LABEL} -> [LABEL].

CONVENTION

\b@F00 : The name or number of the reference created by \cite{FOO}
E.g., if \cite{FOO} -> [17] , then \b@F00 -> 17.

End of historical L^AT_EX 2.09 comments.

\bibitem

```
3 \def\bibitem{\@ifnextchar[\@lbibitem\bibitem}
```

(End definition for \bibitem.)

\@lbibitem

```
4 \def\@lbibitem[#1]#2{\item[\@biblabel{#1}\hfill]\if@filesw
5     {\let\protect\noexpand
6      \immediate
7      \write\@auxout{\string\bibcite{#2}{#1}}}\fi\ignorespaces}
```

(End definition for \@lbibitem.)

\@bibitem

```
8 \def\@bibitem#1{\item\if@filesw \immediate\write\@auxout
9     {\string\bibcite{#1}{\the\value{\@listctr}}}\fi\ignorespaces}
```

(End definition for \@bibitem.)

\bibcite

```
10 \def\bibcite{\@newl@bel b}
```

(End definition for \bibcite.)

\citation

```
11 \let\citation\@gobble
```

(End definition for \citation.)

\cite

```
12 \DeclareRobustCommand\cite{%
13     \@ifnextchar [{\@tempswatrue\citex}{\@tempswafalse\citex[]}]}
```

(End definition for \cite.)

\@citex \penalty\@m added to definition of \@citex to allow a line break after the ‘,’ in citations like [Jones80,Smith77] (Added 23 Oct 86)
space added after the ‘,’ (21 Nov 87)

```
14 \def\@citex[#1]#2{\leavevmode
15     \let\@citea\@empty
16     \@cite{\@for\@citeb:=#2\do
17         {\@citea\def\@citea{\penalty\@m\ }%
18          \edef\@citeb{\expandafter\@firstofone\@citeb\@empty}}%
19     \if@filesw\immediate\write\@auxout{\string\citation{\@citeb}}\fi}
```

Using `\hbox` instead of `\mbox` is fine because of the `\leavevmode` above. In fact the use of a box around the citation contents is more than questionable in my view (FMi), but within 2e I have to keep that for compatibility reasons as it would probably change too many existing documents. Its main reason is to avoid hyphenation of labels such as [FOOB89] into [FOO- B89] so in certain styles it makes sense; but, for example, in author year citations it becomes more than questionable.

So Chris added yet another hook here, as suggested by, at least, Donald Arseneau. Note that this one is inside the first argument of the `\@cite` hook. This decouples the top-level typesetting of the citation from the details of the other business conducted here. All this really needs a complete rethink to get the right modularity.

```

20      \@ifundefined{b@ \@citeb}{\hbox{\reset@font\bfseries ?}}%
21      \G@refundefinedtrue
22      \@latex@warning
23      {Citation ‘\@citeb’ on page \thepage \space undefined}}%
24      {\@cite@ofmt{\csname b@ \@citeb\endcsname}}}{#1}}

```

(End definition for \@citex.)

```

\bibdata
\bibstyle
25 \let\bibdata=\@gobble
26 \let\bibstyle=\@gobble

```

(End definition for \bibdata and \bibstyle.)

```

\bibliography
27 \def\bibliography#1{%
28   \if@filesw
29     \immediate\write\@auxout{\string\bibdata{\zap@space#1 \@empty}}%
30   \fi
31   \@input@{\jobname.bbl}}

```

(End definition for \bibliography.)

```

\bibliographystyle
32 \def\bibliographystyle#1{%
33   \ifx\@begindocumenthook\@undefined\else
34     \expandafter\AtBeginDocument
35   \fi
36   {\if@filesw
37     \immediate\write\@auxout{\string\bibstyle{#1}}%
38   \fi}}

```

(End definition for \bibliographystyle.)

`\nocite` (Added 14 Jun 85)

This puts information on the `.aux` file that causes `BIBTEX` to include the citation list in the bibliography, but puts nothing in the text.

RmS 93/08/06: Made loop for `\nocite` like that for `\@citex`, to get rid of leading spaces.

```

39 \</2ekernel>
40 \< *2ekernel | latexrelease>
41 \< latexrelease>\IncludeInRelease{2021/06/01}%
42 \< latexrelease>          {\nocite}{Allow nocite in preamble}%
43 \def\nocite#1{\@bsphack

```

With the implementation designed already in L^AT_EX 2.09 the `\nocite` command will not work before `\begin{document}` since it tries to write to the `.aux` file which is not open before that point. As a result the “reference” will appear on the terminal and nothing else will happen.

[This would be easy to fix, but then a document using the fix will silently fail on an older release of L^AT_EX, missing all citations done with `\nocite`. Thus we do only generate an error message and leave the fix for a L^AT_EX 2_ε successor.]

Given that we are now a quarter century into using L^AT_EX 2_ε there is no good reason any more do limit ourself to 2.09 considerations. So we now simply delay the `\nocite` if it is issued in the preamble.

```
44 \ifx\@onlypreamble\document
```

Since we are after `\begin{document}` we can do the citations:

```
45 \@for\@citeb:=#1\do{%
46   \edef\@citeb{\expandafter\@firstofone\@citeb}%
47   \if@files\immediate\write\@auxout{\string\citation{\@citeb}}\fi
48   \@ifundefined{b@\@citeb}{\G@refundefinedtrue
49     \latex@warning{Citation ‘\@citeb’ undefined}}{}%
50 \else
```

But before `\begin{document}` we raised an error message in the past but as of 2021/05 not any longer.

```
51 % \latex@error{Cannot be used in preamble}\@eha
```

Instead we delay the declaration to the start of the document. We have to use a late hook for this, so that it comes after the `.aux` file is open for writing and after `\@preamblecmds` was executed to change the above test. Therefore `\AtBeginDocument` would still be too early.

```
52 \AddToHook{begindocument/end}[kernel]{\nocite{#1}}%
53 \fi
54 \esphack}
55 </2kernel | latexrelease>
56 <latexrelease>\EndIncludeInRelease
57 <latexrelease>\IncludeInRelease{0000/00/00}%
58 <latexrelease> \nocite{Allow nocite in preamble}%
59 <latexrelease>
60 <latexrelease>\def\nocite#1{\@bsphack
61 <latexrelease> \ifx\@onlypreamble\document
62 <latexrelease> \@for\@citeb:=#1\do{%
63 <latexrelease> \edef\@citeb{\expandafter\@firstofone\@citeb}%
64 <latexrelease> \if@files\immediate\write\@auxout{\string\citation{\@citeb}}\fi
65 <latexrelease> \@ifundefined{b@\@citeb}{\G@refundefinedtrue
66 <latexrelease> \latex@warning{Citation ‘\@citeb’ undefined}}{}%
67 <latexrelease> \else
68 <latexrelease> \latex@error{Cannot be used in preamble}\@eha
69 <latexrelease> \fi
70 <latexrelease> \esphack}
71 <latexrelease>
72 <latexrelease>\EndIncludeInRelease
73 <*2kernel>
```

Since `\nocite{*}` should not produce a warning about undefined citation keys (see PR 557), we need to set the control sequence `‘\b@*’` to something other than `\relax`. As

a result `\cite{*}` will not warn either (but that never worked with \LaTeX in the first place).

```
74 \expandafter\let\csname b@*\endcsname\@empty
```

(End definition for \nocite.)

1.1 Default definitions

This hook determines the ‘relative formatting’ of the two logical parts of a citation with comment.

`\@cite`

```
75 \def\@cite#1#2{[{#1\if@tempswa , #2\fi}]}
```

(End definition for \@cite.)

`\@cite@ofmt`

This is, in general, a command that appears to have one argument whose value is, in the kernel, a single cs whose name is the expansion of `b@ \@citeb`; the expansion of this cs will typically be some hmode material that produces the detailed typeset form of just the citations themselves.

```
76 \let\@cite@ofmt\hbox
```

(End definition for \@cite@ofmt.)

`\@biblabel`

```
77 \def\@biblabel#1{[#1]}
```

```
78 \</2ekernel>
```

(End definition for \@biblabel.)

File R

ltpage.dtx

1 Page styles and related commands

1.1 Page Style Commands

`\pagestyle{<style>}` : sets the page style of the current and succeeding pages to *style*

`\thispagestyle{<style>}` : sets the page style of the current page only to *style*.

To define a page style *style*, you must define `\ps@style` to set the page style parameters.

1.2 How a page style makes running heads and feet

The `\ps@...` command defines the macros `\@oddhead`, `\@oddfoot`, `\@evenhead`, and `\@evenfoot` to define the running heads and feet. (See output routine.) To make headings determined by the sectioning commands, the page style defines the commands `\chaptermark`, `\sectionmark`, etc., where `\chaptermark{<text>}` is called by `\chapter` to set a mark. The `\...mark` commands and the `\...head` macros are defined with the help of the following macros.

(All the `\...mark` commands should be initialized to no-ops.)

1.3 marking conventions

L^AT_EX extends T_EX's `\mark` facility by producing two kinds of marks a 'left' and a 'right' mark, using the following commands:

`\markboth{<left>}{<right>}` : Adds both marks.

`\markright{<right>}` : Adds a 'right' mark.

`\leftmark` : Used in the output routine, gets the current 'left' mark. Works like T_EX's `\botmark`.

`\rightmark` : Used in the output routine, gets the current 'right' mark. Works like T_EX's `\firstmark`. The marking commands work reasonably well for right marks 'numbered within' left marks—e.g., the left mark is changed by a `\chapter` command and the right mark is changed by a `\section` command. However, it does produce somewhat anomalous results if 2 `\markboth`'s occur on the same page.

Commands like `\tableofcontents` that should set the marks in some page styles use a `\@mkboth` command, which is `\let` by the `\pagestyle` command (`\ps@...`) to `\markboth` for setting the heading or to `\@gobbletwo` to do nothing.

1 `<*2ekernel>`

`\pagestyle` User command to set the page style for this and following pages.

```
2 \def\pagestyle#1{%
3   \@ifundefined{ps@#1}%
4     \undefinedpagestyle
5     {\@nameuse{ps@#1}}}
```

(End definition for `\pagestyle`.)

`\thispagestyle` User command to set the page style for this page only.

```

6 \def\thispagestyle#1{%
7   \ifundefined{ps@#1}%
8     \undefinedpagestyle
9     {\global\@specialpagetrue\gdef\@specialstyle{#1}}

```

(End definition for `\thispagestyle`.)

`\ps@empty` The empty page style: No head or foot line.

```

10 \def\ps@empty{%
11   \let\@mkboth\@gobbletwo\let\@oddhead\@empty\let\@oddfoot\@empty
12   \let\@evenhead\@empty\let\@evenfoot\@empty}

```

(End definition for `\ps@empty`.)

`\ps@plain` The plain page style: No head, centred page number in foot.

```

13 \def\ps@plain{\let\@mkboth\@gobbletwo
14   \let\@oddhead\@empty\def\@oddfoot{\reset@font\hfil\thepage
15   \hfil}\let\@evenhead\@empty\let\@evenfoot\@oddfoot}

```

(End definition for `\ps@plain`.)

`\@leftmark` We implement `\@leftmark` and `\@rightmark` in terms of already defined commands to
`\@rightmark` save token space. We can't get rid of them since they are sometimes used in applications.

```

16 \let\@leftmark\@firstoftwo
17 \let\@rightmark\@secondoftwo

```

(End definition for `\@leftmark` and `\@rightmark`.)

```

18 </2ekernel>
19 <*2ekernel | latexrelease>
20 <latexrelease>\IncludeInRelease{2019/10/01}%
21 <latexrelease>          {\markboth}{Make commands robust}%

```

`\markboth` User commands for setting L^AT_EX marks.

`\markright` Test for `\@nobreak` added 15 Apr 86 in `\markboth` and `\markright` letting `\label` and `\index` to `\relax` added 22 Feb 86 so these commands can appear in sectioning command arguments RmS 91/06/21 Same for `\glossary`

```

22 \DeclareRobustCommand*\markboth[2]{%
23   \begingroup
24     \let\label\relax \let\index\relax \let\glossary\relax
25     \unrestored@protected@xdef\@themark {{#1}{#2}}%
26     \@temptokena \expandafter{\@themark}%
27     \mark{\the\@temptokena}%
28   \endgroup
29   \if@nobreak\ifvmode\nobreak\fi\fi}

```

```

30 \DeclareRobustCommand*\markright[1]{%
31   \begingroup
32     \let\label\relax \let\index\relax \let\glossary\relax

```

Protection is handled inside `\@markright`.

```

33   \expandafter\@markright\@themark {#1}%
34   \@temptokena \expandafter{\@themark}%
35   \mark{\the\@temptokena}%
36 \endgroup
37 \if@nobreak\ifvmode\nobreak\fi\fi}

```

```

(End definition for \markboth and \markright.)

38 </2ekernel | latexrelease>
39 <latexrelease>\EndIncludeInRelease
40 <latexrelease>\IncludeInRelease{0000/00/00}%
41 <latexrelease>                {\markboth}{Make commands robust}%
42 <latexrelease>
43 <latexrelease>\kernel@make@fragile\markboth
44 <latexrelease>\kernel@make@fragile\markright
45 <latexrelease>
46 <latexrelease>\EndIncludeInRelease
47 <*2ekernel>

\@markright
\leftmark 48 \def\@markright#1#2#3{\@temptokena {#1}%
\rightmark 49 \unrestored@protected@xdef\@themark{\the\@temptokena}{#3}}
50 \def\leftmark{\expandafter\@leftmark\botmark\@empty\@empty}
51 \def\rightmark{\expandafter\@rightmark\firstmark\@empty\@empty}

(End definition for \@markright, \leftmark, and \rightmark.)

\@themark Initialise LATEX's marks without setting a TEX mark <whatsit>.
52 \def\@themark{\{}{\}}

(End definition for \@themark.)

\mark Test versions of LATEX 2ε initialised TEX's \mark system at this point, but this was
removed before the first release.

AtBeginDocument{\mark{\{}{\}}\}}

(End definition for \mark.)

\raggedbottom \raggedbottom typesets pages with no vertical stretch, so they have their natural height
instead of all being exactly the same height. (Uses a space of .0001fil to avoid interfering
with the 1fil space of \newpage.)

53 \DeclareRobustCommand\raggedbottom{%
54 \def\@textbottom{\vskip \z@ \@plus.0001fil}\let\@texttop\relax}

(End definition for \raggedbottom.)

\flushbottom \flushbottom: Inverse of \raggedbottom — makes all pages the same height.

55 \DeclareRobustCommand\flushbottom{%
56 \let\@textbottom\relax \let\@texttop\relax}

(End definition for \flushbottom.)

\sloppy \sloppy will never (well, hardly ever) produce overfull boxes, but may produce underfull
ones. (14 June 85)

57 \DeclareRobustCommand\sloppy{%
58 \tolerance 9999%
59 \emergencystretch 3em%
60 \hfuzz .5\p@
61 \vfuzz\hfuzz}

(End definition for \sloppy.)

```

sloppypar A sloppypar environment is equivalent to `{\par \sloppy ... \par}`.

```

62 \def\sloppypar{\par\sloppy}
63 \def\endsloppypar{\par}

```

\fussy Resets T_EX's parameters to their normal finicky values.

```

64 \DeclareRobustCommand\fussy{%
65   \emergencystretch\z@
66   \tolerance 200%
67   \hfuzz .1\p@
68   \vfuzz\hfuzz}

```

(End definition for \fussy.)

\overfullrule L^AT_EX default is no overfull box rule. Changed by document class option.

```

69 \overfullrule 0pt

```

(End definition for \overfullrule.)

```

70 </2ekernel>

```


File S

ltclass.dtx

1 Introduction

This file implements the following declarations, which replace `\documentstyle` in $\text{\LaTeX} 2_{\epsilon}$ documents.

Note that old documents containing `\documentstyle` will be run using a compatibility option—thus keeping everyone happy, we hope!

The overall idea is that there are two types of ‘style files’: ‘class files’ which define elements and provide a default formatting for them; and ‘packages’ which provide extra functionality. One difference between $\text{\LaTeX} 2_{\epsilon}$ and $\text{\LaTeX} 2.09$ is that $\text{\LaTeX} 2_{\epsilon}$ packages may have options. Note that options to classes/packages may be implemented such that they input files, but these file names are not necessarily directly related to the option name.

2 User interface

`\documentclass[<main-option-list>]{<class>}[<version>]`

There must be exactly one such declaration, and it must come first. The *<main-option-list>* is a list of options which can modify the formatting of elements which are defined in the *<class>* file as well as in all following `\usepackage` declarations (see below). The *<version>* is a version number, beginning with a date in the format YYYY/MM/DD. If an older version of the class is found, a warning is issued.

`\documentstyle[<main-option-list>]{<class>}[<version>]`

The `\documentstyle` declaration is kept in order to maintain upward compatibility with $\text{\LaTeX} 2.09$ documents. It is similar to `\documentclass`, but it causes all options in *<main-option-list>* that the *<class>* does not use to be passed to `\RequirePackage` after the options have been processed. This maintains compatibility with the 2.09 behaviour. Also a flag is set to indicate that the document is to be processed in $\text{\LaTeX} 2.09$ compatibility mode. As far as most packages are concerned, this only affects the warnings and errors \LaTeX generates. This flag does affect the definition of font commands, and `\sloppy`.

`\usepackage[<package-option-list>]{<package-list>}[<version>]`

There can be any number of these declarations. All packages in *<package-list>* are called with the same options.

Each *<package>* file defines new elements (or modifies those defined in the *<class>*), and thus extends the range of documents which can be processed. The *<package-option-list>* is a list of options which can modify the formatting of elements defined in the *<package>* file. The *<version>* is a version number, beginning with a date in the format YYYY/MM/DD. If an older version of the package is found, a warning is issued.

Each package is loaded only once. If the same package is requested more than once, nothing happens, unless the package has been requested with options that were not given the first time it was loaded, in which case an error is produced.

As well as processing the options given in the *<package-option-list>*, each package processes the *<main-option-list>*. This means that options that affect all of the packages can be given globally, rather than repeated for every package.

Note that class files have the extension `.cls`, packages have the extension `.sty`.

`filecontents` The environment `filecontents` is intended for passing the contents of packages, options, or other files along with a document in a single file. It has one argument, which is the name of the file to create. If that file already exists (maybe only in the current directory if the OS supports a notion of a ‘current directory’ or ‘default directory’) then nothing happens (except for an information message) and the body of the environment is bypassed. Otherwise, the body of the environment is written verbatim to the file name given as the first argument, together with some comments about how it was produced.

The environment can also be called with an optional argument which is used to alter some of its behavior: option `force` or `overwrite` will allow for overwriting existing files, option `nosearch` will only check the current directory when looking if the file exists. This can be useful if you want to generate a local (modified) copy of some file that is already in the search tree of \TeX . Finally, you can use `noheader` to prevent it from writing the standard blurb at the top of the file (this is actually the same as using the star form of the environment).

The environment is now allowed anywhere in the document, but to ensure that all packages or options necessary are available when the document is run, it is normally best to place it at the top of your file (before `\documentclass`). A possible use case for using it inside the document body is if you want to reuse some text several times in the document you could then write it and later use `\input` to retrieve it where needed.

The begin and end tags should each be on a line by itself.

2.1 Option processing

When the options are processed, they are divided into two types: *local* and *global*:

- For a class, the options in the `\documentclass` command are local.
- For a package, the options in the `\usepackage` command are local, and the options in the `\documentclass` command are global.

The options for `\documentclass` and `\usepackage` are processed in the following way:

1. The local and global options that have been declared (using `\DeclareOption` as described below) are processed first.

In the case of `\ProcessOptions`, they are processed in the order that they were declared in the class or package.

In the case of `\ProcessOptions*`, they are processed in the order that they appear in the option-lists. First the global options, and then the local ones.

2. Any remaining local options are dealt with using the default option (declared using the `\DeclareOption*` declaration described below). For document classes, this usually does nothing, but records the option on a list of unused options. For packages, this usually produces an error.

Finally, when `\begin{document}` is reached, if there are any global options which have not been used by either the class or any package, the system will produce a warning.

3 Class and Package interface

3.1 Class name and version

`\ProvidesClass` A class can identify itself with the `\ProvidesClass{<name>}[<version>]` command. The `<version>` should begin with a date in the format YYYY/MM/DD.

3.2 Package name and version

`\ProvidesPackage` A package can identify itself with the `\ProvidesPackage{<name>}[<version>]` command. The `<version>` should begin with a date in the format YYYY/MM/DD.

3.3 Requiring other packages

`\RequirePackage` Packages or classes can load other packages using `\RequirePackage[<options>]{<name>}[<version>]`. If the package has already been loaded, then nothing happens unless the requested options are not a subset of the options with which it was loaded, in which case an error is called.

`\LoadClass` Similar to `\RequirePackage`, but for classes, may not be used in package files.

`\PassOptionsToPackage` Packages can pass options to other packages using:

`\PassOptionsToPackage{<options>}{<package>}`.

`\PassOptionsToClass` This adds the `<options>` to the options list of any future `\RequirePackage` or `\usepackage` command. For example:

```
\PassOptionsToPackage{foo,bar}{fred}
```

is the same as:

```
\RequirePackage[foo,bar,baz]{fred}
```

`\LoadClassWithOptions` `\LoadClassWithOptions{<name>}[<version>]:`

This is similar to `\LoadClass`, but it always calls class `<name>` with exactly the same option list that is being used by the current class, rather than an option explicitly supplied or passed on by `\PassOptionsToClass`. `\RequirePackageWithOptions` is the analogous command for packages.

This is mainly intended to allow one class to simply build on another, for example:

```
\LoadClassWithOptions{article}
```

This should be contrasted with the slightly different construction

```
\DeclareOption*{\PassOptionsToClass{\CurrentOption}{article}}
\ProcessOptions
\LoadClass{article}
```

As used here, the effects are more or less the same, but the version using `\LoadClassWithOptions` is slightly quicker (and less to type). If, however, the class declares options of its own then the two constructions are different; compare, for example:

```
\DeclareOption{landscape}{...}
\ProcessOptions
\LoadClassWithOptions{article}
```

with:

```
\DeclareOption{landscape}{...}
\DeclareOption*{\PassOptionsToClass{\CurrentOption}{article}}
\ProcessOptions
\LoadClass{article}
```

In the first case, the `article` class will be called with option `landscape` precisely when the current class is called with this option; but in the second example it will not as in that case `article` is only passed options by the default option handler, which is not used for `landscape` as that option is explicitly declared.

<pre>\@ifpackageloaded \@ifclassloaded \@ifpackagelater \@ifclasslater \@ifpackagewith \@ifclasswith</pre>	<p>To find out if a package has already been loaded, use</p> <pre>\@ifpackageloaded{<package>}{<true>}{<false>}.</pre> <p>To find out if a package has already been loaded with a version equal to or more recent than <code><version></code>, use</p> <pre>\@ifpackagelater{<package>}{<version>}{<true>}{<false>}.</pre> <p>To find out if a package has already been loaded with at least the options <code><options></code>, use</p> <pre>\@ifpackagewith{<package>}{<options>}{<true>}{<false>}.</pre>
--	---

There exists one package that can't be tested with the above commands: the `fontenc` package pretends that it was never loaded to allow for repeated reloading with different options (see `ltoutenc.dtx` for details).

3.4 Declaring new options

Options for classes and packages are built using the same macros.

<pre>\DeclareOption \DeclareOption*</pre>	<p>To define a builtin option, use <code>\DeclareOption{<name>}{<code>}</code>.</p> <p>To define the default action to perform for local options which have not been declared, use <code>\DeclareOption*{<code>}</code>.</p>
---	--

Note: there should be no use of

`\RequirePackage`, `\DeclareOption`, `\DeclareOption*` or `\ProcessOptions` inside `\DeclareOption` or `\DeclareOption*`.

Possible uses for `\DeclareOption*` include:

```
\DeclareOption*{}

```

Do nothing. Silently accept unknown options. (This suppresses the usual warnings.)

```
\DeclareOption*{@unkownoptionerror}

```

Complain about unknown local options. (The initial setting for package files.)

```
\DeclareOption*{\PassOptionsToPackage{\CurrentOption}{<pkg-name>}}

```

Handle the current option by passing it on to the package `<pkg-name>`, which will presumably be loaded via `\RequirePackage` later in the file. This is useful for building 'extension' packages, that perhaps handle a couple of new options, but then pass everything else on to an existing package.

```
\DeclareOption*{\InputIfFileExists{xx-\CurrentOption.yyy}%
{}%
{\OptionNotUsed}}
```

Handle the option `foo` by loading the file `xx-foo.yyy` if it exists, otherwise do nothing, but declare that the option was not used. Actually the `\OptionNotUsed` declaration is only needed if this is being used in class files, but does no harm in package files.

3.5 Safe Input Macros

<code>\InputIfFileExists</code>	<code>\InputIfFileExists{<file>}{<then>}{<else>}</code> Inputs <i><file></i> if it exists. Immediately before the input, <i><then></i> is executed. Otherwise <i><else></i> is executed.
<code>\IfFileExists</code>	As above, but does not input the file. One thing you might like to put in the <i><else></i> clause is
<code>\@missingfileerror</code>	This starts an interactive request for a filename, supplying default extensions. Just hitting return causes the whole input to be skipped and entering x quits the current run,
<code>\input</code>	This has been redefined from the L ^A T _E X 2.09 definition, in terms of the new commands <code>\InputIfFileExists</code> and <code>\@missingfileerror</code> .
<code>\listfiles</code>	Giving this declaration in the preamble causes a list of all files input via the ‘safe input’ commands to be listed at the end. Any strings specified in the optional argument to <code>\ProvidesPackage</code> are listed alongside the file name. So files in standard (and other non-standard) distributions can put informative strings in this argument.

4 Implementation

	<code>1 <*2kernel></code>
<code>\if@compatibility</code>	The flag for compatibility mode. <code>2 \newif\if@compatibility</code> (End definition for <code>\if@compatibility</code> .)
<code>\@documentclasshook</code>	This legacy hook is called after the first <code>\documentclass</code> command. It is <i>not</i> integrated with the new 2020 hook management system! By default this checks to see if <code>\@normalsize</code> is undefined, and if so, sets it to <code>\normalsize</code> . <code>3 \def\@documentclasshook{%</code> <code>4 \ifx\@normalsize\undefined</code> <code>5 \let\@normalsize\normalsize</code> <code>6 \fi</code> <code>7 }</code> (End definition for <code>\@documentclasshook</code> .)
<code>\@declaredoptions</code>	This list is automatically built by <code>\DeclareOption</code> . It is the list of options (separated by commas) declared in the class or package file and it defines the order in which the corresponding <code>\ds@<option></code> commands are executed. All local <i><option></i> s which are not declared will be processed in the order defined by the optional argument of <code>\documentclass</code> or <code>\usepackage</code> . <code>8 \let\@declaredoptions\@empty</code> (End definition for <code>\@declaredoptions</code> .)
<code>\@classoptionslist</code>	List of options of the main class. <code>9 \let\@classoptionslist\relax</code> <code>10 \@onlypreamble\@classoptionslist</code> (End definition for <code>\@classoptionslist</code> .)
<code>\@raw@classoptionslist</code>	List of options of the main class (unprocessed). <code>11 \let\@raw@classoptionslist\relax</code>

```

(End definition for \@raw@classoptionslist.)

\@unusedoptionlist List of options of the main class that haven't been declared or loaded as class option files.
12 \let\@unusedoptionlist\@empty
13 \@onlypreamble\@unusedoptionlist
(End definition for \@unusedoptionlist.)

\CurrentOption Name of current package or option.
14 \let\CurrentOption\@empty
(End definition for \CurrentOption.)

\@currpath Path to the current file if explicitly given.
15 </2ekernel>
16 <*2ekernel | latexrelease>
17 <latexrelease>
18 <latexrelease>\IncludeInRelease{2020/10/01}{\@currpath}%
19 <latexrelease> {Add \@currpath}%
20 \let\@currpath\@empty
21 <latexrelease>\EndIncludeInRelease
22 %
23 <latexrelease>\IncludeInRelease{0000/00/00}{\@currpath}%
24 <latexrelease> {Add \@currpath}%
25 <latexrelease>\let\@currpath\@undefined
26 <latexrelease>\EndIncludeInRelease
27 </2ekernel | latexrelease>
28 <*2ekernel>
(End definition for \@currpath.)

\@currname Name of current package or option.
29 \let\@currname\@empty
(End definition for \@currname.)

\@current The current file extension.
30 \global\let\@current=\@empty
(End definition for \@current.)

\@clsextension The two possible values of \@current.
\@pkgextension
31 \def\@clsextension{cls}
32 \def\@pkgextension{sty}
33 \@onlypreamble\@clsextension
34 \@onlypreamble\@pkgextension
(End definition for \@clsextension and \@pkgextension.)

```

```

\@pushfilename Commands to push and pop the file name and extension.
\@popfilename #1 current name.
\@currnamestack #2 current extension.
                #3 current catcode of @.
                #4 Rest of the stack.
35 </2ekernel>
36 <{*2ekernel | latexrelease>
37 <latexrelease>
38 <latexrelease>\IncludeInRelease{2020/10/01}{\@pushfilename}%
39 <latexrelease> {Add \@expl@push@filename@@ and \@expl@push@filename@aux@@}%
40 \def\@pushfilename{%

```

The push and pop macros are injected in \@pushfilename and \@popfilename so that they correctly keep track of the hook labels.

This needs cleanup with the expl3 interfaces also playing here, e.g., \@expl@push@filename@@ needs cleanup and (and should probably not have this name either).

```

41   \@expl@push@filename@@
42   \xdef\@currnamestack{%
43     {\@currname}%
44     {\@currentx}%
45     {\the\catcode'\@}%
46     \@currnamestack}%

```

Temporarily add a stack for \@currpath here. This should be integrated in the main file stack eventually, but other packages rely on \@currnamestack having three elements per file, so that isn't a trivial change. The prefix \@kernel@... hopefully discourages people from using it.

```

47   \xdef\@kernel@currpathstack{%
48     {\@currpath}%
49     \@kernel@currpathstack}%
50   \@expl@push@filename@aux@@}
51 <latexrelease>\EndIncludeInRelease

```

The following version of \@pushfilename didn't formally exist in this file, but in the 2020/02/02 release, expl3 was preloaded and it patched \@pushfilename (and \@popfilename) by adding some hooks in there. But rolling back to 2020/02/02, expl3 doesn't patch these macros again, so rolling back has to take those hooks into account. Same goes for \@popfilename.

```

52 <latexrelease>
53 <latexrelease>\IncludeInRelease{2020/02/02}{\@pushfilename}%
54 <latexrelease> {Add \@expl@push@filename@@}%
55 <latexrelease>\def\@pushfilename{%
56 <latexrelease>   \@expl@push@filename@@
57 <latexrelease>   \xdef\@currnamestack{%
58 <latexrelease>     {\@currname}%
59 <latexrelease>     {\@currentx}%
60 <latexrelease>     {\the\catcode'\@}%
61 <latexrelease>     \@currnamestack}%
62 <latexrelease>     \@expl@push@filename@aux@@}
63 <latexrelease>\EndIncludeInRelease
64 <latexrelease>

```

When we roll back from a release that has expl3 preloaded, the definitions of \@pushfilename and \@popfilename can't be completely rolled back otherwise expl3-based packages won't have the automatic \ExplSyntaxOff at the end. Here and

below for `\@popfilename`, we don't roll back all the way through if coming from \LaTeX > 2020-02-02.

```

65 <latexrelease>\IncludeInRelease{0000/00/00}{\@pushfilename}%
66 <latexrelease> {Add \@expl@push@filename@@ and \@expl@push@filename@aux@@}%
67 <latexrelease>\ifnum\sourceLaTeXdate<20200202\relax
68 <latexrelease> \GenericInfo{}\{Defining 00-00-00\string\@pushfilename.\}
69 <latexrelease>\def\@pushfilename{%
70 <latexrelease> \xdef\@currnamestack{%
71 <latexrelease> {\@currname}%
72 <latexrelease> {\@currentx}%
73 <latexrelease> {\the\catcode'\@}%
74 <latexrelease> \@currnamestack}}
75 <latexrelease>\else
76 <latexrelease> \GenericInfo{}\{Defining 2020-02-02\string\@pushfilename.\}
77 <latexrelease>\def\@pushfilename{%
78 <latexrelease> \@expl@push@filename@@
79 <latexrelease> \xdef\@currnamestack{%
80 <latexrelease> {\@currname}%
81 <latexrelease> {\@currentx}%
82 <latexrelease> {\the\catcode'\@}%
83 <latexrelease> \@currnamestack}%
84 <latexrelease> \@expl@push@filename@aux@@}
85 <latexrelease>\fi
86 <latexrelease>\EndIncludeInRelease
87 \onlypreamble\@pushfilename

88 <latexrelease>
89 <latexrelease>\IncludeInRelease{2020/10/01}{\@popfilename}%
90 <latexrelease> {Add \@expl@pop@filename@@}%
91 \def\@popfilename{\@expl@@@hook@curr@name@pop@@
92 \expandafter\@p@pfilename\@currnamestack\@nil

```

Same for popping:

```

93 \expandafter\@p@pfilepath\@kernel@currpathstack\@nil
94 \@expl@pop@filename@@}
95 <latexrelease>\EndIncludeInRelease
96 <latexrelease>
97 <latexrelease>\IncludeInRelease{2020/02/02}{\@popfilename}%
98 <latexrelease> {Add \@expl@push@filename@@}%
99 <latexrelease>\def\@popfilename{\expandafter\@p@pfilename\@currnamestack\@nil
100 <latexrelease> \@expl@pop@filename@@}
101 <latexrelease>\EndIncludeInRelease
102 <latexrelease>

103 <latexrelease>\IncludeInRelease{0000/00/00}{\@popfilename}%
104 <latexrelease> {Add \@expl@push@filename@@ and \@expl@push@filename@aux@@}%
105 <latexrelease>\ifnum\sourceLaTeXdate<20200202\relax
106 <latexrelease> \GenericInfo{}\{Defining 00-00-00\string\@popfilename.\}
107 <latexrelease>\def\@popfilename{\expandafter\@p@pfilename\@currnamestack\@nil}
108 <latexrelease>\else
109 <latexrelease> \GenericInfo{}\{Defining 2020-02-02\string\@popfilename.\}
110 <latexrelease>\def\@popfilename{\expandafter\@p@pfilename\@currnamestack\@nil
111 <latexrelease> \@expl@pop@filename@@}
112 <latexrelease>\fi
113 <latexrelease>\EndIncludeInRelease

```



```

114 \@onlypreamble\@popfilename
115 </2ekernel | latexrelease>
116 <*2ekernel>

117 \def\@p@pfilename#1#2#3#4\@nil{%
118   \gdef\@currname{#1}%
119   \gdef\@currentt{#2}%
120   \catcode'\@#3\relax
121   \gdef\@currnamestack{#4}}
122 \@onlypreamble\@p@pfilename

123 \gdef\@currnamestack{}
124 \@onlypreamble\@currnamestack

(End definition for \@pushfilename, \@popfilename, and \@currnamestack.)

```

\@kernel@currpathstack Path to the current file if explicitly given. The auxiliary is needed here to insert a `\@empty` to prevent the loss of braces.

```

125 </2ekernel>
126 <*2ekernel | latexrelease>
127 <latexrelease>
128 <latexrelease>\IncludeInRelease{2020/10/01}{\@kernel@currpathstack}%
129 <latexrelease> {Add \@kernel@currpathstack}%
130 \gdef\@kernel@currpathstack{}
131 <latexrelease>\g@addto@macro\@kernel@currpathstack{}}
132 \def\@p@pfilepath#1{%
133   \gdef\@currpath{#1}\@p@pfilepath@aux\@empty}
134 \def\@p@pfilepath@aux#1\@nil{%
135   \xdef\@kernel@currpathstack{#1}}
136 <latexrelease>\EndIncludeInRelease
137 %
138 <latexrelease>\IncludeInRelease{0000/00/00}{\@kernel@currpathstack}%
139 <latexrelease> {Add \@kernel@currpathstack}%
140 <latexrelease>\let\@kernel@currpathstack\@undefined
141 <latexrelease>\let\@p@pfilepath\@undefined
142 <latexrelease>\let\@p@pfilepath@aux\@undefined
143 <latexrelease>\EndIncludeInRelease
144 </2ekernel | latexrelease>
145 <*2ekernel>

(End definition for \@kernel@currpathstack.)

```

\@optionlist Returns the option list of the file.

```

146 \def\@optionlist#1{%
147   \@ifundefined{opt@#1}\@empty{\csname opt@#1\endcsname}}
148 \@onlypreamble\@optionlist

(End definition for \@optionlist.)

```

\@ifpackageloaded **\@ifclassloaded** `\@ifpackageloaded{<name>}` Checks to see whether a file has been loaded.

```

149 \def\@ifpackageloaded{\@ifl@aded\@pkgextension}
150 \def\@ifclassloaded{\@ifl@aded\@clsextension}
151 \@onlypreamble\@ifpackageloaded
152 \@onlypreamble\@ifclassloaded

```

```

153 \def\@ifl@aded#1#2{%
154   \expandafter\ifx\csname ver@#2.#1\endcsname\relax
155   \expandafter\@secondoftwo
156   \else
157   \expandafter\@firstoftwo
158   \fi}
159 \@onlypreamble\@ifl@aded

```

(End definition for \@ifpackagelater and \@ifclasslater.)

`\@ifpackagelater` `\@ifclasslater` `\@ifpackagelater{<name>}{YYYY/MM/DD}{<true code>}{<false code>}` Checks that the package loaded is more recent or equal to the given date. A better name for it would therefore be `\@ifpackagelaterorequal` but it is in use for more than 30 years, so ...

```

160 \def\@ifpackagelater{\@ifl@ter\@pkgextension}
161 \def\@ifclasslater{\@ifl@ter\@clsextension}
162 \@onlypreamble\@ifpackagelater
163 \@onlypreamble\@ifclasslater

```

(End definition for \@ifpackagelater and \@ifclasslater.)

`\IfPackageAtLeastTF` `\IfFormatAtLeastTF{YYYY/MM/DD}{<true code>}{<false code>}` Test if the format is later or equal to the given date.

```

\IfClassAtLeastTF
\IfFormatAtLeastTF
164 \</2ekernel>
165 \< *2ekernel | latexrelease>
166 \< latexrelease> \IncludeInRelease{2020/10/01}%
167 \< latexrelease> \IfFormatAtLeastTF{Test format date}%
168 \def\IfFormatAtLeastTF{\@ifl@t@r\fmtversion}
169 \let\IfPackageAtLeastTF\@ifpackagelater
170 \let\IfClassAtLeastTF\@ifclasslater
171 \@onlypreamble\IfFormatAtLeastTF
172 \@onlypreamble\IfPackageAtLeastTF
173 \@onlypreamble\IfClassAtLeastTF

```

For rollback pretend it was available since the beginning of dawn.

```

174 \</2ekernel | latexrelease>
175 \< latexrelease> \EndIncludeInRelease
176 \< latexrelease> \IncludeInRelease{0000/00/00}%
177 \< latexrelease> \IfFormatAtLeastTF{Test format date}%
178 \< latexrelease> \def\IfFormatAtLeastTF{\@ifl@t@r\fmtversion}
179 \< latexrelease> \let\IfPackageAtLeastTF\@ifpackagelater
180 \< latexrelease> \let\IfClassAtLeastTF\@ifclasslater
181 \< latexrelease> \EndIncludeInRelease
182 \< *2ekernel>

```

(End definition for \IfPackageAtLeastTF, \IfClassAtLeastTF, and \IfFormatAtLeastTF.)

`\@ifl@ter`

```

183 \def\@ifl@ter#1#2{%
184   \expandafter\@ifl@t@r
185   \csname ver@#2.#1\endcsname}
186 \@onlypreamble\@ifl@ter
187 \</2ekernel>

```

This internal macro is also used in `\NeedsTeXFormat`.

```

188 <latexrelease>\IncludeInRelease{2018/04/01}%
189 <latexrelease>          {\@ifl@t@r}{Guard against bad input}%
190 <*2ekernel | latexrelease>
191 \def\@ifl@t@r#1#2{%
192   \ifnum\expandafter\@parse@version@#1//00\@nil<%
193     \expandafter\@parse@version@#2//00\@nil
194     \expandafter\@secondoftwo
195   \else
196     \expandafter\@firstoftwo
197   \fi}
198 \def\@parse@version@#1{\@parse@version0#1}
199 </2ekernel | latexrelease>
200 <latexrelease>\EndIncludeInRelease
201 <latexrelease>\IncludeInRelease{0000/00/00}%
202 <latexrelease>          {\@ifl@t@r}{Guard against bad input}%
203 <latexrelease>\def\@ifl@t@r#1#2{%
204 <latexrelease>   \ifnum\expandafter\@parse@version#1//00\@nil<%
205 <latexrelease>     \expandafter\@parse@version#2//00\@nil
206 <latexrelease>     \expandafter\@secondoftwo
207 <latexrelease>   \else
208 <latexrelease>     \expandafter\@firstoftwo
209 <latexrelease>   \fi}
210 <latexrelease>\let\@parse@version@\@undefined
211 <latexrelease>\EndIncludeInRelease
212 <*2ekernel>
213 \@onlypreamble\@ifl@t@r

```

(End definition for `\@ifl@ter`.)

```

214 </2ekernel>
215 <*2ekernel | latexreleasefirst>
216 \def\@parse@version#1/#2/#3#4#5\@nil{%
217 \@parse@version@dash#1-#2-#3#4\@nil
218 }

```

The `\if` test here ensures that an argument with no `/` or `-` produces 0 (actually 00).

```

219 \def\@parse@version@dash#1-#2-#3#4#5\@nil{%
220   \if\relax#2\relax\else#1\fi#2#3#4 }
221 </2ekernel | latexreleasefirst>
222 <*2ekernel>

```

`\@ifpackagewith` `\@ifclasswith` `\@ifpackagewith{<name>}{<option-list>}` Checks that `<option-list>` is a subset of the options **with** which `<name>` was loaded.

```

223 \def\@ifpackagewith{\@ifoptions\@pkgextension}
224 \def\@ifclasswith{\@ifoptions\@clsextension}
225 \@onlypreamble\@ifpackagewith
226 \@onlypreamble\@ifclasswith
227 \def\@ifoptions#1#2{%
228   \expandtwoargs\@if@pti@ns{\@optionlist{#2.#1}}
229 \@onlypreamble\@ifoptions

```

Probably shouldn't use \CurrentOption here... (changed to \reserved@b.)

```

230 </2ekernel>
231 <latexrelease>\IncludeInRelease{2017/01/01}%
232 <latexrelease>      {\@ifopti@ns}{Spaces in option clash check}%
233 <*2ekernel | latexrelease>
234 \def\@ifopti@ns#1#2{%
235   \let\reserved@a\@firstoftwo

236   \edef\reserved@b{\zap@space#2 \@empty}%
237   \@for\reserved@b:=\reserved@b\do{%
238     \ifx\reserved@b\@empty
239     \else
240       \expandafter\in@\expandafter{\expandafter,\reserved@b,}{, #1,}%
241       \ifin@
242       \else
243         \let\reserved@a\@secondoftwo
244       \fi
245     \fi
246   }%
247   \reserved@a}
248 </2ekernel | latexrelease>
249 <latexrelease>\EndIncludeInRelease
250 <latexrelease>\IncludeInRelease{0000/00/00}%
251 <latexrelease>      {\@ifopti@ns}{Spaces in option clash check}%
252 <latexrelease>\def\@ifopti@ns#1#2{%
253 <latexrelease> \let\reserved@a\@firstoftwo
254 <latexrelease> \@for\reserved@b:=#2\do{%
255 <latexrelease> \ifx\reserved@b\@empty
256 <latexrelease>   \else
257 <latexrelease>   \expandafter\in@\expandafter
258 <latexrelease>     {\expandafter,\reserved@b,}{, #1,}%
259 <latexrelease>   \ifin@
260 <latexrelease>   \else
261 <latexrelease>     \let\reserved@a\@secondoftwo
262 <latexrelease>   \fi
263 <latexrelease> \fi
264 <latexrelease> }%
265 <latexrelease> \reserved@a}
266 <latexrelease>\EndIncludeInRelease
267 <*2ekernel>

268 \@onlypreamble\@ifopti@ns

```

(End definition for \@ifpackagewith and \@ifclasswith.)

\ProvidesPackage Checks that the current filename is correct, and defines \ver@filename.

```

269 </2ekernel>
270 <latexrelease>\IncludeInRelease{2020/10/01}%
271 <latexrelease> {\ProvidesPackage}{Check name with \strcmp}%
272 <*2ekernel | latexrelease>
273 \def\ProvidesPackage#1{%
274   \xdef\@tempa{#1}%

```

Here \@currpath is explicitly added to the file name to report when a package or class is loaded using an explicit path. Loading using a path in the argument is supported but not encouraged.

```

275 \expandtwoargs\@expl@str@if@eq@nnTF
276 {\@gtempa}{\@currpath\@currname}{\@%
277 \latex@warning@no@line{You have requested
278 \cls@pkg\space'\@currpath\@currname',\MessageBreak
279 but the \cls@pkg\space provides '#1'}%
280 }%
281 \ifnextchar[\@pr@videpackage{\@pr@videpackage[]}]%
282 \onlypreamble\ProvidesPackage
283 \</2ekernel | latexrelease>
284 \<latexrelease>\EndIncludeInRelease
285 %
286 \<latexrelease>\IncludeInRelease{0000/00/00}%
287 \<latexrelease> {\ProvidesPackage}{Undo: check name with \strcmp}%
288 \<latexrelease>\def\ProvidesPackage#1{%
289 \<latexrelease> \xdef\@gtempa{#1}%
290 \<latexrelease> \ifx\@gtempa\@currname\else
291 \<latexrelease> \latex@warning@no@line{You have requested
292 \<latexrelease> \cls@pkg\space'\@currname',\MessageBreak
293 \<latexrelease> but the \cls@pkg\space provides '#1'}%
294 \<latexrelease> \fi
295 \<latexrelease> \ifnextchar[\@pr@videpackage{\@pr@videpackage[]}]%
296 \<latexrelease>\EndIncludeInRelease
297 \<2ekernel>

```

(End definition for \ProvidesPackage.)

\@pr@videpackage This is the helper command for \ProvidesPackage. It tries to be cautious when handling the identification string in case it contains UTF-8 characters.

```

298 \</2ekernel>
299 \<2ekernel | latexrelease>
300 \<latexrelease>\IncludeInRelease{2020/10/01}%
301 \<latexrelease> {\@pr@videpackage}{Allow for package substitution}%
302 \def\@pr@videpackage#1{%
303 \expandafter\protected@xdef % <-- protected...
304 \csname ver@\@currname.\@current\endcsname{#1}% Loaded package
305 \expandafter\let
306 \csname ver@\@currpkg@reqd\expandafter\endcsname % Requested package
307 \csname ver@\@currname.\@current\endcsname
308 \ifx\@current\@clsextension
309 \typeout{Document Class: \@gtempa\space#1}%
310 \else
311 \protected@wlog{Package: \@gtempa\space#1}% <--- protected
312 \fi}

```

(End definition for \@pr@videpackage.)

\protected@wlog This is like plain T_EX's \wlog but gracefully handles protected commands.

```

313 \long\def\protected@wlog#1{\begingroup
314 \set@display@protect
315 \immediate \write \m@ne {#1}\endgroup }

```

(End definition for \protected@wlog.)

```

316 \</2ekernel | latexrelease>
317 \<latexrelease>\EndIncludeInRelease

```

```

318 <latexrelease>\IncludeInRelease{2020/02/02}%
319 <latexrelease>          {\@pr@videpackage}{Protection for package info}%
320 <latexrelease>
321 <latexrelease>\def\@pr@videpackage[#1]{%
322 <latexrelease>  \expandafter\protected@xdef          %      <-- protected...
323 <latexrelease>    \csname ver@\@currname.\@current\endcsname{#1}%
324 <latexrelease>\ifx\@current\@clsextension
325 <latexrelease>  \typeout{Document Class: \@gtempa\space#1}%
326 <latexrelease>  \else
327 <latexrelease>    \protected@wlog{Package: \@gtempa\space#1}%  <--- protected
328 <latexrelease>  \fi}
329 <latexrelease>
330 <latexrelease>\EndIncludeInRelease
331 <latexrelease>\IncludeInRelease{0000/00/00}%
332 <latexrelease>          {\@pr@videpackage}{Protection for package info}%
333 <latexrelease>
334 <latexrelease>\def\@pr@videpackage[#1]{%
335 <latexrelease>  \expandafter\xdef\csname ver@\@currname.\@current\endcsname{#1}%
336 <latexrelease>  \ifx\@current\@clsextension
337 <latexrelease>    \typeout{Document Class: \@gtempa\space#1}%
338 <latexrelease>  \else
339 <latexrelease>    \wlog{Package: \@gtempa\space#1}%
340 <latexrelease>  \fi}
341 <latexrelease>\let\protected@wlog\@undefined
342 <latexrelease>
343 <latexrelease>\EndIncludeInRelease
344 (*2ekernel)
345 \@onlypreamble\@pr@videpackage

```

\ProvidesClass Like **\ProvidesPackage**, but for classes. This needs a dummy latexrelease block to copy the definition of **\ProvidesPackage** as it changes across releases.

```

346 </2ekernel>
347 <latexrelease>\IncludeInRelease{0000/00/00}%
348 <latexrelease>  {\ProvidesClass}{Track \ProvidesPackage}%
349 (*2ekernel | latexrelease)
350 \let\ProvidesClass\ProvidesPackage
351 \@onlypreamble\ProvidesClass
352 </2ekernel | latexrelease>
353 <latexrelease>\EndIncludeInRelease
354 (*2ekernel)

```

(End definition for \ProvidesClass.)

\ProvidesFile Like **\ProvidesPackage**, but for arbitrary files. Do not apply **\@onlypreamble** to these, as we may want to label files input during the document.

```

\@providesfile 355 \def\ProvidesFile#1{%
356   \begingroup
357   \catcode'\ 10 %
358   \ifnum \endlinechar<256 %
359     \ifnum \endlinechar>\m@ne
360       \catcode\endlinechar 10 %
361   \fi

```

```

362 \fi
363 \@makeother\/%
364 \@makeother\&%
365 \kernel@ifnextchar[{\@providesfile{#1}}{\@providesfile{#1}[]}]

```

During intex a special version of \@providesfile is used. The real definition is installed right at the end, in ltfinal.dtx.

```

def\@providesfile#1[#2]{%
  \wlog{File: #1 #2}%
  \expandafter\xdef\csname ver@#1\endcsname{#2}%
\endgroup}

```

(End definition for \ProvidesFile and \@providesfile.)

\PassOptionsToPackage If the package has been loaded, we check that it was first loaded with the options.
 \PassOptionsToClass Otherwise we add the option list to that of the package.

```

366 </2kernel>
367 <latexrelease>\IncludeInRelease{2021/06/01}%
368 <latexrelease>          {\@passOptions}{Raw option lists}%
369 <*2kernel | latexrelease>
370 \def\@passOptions#1#2#3{%
371   \@expl@@@filehook@set@curr@file@@@nNN
372   {\@expl@@@filehook@resolve@file@subst@@w #3.#1\@nil}%
373   \reserved@a\reserved@b
374   \@expl@@@filehook@clear@replacement@flag@@
375   \expandafter\xdef\csname opt@\reserved@a\endcsname{%
376     \ifundefined{opt@\reserved@a}\@empty
377     {\csname opt@\reserved@a\endcsname,}%
378     \zap@space#2 \@empty}%
379   \expandafter\let
380     \csname opt@#3.#1\expandafter\endcsname
381     \csname opt@\reserved@a\endcsname

```

Extend raw option list

```

382   \@ifundefined{@raw@opt@#3.#1}%
383     {\expandafter\gdef\csname @raw@opt@#3.#1\endcsname{#2}}%
384     {\expandafter\g@addto@macro\csname @raw@opt@#3.#1\endcsname{, #2}}%
385   }
386 </2kernel | latexrelease>
387 <latexrelease>\EndIncludeInRelease
388 <latexrelease>\IncludeInRelease{2020/10/01}{\@passOptions}
389 <latexrelease> {Add file replacement in \@passOptions}%
390 <latexrelease>
391 <latexrelease>\def\@passOptions#1#2#3{%
392 <latexrelease>   \@expl@@@filehook@set@curr@file@@@nNN
393 <latexrelease>   {\@expl@@@filehook@resolve@file@subst@@w #3.#1\@nil}%
394 <latexrelease>   \reserved@a\reserved@b
395 <latexrelease>   \@expl@@@filehook@clear@replacement@flag@@
396 <latexrelease>   \expandafter\xdef\csname opt@\reserved@a\endcsname{%
397 <latexrelease>     \ifundefined{opt@\reserved@a}\@empty
398 <latexrelease>     {\csname opt@\reserved@a\endcsname,}%
399 <latexrelease>     \zap@space#2 \@empty}%

```

```

400 <latexrelease> \expandafter\let
401 <latexrelease> \csname opt@#3.#1\expandafter\endcsname
402 <latexrelease> \csname opt@\reserved@a\endcsname}
403 <latexrelease>\EndIncludeInRelease

404 <latexrelease>\IncludeInRelease{0000/00/00}{\@pass@options}
405 <latexrelease> {\@pass@options}%
406 <latexrelease>
407 <latexrelease>\def\@pass@options#1#2#3{%
408 <latexrelease> \expandafter\xdef\csname opt@#3.#1\endcsname{%
409 <latexrelease> \ifundefined{opt@#3.#1}\@empty
410 <latexrelease> {\csname opt@#3.#1\endcsname,}%
411 <latexrelease> \zap@space#2 \@empty}}
412 <latexrelease>\EndIncludeInRelease
413 (*2ekernel)

414 \@onlypreamble\@pass@options

415 \def\PassOptionsToPackage{\@pass@options\@pkgextension}
416 \def\PassOptionsToClass{\@pass@options\@clsextension}
417 \@onlypreamble\PassOptionsToPackage
418 \@onlypreamble\PassOptionsToClass

```

(End definition for \PassOptionsToPackage and \PassOptionsToClass.)

\DeclareOption Adds an option as a \ds@ command, or the default \default@ds command.

```

\DeclareOption*
419 \def\DeclareOption{%
420 \let\@fileswithopti@ns\@badrequireerror
421 \ifstar\@defdefault@ds\@declareoption}
422 \long\def\@declareoption#1#2{%
423 \xdef\@declaredoptions{\@declaredoptions,#1}%
424 \toks@{#2}%
425 \expandafter\edef\csname ds@#1\endcsname{\the\toks@}}
426 \long\def\@defdefault@ds#1{%
427 \toks@{#1}%
428 \edef\default@ds{\the\toks@}}
429 \@onlypreamble\DeclareOption
430 \@onlypreamble\@declareoption
431 \@onlypreamble\@defdefault@ds

```

(End definition for \DeclareOption and \DeclareOption*.)

\OptionNotUsed If we are in a class file, add \CurrentOption to the list of unused options. Otherwise, in a package file do nothing.

```

432 </2ekernel>
433 <latexrelease>\IncludeInRelease{2021/06/01}%
434 <latexrelease> {\OptionNotUsed}{filter unused option list}%
435 <*2ekernel | latexrelease>
436 \def\@remove@eq@value#1=#2\@nil{#1}

437 \def\OptionNotUsed{%
438 \ifx\@current\@clsextension
439 \xdef\@unusedoptionlist{%
440 \ifx\@unusedoptionlist\@empty\else\@unusedoptionlist,\fi
441 \expandafter\@remove@eq@value\CurrentOption=\@nil}%
442 \fi}
443 </2ekernel | latexrelease>

```



```

444 <latexrelease>\EndIncludeInRelease
445 <latexrelease>\IncludeInRelease{0000/00/00}%
446 <latexrelease>          {\OptionNotUsed}{filter unused option list}%
447 <latexrelease>\let\@remove@eq@value\undefined

448 <latexrelease>\def\OptionNotUsed{%
449 <latexrelease>  \ifx\@current\@clsextension
450 <latexrelease>    \xdef\@unusedoptionlist{%
451 <latexrelease>      \ifx\@unusedoptionlist\@empty\else\@unusedoptionlist,\fi
452 <latexrelease>      \CurrentOption}%
453 <latexrelease>  \fi}
454 <latexrelease>\EndIncludeInRelease
455 <*2ekernel>

456 \@onlypreamble\OptionNotUsed

(End definition for \OptionNotUsed and \@remove@eq@value.)

```

\default@ds The default option code. Set by **\@onefilewithoptions** to either **\OptionNotUsed** for classes, or **\@unknownoptionerror** for packages. This may be reset in either case with **\DeclareOption***.

```

457 % \let\default@ds\OptionNotUsed

(End definition for \default@ds.)

```

\ProcessOptions **\ProcessOptions** calls **\ds@option** for each known package option, then calls **\default@ds** for each option on the local options list. Finally resets all the declared options to **\relax**. The empty option does nothing, this has to be reset on the off chance it's set to **\relax** if an empty element gets into the **\@declaredoptions** list.

The star form is similar but executes options given in the order specified in the document, not the order they are declared in the file. In the case of packages, global options are executed before local ones.

```

458 \def\ProcessOptions{%
459   \let\ds@\@empty
460   \edef\@curroptions{\@optionlist{\@currname.\@current}}%
461   \@ifstar\@xprocessoptions\@processoptions}
462 \@onlypreamble\ProcessOptions

463 \def\@processoptions{%
464   \@for\CurrentOption:=\@declaredoptions\do{%
465     \ifx\CurrentOption\@empty\else
466       \@expandtwoargs\in@{,\CurrentOption,}%
467       ,\ifx\@current\@clsextension\else\@classoptionslist,\fi
468       \@curroptions,}%
469     \ifin@
470       \@useoption
471       \expandafter\let\csname ds@\CurrentOption\endcsname\@empty
472     \fi
473   \fi}%
474 \@processoptions}
475 \@onlypreamble\@processoptions

476 </2ekernel>
477 <latexrelease>\IncludeInRelease{2021/06/01}%
478 <latexrelease>          {\@xprocessoptions}{safer \@processoptions}%

```

```

479 <*2ekernel | latexrelease>
480 \def\@xprocessoptions{%
481   \ifx\@current\@clsextension\else
482   \ifx\@classoptionslist\relax\else
483   \@for\CurrentOption:=\@classoptionslist\do{%
484     \ifx\CurrentOption\@empty\else
485     \@ifundefined{ds@\CurrentOption}{\fi}%
486     \useoption
487     \expandafter\let\csname ds@\CurrentOption\endcsname\@empty
488   }%
489   \fi}%
490 \fi
491 \fi
492 \@processoptions}
493 </2ekernel | latexrelease>
494 <latexrelease>\EndIncludeInRelease
495 <latexrelease>\IncludeInRelease{0000/00/00}%
496 <latexrelease>          {\@xprocessoptions}{safer \@xprocessoptions}%
497 <latexrelease>\let\@remove@eq@value\@undefined
498 <latexrelease>\def\@xprocessoptions{%
499 <latexrelease>  \ifx\@current\@clsextension\else
500 <latexrelease>    \@for\CurrentOption:=\@classoptionslist\do{%
501 <latexrelease>      \ifx\CurrentOption\@empty\else
502 <latexrelease>        \@expandtwoargs\in@{\, \CurrentOption,}{\, \@declaredoptions,}%
503 <latexrelease>        \ifin@
504 <latexrelease>        \useoption
505 <latexrelease>        \expandafter\let\csname ds@\CurrentOption\endcsname\@empty
506 <latexrelease>        \fi
507 <latexrelease>      \fi}%
508 <latexrelease>  \fi
509 <latexrelease>  \@processoptions}
510 <latexrelease>\EndIncludeInRelease
511 <*2ekernel>
512 \@onlypreamble\@xprocessoptions

```

The common part of \ProcessOptions and \ProcessOptions*.

```

513 </2ekernel>
514 <*2ekernel | latexrelease>
515 <latexrelease>\IncludeInRelease{2020/10/01}%
516 <latexrelease>          {\@processoptions}{Unused options issue}%
517 \def\@processoptions{%
518   \@for\CurrentOption:=\@curroptions\do{%
519     \@ifundefined{ds@\CurrentOption}%
520     {\useoption
521      \default@ds}%

```

There should not be any non-empty definition of \CurrentOption at this point, as all the declared options were executed earlier. This is for compatibility with 2.09 styles which use \def\ds@... directly, and so have options which do not appear in \@declaredoptions.

```

522     \useoption}%

```

Clear all the definitions for option code. First set all the declared options to \relax, then reset the ‘default’ and ‘empty’ options. and the lst of declared options.

```

523   \@for\CurrentOption:=\@declaredoptions\do{%

```

```

524 \expandafter\let\csname ds@\CurrentOption\endcsname\relax}%
525 \let\CurrentOption\@empty
526 \let\@fileswith@pti@ns\@fileswith@pti@ns
527 \AtEndOfPackage{\expandafter\let
528 \csname unprocessedoptions-\@currname.\@current\endcsname
529 \relax}}
530 \@onlypreamble\@process@pti@ns
531 \</2ekernel | latexrelease>
532 \<latexrelease>\EndIncludeInRelease
533 \<latexrelease>\IncludeInRelease{0000/00/00}%
534 \<latexrelease> \<@process@pti@ns>{Unused options issue}%
535 \<latexrelease>
536 \<latexrelease>\def\@process@pti@ns{%
537 \<latexrelease> \<@for\CurrentOption:=\@curroptions\do{%
538 \<latexrelease> \<@ifundefined{ds@\CurrentOption}%
539 \<latexrelease> \<\use@option
540 \<latexrelease> \<default@ds}%
541 \<latexrelease> \<\use@option}%
542 \<latexrelease> \<@for\CurrentOption:=\@declaredoptions\do{%
543 \<latexrelease> \<expandafter\let\csname ds@\CurrentOption\endcsname\relax}%
544 \<latexrelease> \<let\CurrentOption\@empty
545 \<latexrelease> \<let\@fileswith@pti@ns\@fileswith@pti@ns
546 \<latexrelease> \<AtEndOfPackage{\let\@unprocessedoptions\relax}}
547 \<latexrelease>\EndIncludeInRelease
548 \<*2ekernel>

```

(End definition for \ProcessOptions and \ProcessOptions*.)

\@options \@options is a synonym for \ProcessOptions* for upward compatibility with L^AT_EX 2.09 style files.

```

549 \def\@options{\ProcessOptions*}
550 \@onlypreamble\@options

```

(End definition for \@options.)

\use@option Execute the code for the current option.

```

551 \</2ekernel>
552 \<latexrelease>\IncludeInRelease{2021/06/01}%
553 \<latexrelease> \<\use@option>{filter unused option list}%
554 \<*2ekernel | latexrelease>
555 \def\use@option{%
556 \<\expandtwoargs\@removeelement
557 \<\expandafter\@remove@eq@value\CurrentOption=\@nil}%
558 \<\unusedoptionlist\@unusedoptionlist
559 \<\csname ds@\CurrentOption\endcsname>
560 \</2ekernel | latexrelease>
561 \<latexrelease>\EndIncludeInRelease
562 \<latexrelease>\IncludeInRelease{0000/00/00}%
563 \<latexrelease> \<\use@option>{filter unused option list}%
564 \<latexrelease>\def\use@option{%
565 \<latexrelease> \<\expandtwoargs\@removeelement\CurrentOption
566 \<latexrelease> \<\unusedoptionlist\@unusedoptionlist
567 \<latexrelease> \<\csname ds@\CurrentOption\endcsname>
568 \<latexrelease>\EndIncludeInRelease
569 \<*2ekernel>

```

570 \@onlypreamble\@use@option

(End definition for \@use@option.)

\ExecuteOptions \ExecuteOptions{<option-list>} executes the code declared for each option.

```
571 </2ekernel>
572 <latexrelease>\IncludeInRelease{2017/01/01}%
573 <latexrelease> \ExecuteOptions}{Spaces in \ExecuteOptions}%
574 <*2ekernel | latexrelease>
575 \def\ExecuteOptions#1{%
```

Use \@fortmp here as it is anyway cleared during \@for loop so does not change any existing names.

```
576 \edef\@fortmp{\zap@space#1 \@empty}%
577 \def\reserved@a##1\@nil{%
578 \@for\CurrentOption:=\@fortmp\do
579 {\csname ds@\CurrentOption\endcsname}%
580 \edef\CurrentOption{##1}}%
581 \expandafter\reserved@a\CurrentOption\@nil}
582 </2ekernel | latexrelease>
583 <latexrelease>\EndIncludeInRelease
584 <latexrelease>\IncludeInRelease{0000/00/00}%
585 <latexrelease> \ExecuteOptions}{Spaces in \ExecuteOptions}%
586 <latexrelease>\def\ExecuteOptions#1{%
587 <latexrelease> \def\reserved@a##1\@nil{%
588 <latexrelease> \@for\CurrentOption:=#1\do
589 <latexrelease> {\csname ds@\CurrentOption\endcsname}%
590 <latexrelease> \edef\CurrentOption{##1}}%
591 <latexrelease> \expandafter\reserved@a\CurrentOption\@nil}
592 <latexrelease>\EndIncludeInRelease
593 <*2ekernel>
```

594 \@onlypreamble\ExecuteOptions

(End definition for \ExecuteOptions.)

The top-level commands, which just set some parameters then call the internal command, \@fileswithoptions.

\documentclass The main new-style class declaration.

```
595 \def\documentclass{%
596 \let\documentclass\@twoclasseserror
597 \if@compatibility\else\let\usepackage\RequirePackage\fi
598 \@fileswithoptions\@clsextension}
599 \@onlypreamble\documentclass
```

(End definition for \documentclass.)

\documentstyle 2.09 style class ‘style’ declaration.

```
600 \def\documentstyle{%
601 \makeatletter\input{latex209.def}\makeatother
602 \documentclass}
603 \@onlypreamble\documentstyle
```

(End definition for \documentstyle.)

`\RequirePackage` Load package if not already loaded.

```
604 \def\RequirePackage{%
605   \@fileswithoptions\@pkgextension}
606 \@onlypreamble\RequirePackage
```

(End definition for \RequirePackage.)

`\LoadClass` Load class.

```
607 \def\LoadClass{%
608   \ifx\@current\@pkgextension
609     \@latex@error
610       {\noexpand\LoadClass in package file}%
611       {You may only use \noexpand\LoadClass in a class file.}%
612   \fi
613   \@fileswithoptions\@clsextension}
614 \@onlypreamble\LoadClass
```

(End definition for \LoadClass.)

`\@loadwithoptions` Pass the current option list on to a class or package. #1 is `\@cls-or-pkgextension`, #2 is `\RequirePackage` or `\LoadClass`, #3 is the class or package to be loaded.

```
615 \def\@loadwithoptions#1#2#3{%
616   \expandafter\let\csname opt@#3.#1\expandafter\endcsname
617     \csname opt@\@currname.\@current\endcsname
618   #2{#3}}
619 \@onlypreamble\@loadwithoptions
```

(End definition for \@loadwithoptions.)

`\LoadClassWithOptions` Load class ‘#1’ with the current option list.

```
620 \def\LoadClassWithOptions{%
621   \@loadwithoptions\@clsextension\LoadClass}
622 \@onlypreamble\LoadClassWithOptions
```

(End definition for \LoadClassWithOptions.)

`\RequirePackageWithOptions` Load package ‘#1’ with the current option list.

```
623 </2ekernel>
624 < *2ekernel | latexrelease>
625 < latexrelease> \IncludeInRelease{2020/10/01}%
626 < latexrelease>           {\RequirePackageWithOptions}{Unused options issue}%
627 \def\RequirePackageWithOptions{%
```

The resetting of the unprocessed options is now done on a par package basis.

```
628   \AtEndOfPackage{\expandafter\let
629                     \csname unprocessedoptions-\@currname.\@current\endcsname
630                     \relax}%
631   \@loadwithoptions\@pkgextension\RequirePackage}
632 \@onlypreamble\RequirePackageWithOptions
633 </2ekernel | latexrelease>
634 < latexrelease> \EndIncludeInRelease
```

```

635 <latexrelease>\IncludeInRelease{0000/00/00}%
636 <latexrelease>                {\RequirePackageWithOptions}{Unused options issue}%
637 <latexrelease>
638 <latexrelease>\def\RequirePackageWithOptions{%
639 <latexrelease> \AtEndOfPackage{\let\@unprocessedoptions\relax}%
640 <latexrelease> \loadwithoptions\@pkgextension\RequirePackage}
641 <latexrelease>\EndIncludeInRelease
642 <*2ekernel>

```

(End definition for \RequirePackageWithOptions.)

\usepackage To begin with, \usepackage produces an error. This is reset by \documentclass.

```

643 \def\usepackage#1{%
644   \@latex@error
645     {\noexpand \usepackage before \string\documentclass}%
646     {\noexpand \usepackage may only appear in the document
647      preamble, i.e.,\MessageBreak
648      between \noexpand\documentclass and
649      \string\begin{document}.}%
650   \@gobble}
651 \@onlypreamble\usepackage

```

(End definition for \usepackage.)

\NeedsTeXFormat Check that the document is running on the correct system.

```

652 \def\NeedsTeXFormat#1{%
653   \def\reserved@a{#1}%
654   \ifx\reserved@a\fmtname
655     \expandafter\@needsformat
656   \else
657     \@latex@error{This file needs format ‘\reserved@a’%
658     \MessageBreak but this is ‘\fmtname’}{%
659     The current input file will not be processed
660     further,\MessageBreak
661     because it was written for some other flavor of
662     TeX.\MessageBreak\@ehd}%

```

If the file is not meant to be processed by L^AT_EX 2_ε we stop inputting it, but we do not end the run. We just end inputting the current file.

```

663   \endinput \fi}
664 \@onlypreamble\NeedsTeXFormat

665 \def\@needsformat{%
666   \@ifnextchar[%]
667     \@needsformat
668   {}
669 \@onlypreamble\@needsformat

670 \def\@needsformat[#1]{%
671   \ifl@t@r\fmtversion{#1}{}%
672   {\@latex@warning@no@line
673     {You have requested release ‘#1’ of LaTeX,\MessageBreak
674     but only release ‘\fmtversion’ is available}}
675 \@onlypreamble\@needsformat

```

(End definition for \NeedsTeXFormat.)

`\zap@space` `\zap@space foo⟨space⟩\@empty` removes all spaces from `foo` that are not protected by `{ }` groups.

```

676 \def\zap@space#1 #2{%
677   #1%
678   \ifx#2\@empty\else\expandafter\zap@space\fi
679   #2}

```

(End definition for `\zap@space`.)

`\@fileswithoptions` The common part of `\documentclass` and `\usepackage`.

```

680 \def\@fileswithoptions#1{%
681   \@ifnextchar[%]
682     {\@fileswith@ptions#1}%
683     {\@fileswith@ptions#1[]}}
684 \@onlypreamble\@fileswithoptions

685 \def\@fileswith@ptions#1[#2]#3{%
686   \@ifnextchar[%]
687     {\@fileswith@ptions#1[#{#2}]#3}%
688     {\@fileswith@ptions#1[#{#2}]#3[]}}
689 \@onlypreamble\@fileswith@ptions

```

Then we do some work.

First of all, we define the global variables. Then we look to see if the file has already been loaded. If it has, we check that it was first loaded with at least the current options. If it has not, we add the current options to the package options, set the default version to be 0000/00/00, and load the file if we can find it. Then we check the version number.

Finally, we restore the old file name, reset the default option, and we set the catcode of `@`.

For classes, we can immediately process the file. For other types, `#2` could be a comma separated list, so loop through, processing each one separately.

```

690 ⟨/2ekernel⟩
691 ⟨latexrelease⟩\IncludeInRelease{2020/10/01}%
692 ⟨latexrelease⟩      {\@fileswith@ptions}{ifx tests in \@fileswith@ptions}%
693 ⟨*2ekernel | latexrelease⟩
694 \def\@fileswith@ptions#1[#2]#3[#4]{%
695   \ifx#1\@clsextension
696     \ifx\@classoptionslist\relax
697       \xdef\@classoptionslist{\zap@space#2 \@empty}%

```

Save raw class list.

```

698     \gdef\@raw@classoptionslist{#2}%
699     \def\reserved@a{%
700       \@onefilewithoptions#3[#{#2}][#{#4}]#1%
701       \@documentclasshook}%
702   \else
703     \def\reserved@a{%
704       \@onefilewithoptions#3[#{#2}][#{#4}]#1%
705     \fi
706   \else

```

build up a list of calls to `\@onefilewithoptions` (one for each package) without thrashing the parameter stack.

```

707     \def\reserved@b##1,{%

```

If #1 is \@nnil we have reached the end of the list (older version used \@nil here but \@nil is undefined so \ifx equal to all undefined commands)

```

708 \ifx\@nnil##1\relax\else
If \ifx\@nnil##1\@nnil is true then #1 is (presumably) empty (Older code used \relax
which is slightly easier to get into #1 by mistake, which would spoil this test.)
709 \ifx\@nnil##1\@nnil\else
710 \noexpand\@onefilewithoptions##1[#{#2}][#{#4}]%
711 \noexpand\@pkgextension
712 \fi
713 \expandafter\reserved@b
714 \fi}%
715 \edef\reserved@a{\zap@space#3 \@empty}%
716 \edef\reserved@a{\expandafter\reserved@b\reserved@a,\@nnil,}%
717 \fi
718 \reserved@a}
719 </2ekernel | latexrelease>
720 <latexrelease>\EndIncludeInRelease
721 <latexrelease>\IncludeInRelease{2017/01/01}%
722 <latexrelease> \{\@fileswith@pti@ns}{\ifx tests in \@fileswith@pti@ns}%
723 <latexrelease>\def\@fileswith@pti@ns#1[#2]#3[#4]{%
724 <latexrelease> \ifx#1\@clsextension
725 <latexrelease> \ifx\@classoptionslist\relax
726 <latexrelease> \xdef\@classoptionslist{\zap@space#2 \@empty}%
727 <latexrelease> \def\reserved@a{%
728 <latexrelease> \@onefilewithoptions#3[#{#2}][#{#4}]#1%
729 <latexrelease> \@documentclasshook}%
730 <latexrelease> \else
731 <latexrelease> \def\reserved@a{%
732 <latexrelease> \@onefilewithoptions#3[#{#2}][#{#4}]#1}%
733 <latexrelease> \fi
734 <latexrelease> \else
735 <latexrelease> \def\reserved@b##1,{%
736 <latexrelease> \ifx\@nnil##1\relax\else
737 <latexrelease> \ifx\@nnil##1\@nnil\else
738 <latexrelease> \noexpand\@onefilewithoptions##1[#{#2}][#{#4}]%
739 <latexrelease> \noexpand\@pkgextension
740 <latexrelease> \fi
741 <latexrelease> \expandafter\reserved@b
742 <latexrelease> \fi}%
743 <latexrelease> \edef\reserved@a{\zap@space#3 \@empty}%
744 <latexrelease> \edef\reserved@a{\expandafter\reserved@b\reserved@a,\@nnil,}%
745 <latexrelease> \fi
746 <latexrelease> \reserved@a}
747 <latexrelease>\EndIncludeInRelease
748 <latexrelease>\IncludeInRelease{0000/00/00}%
749 <latexrelease> \{\@fileswith@pti@ns}{\ifx tests in \@fileswith@pti@ns}%
750 <latexrelease>\def\@fileswith@pti@ns#1[#2]#3[#4]{%
751 <latexrelease> \ifx#1\@clsextension
752 <latexrelease> \ifx\@classoptionslist\relax
753 <latexrelease> \xdef\@classoptionslist{\zap@space#2 \@empty}%
754 <latexrelease> \def\reserved@a{%
755 <latexrelease> \@onefilewithoptions#3[#{#2}][#{#4}]#1%

```



```

756 <latexrelease> \documentclasshook}%
757 <latexrelease> \else
758 <latexrelease> \def\reserved@a{%
759 <latexrelease> \onefilewithoptions#3[#{#2}][#{#4}]#1}%
760 <latexrelease> \fi
761 <latexrelease> \else
762 <latexrelease> \def\reserved@b##1,{%
763 <latexrelease> \ifx\@nil##1\relax\else
764 <latexrelease> \ifx\relax##1\relax\else
765 <latexrelease> \noexpand\onefilewithoptions##1[#{#2}][#{#4}]%
766 <latexrelease> \noexpand\@pkgextension
767 <latexrelease> \fi
768 <latexrelease> \expandafter\reserved@b
769 <latexrelease> \fi}%
770 <latexrelease> \edef\reserved@a{\zap@space#3 \empty}%
771 <latexrelease> \edef\reserved@a{%
772 <latexrelease> \expandafter\reserved@b\reserved@a,\@nil,}%
773 <latexrelease> \fi
774 <latexrelease> \reserved@a}
775 <latexrelease>\EndIncludeInRelease
776 (*2ekernel)

777 \@onlypreamble\@fileswith@ptions

```

This macro is used when loading packages or classes.

`\load@onefilewithoptions` Have the main argument as #1, so we only need one `\expandafter` above.

```

778 </2ekernel>
779 (*2ekernel | latexrelease)
780 <latexrelease>\IncludeInRelease{2020/10/01}%
781 <latexrelease> \{\onefilewithoptions\}{Hooks and unused options issue}%
782 \def\@onefilewithoptions#1[#2][#3]#4{%

```

We have to sanitise file names, so that something like

```

\usepackage{some/local/path/array}
\usepackage{array}

```

won't load `array.sty` twice. It is remotely possible that those are two different files, but as a matter of principles, we will consider that the base file name uniquely identifies a package, regardless of where it lives. This assumption already holds for file hooks, for example, which address the hook to a file by its base name only.

We'll use `\@expl@@@filehook@set@curr@file@@nNN` to parse the file name and return the `<path>` and `<base+ext>` in separate token lists. Further ahead, most operations use `\@currname` which doesn't have a path attached to it; only few actions prepend `\@currpath` to `\@currname` (namely loading, as we have to respect the given path).

A file substitution isn't followed just yet because at this point we are parsing user input, so the file is still what the user asked for, and not the file actually loaded.

```

783 \@expl@@@filehook@set@curr@file@@nNN{#1.#4}\reserved@a\reserved@b
784 \edef\reserved@c{\def\noexpand\reserved@c####1%
785 \detokenize\expandafter{\expanded{.#4}}}%
786 \noexpand\@nil{\def\noexpand\reserved@a{####1}}}\reserved@c
787 \expandafter\reserved@c\reserved@a\@nil
788 \@pushfilename
789 \xdef\@currname{\string@makeletter\reserved@a}%

```

```

790 \xdef\@currpath{\ifx\reserved@b\@empty\else\reserved@b/\fi}%
791 \global\let\@currentx#4%

```

The command `\ver@<file>.<ext>` is used to signal that a package is already loaded, either because it is in fact loaded, or because its loading was suppressed. In minimal installations, said package may not exist but still have its loading suppressed with `\ver@<file>.<ext>`, so before checking if the file exists we have to check that we do need to load it with `\@ifl@aded`. If we don't, then there's no point in checking for a typo or load-disabling.

```

792 \@ifl@aded\@currentx\@currname

```

If the package is already loaded, check that there were no option clashes:

```

793 {\@if@options\@currentx\@currname}{#2}{}%
794 {\@latex@error
795   {Option clash for \@cls@pkg\space \@currname}%
796   {The package \@currname\space has already been loaded
797   with options:\MessageBreak
798   \space\space[\@optionlist{\@currname.\@currentx}]\MessageBreak
799   There has now been an attempt to load it
800   with options\MessageBreak
801   \space\space[#2]\MessageBreak
802   Adding the global options:\MessageBreak
803   \space\space
804   \@optionlist{\@currname.\@currentx},#2\MessageBreak
805   to your \noexpand\documentclass declaration may fix this.%
806   \MessageBreak
807   Try typing \space <return> \space to proceed.}}%
808 \@firstofone}%
809 {\makeatletter

```

The next line seems to be necessary for 2.09 compatibility (the way the code is written there) This seems questionable and should be look at as in 2e it is definitely unnecessary at this point!

```

810 \@reset@ptions

```

First we take the `<name>` and `<ext>` given in the argument and check if the file exists, and issue an error otherwise asking for a correction with `\@missingfileerror`. For checking if the file exists we use `\@currpath` (usually empty) before `\@currname`.

```

811 \@IfFileExists{\@currpath\@currname.\@currentx}{}%
812 {\@missing@onefilewithoptions{#2}}%

```

If `\@currname` is empty (the user replied to the “Enter file name” prompt with `<RETURN>`), so stop here (do `\@popfilename` to pop the item just added above).

This `\@gobble` omits the date check at the end.

```

813 \ifx\@currname\@empty
814 \expandafter\@gobble
815 \else

```

If the file exists, check if it was load-prevented, and otherwise do the bookkeeping with `\@filehook@file@push` then call `\set@curr@file` to set `\@curr@file` (and do any required substitution), then actually load the class/package with `\load@onefile@withoptions`. `\set@curr@file` also needs the file path.

```

816 \@disable@packageload@do{\@currname.\@currentx}%
817 {\@expl@@@filehook@file@push@@
818 \set@curr@file{\@currpath\@currname.\@currentx}%
819 \@filehook@set@CurrentFile

```

The `\set@curr@file` line above might have replaced the file, so `\@currname` and `\@currentx` may no longer hold the actual package being loaded, so in that case we need to update these two token lists (`\@curr@file` holds the file name after replacement, so we parse that).

The requested file is saved in `\@currpkg@reqd` to be used in `\InputIfFileExists` later: if the updated `\@currname` and `\@currentx` are used we lose track of the substitution, so `\CurrentFile` and `\CurrentFileUsed` will be (incorrectly) the same.

```
820         \expandafter\@swaptwoargs\expandafter
821         {\expandafter{\@currpkg@reqd}}}%
822         {% <
```

`\@currpkg@reqd` doesn't take a path because it is used later to assign `\opt@...` and `\ver@...`

```
823         \edef\@currpkg@reqd{\@currname.\@currentx}%
824         \ifx\CurrentFile\CurrentFileUsed
825         \else
826         \filename@parse\@curr@file
827         \edef\@currpath{\string@makeletter\filename@area}%
828         \edef\@currname{\string@makeletter\filename@base}%
829         \edef\@currentx{\string@makeletter\filename@ext}%
830         \fi
831         \load@onefile@withoptions{#2}%
832         \def\@currpkg@reqd%\@currpkg@reqd{
833         }% >
```

Now just clean up and exit.

```
834         \expl@@@filehook@file@pop@@}%
835         \expandafter\@firstofone
836         \fi}%
```

Except in the case where `\@currname` is empty, the date is checked against the date marked in the package file:

```
837         {\@ifl@ter\@currentx{\@currname}{#3}{}}%
838         {\@latex@warning@no@line
839         {You have requested,\on@line,
840         version\MessageBreak
841         '#3' of \@cls@pkg\space \@currname,\MessageBreak
842         but only version\MessageBreak
843         '\csname ver@\@currname.\@currentx\endcsname'\MessageBreak
844         is available}}}%

845         \ifx\@currentx\@clsextension\let\LoadClass\@twoloadclasserror\fi}%
846         \@popfilename
847         \@reset@options}
```

```
848 \let\@currpkg@reqd\@empty
```

```
849 \@onlypreamble\@onefilewithoptions
```

The kernel no longer uses `\@unprocessedoptions`

```
850 \let\@unprocessedoptions\@undefined
```

Now the action taken when a file is not found. Path must be included here as it eventually leads to a file lookup.

```
851 \def\@missing@onefilewithoptions#1{%
```

```

852 \missingfileerror{\@currpath\@currname}\@currentx
853 \global\let\@currpath\@missingfile@area
854 \global\let\@currname\@missingfile@base
855 \global\let\@currentx\@missingfile@ext}

```

Now the code that actually does the file loading:

```

\load@onefile@withoptions 856 \def\load@onefile@withoptions#1{%
857 \let\CurrentOption\empty
858 \reset@options

```

Grab everything in a macro, so the parameter stack is popped before any processing begins.

```

859 \def\reserved@a{%
860 \pass@options\@currentx{#1}\@currname}%

861 \expandafter\let
862 \csname opt@\@currpkg@reqd\expandafter\endcsname
863 \csname opt@\@currname.\@currentx\endcsname
864 \global\expandafter
865 \let\csname ver@\@currname.\@currentx\endcsname\empty

```

We initialize `\...-h@@k` here and only if we load the file so that it remains undefined otherwise.

```

866 \expandafter\let\csname\@currname.\@currentx-h@@k\endcsname\empty

```

When the current extension is `\@pkgextension` we are loading a package otherwise, if it is `\@clsextension`, a class, so depending on that we execute different hooks. If the extension is neither, then it is another type of file without special hooks.

```

867 %-----
868 \ifx\@currentx\@pkgextension
869 \UseHook{package/before}%
870 \UseHook{package/before/\@currname}%
871 \else
872 \ifx\@currentx\@clsextension
873 \UseHook{class/before}%
874 \UseHook{class/before/\@currname}%
875 \fi
876 \fi

```

Now actually load the file (at this point we are certain it exists, but use `\InputIfFileExists` so that file hooks are executed). `\@currpath` is needed here too.

```

877 \InputIfFileExists{\@currpath\@currpkg@reqd}{}%
878 {\@latex@error
879 {The \@cls@pkg\space\@currpkg@reqd\space failed to load}\@ehd}%
880 %-----

```

In older versions of the code `\@unprocessedoptions` would generate an error for each specified option in a package unless a `\ProcessOptions` has appeared in the package file.

This has changed in 2020. We now use a separate macro per package to avoid interference in case of nested packages. The whole code for handling this issue (GitHub 22) was provided by Hironobu Yamashita, thanks for that.

```

881 \expandafter\let\csname unprocessedoptions-\@currname.\@currentx\endcsname
882 \@unprocessedoptions
883 \csname\@currname.\@currentx-h@@k\endcsname

```

```

884 \expandafter\let\csname\@currname.\@currentx-h@@k\endcsname
885 \undefined

```

Catch the case where the packages has handled the options and redefined `\@unprocessedoptions` to `\relax` (old interface). In that case no error should be produced.

```

886 \ifx\@unprocessedoptions\relax
887 \let\@unprocessedoptions\undefined

```

Otherwise run the per package set of unused options.

```

888 \else
889 \csname unprocessedoptions-\@currname.\@currentx\endcsname
890 \fi

```

In either case we drop the macro afterwards as it is no longer needed.

```

891 \expandafter\let
892 \csname unprocessedoptions-\@currname.\@currentx\endcsname
893 \undefined

```

And same procedure, James, when we are finished loading, except that the hook order is now reversed.

```

894 %-----
895 \ifx\@currentx\@pkgextension
896 \UseHook{package/after/\@currname}%
897 \UseHook{package/after}%
898 \else
899 \ifx\@currentx\@clsextension
900 \UseHook{class/after/\@currname}%
901 \UseHook{class/after}%
902 \fi
903 \fi}%
904 %-----
905 \@ifl@aded\@currentx\@currname{}\{\reserved@a{}\}

906 </2ekernel | latexrelease>
907 <latexrelease>\EndIncludeInRelease
908 <latexrelease>\IncludeInRelease{0000/00/00}%
909 <latexrelease> \{\@onefilewithoptions\}{Hooks and unused options issue}%
910 <latexrelease>

```

Because of the way `\@onfilewithoptions` is changed for rollback handling below we have to define `\load@onefilewithoptions` when rolling back!

```

911 <latexrelease>\def\load@onefilewithoptions#1[#2][#3]#4{%
912 <latexrelease> \@pushfilename
913 <latexrelease> \xdef\@currname{#1}%
914 <latexrelease> \global\let\@currentx#4%
915 <latexrelease> \let\CurrentOption\@empty
916 <latexrelease> \@reset@ptions
917 <latexrelease> \makeatletter
918 <latexrelease> \def\reserved@a{%
919 <latexrelease> \@ifl@aded\@currentx{#1}%
920 <latexrelease> \{\@if@ptions\@currentx{#1}\{#2\}\}%
921 <latexrelease> \{\@latex@error
922 <latexrelease> \{Option clash for \@cls@pkg\space #1\}%
923 <latexrelease> \{The package #1 has already been loaded
924 <latexrelease> with options:\MessageBreak

```

```

925 <latexrelease> \space\space[\@optionlist{#1.\@currentt}]\MessageBreak
926 <latexrelease> There has now been an attempt to load it
927 <latexrelease> with options\MessageBreak
928 <latexrelease> \space\space[#2]\MessageBreak
929 <latexrelease> Adding the global options:\MessageBreak
930 <latexrelease> \space\space
931 <latexrelease> \@optionlist{#1.\@currentt},#2\MessageBreak
932 <latexrelease> to your \noexpand\documentclass declaration may fix this.%
933 <latexrelease> \MessageBreak
934 <latexrelease> Try typing \space <return> \space to proceed.}}}%
935 <latexrelease> {\@passoptions\@currentt{#2}{#1}%
936 <latexrelease> \global\expandafter
937 <latexrelease> \let\csname ver@\@currname.\@currentt\endcsname\@empty
938 <latexrelease> \expandafter\let\csname\@currname.\@currentt-h@@k\endcsname\@empty
939 <latexrelease> \InputIfFileExists
940 <latexrelease> {\@currname.\@currentt}%
941 <latexrelease> {}%
942 <latexrelease> {\@missingfileerror\@currname\@currentt}%
943 <latexrelease> \let\@unprocessedoptions\@unprocessedoptions
944 <latexrelease> \csname\@currname.\@currentt-h@@k\endcsname
945 <latexrelease> \expandafter\let\csname\@currname.\@currentt-h@@k\endcsname
946 <latexrelease> \undefined
947 <latexrelease> \@unprocessedoptions}%
948 <latexrelease> \@ifl@ter\@currentt{#1}{#3}{}%
949 <latexrelease> {\@latex@warning@no@line
950 <latexrelease> {You have requested,\on@line,
951 <latexrelease> version\MessageBreak
952 <latexrelease> ‘#3’ of \cls@pkg\space #1,\MessageBreak
953 <latexrelease> but only version\MessageBreak
954 <latexrelease> ‘\csname ver@#1.\@currentt\endcsname’\MessageBreak
955 <latexrelease> is available}}}%
956 <latexrelease> \ifx\@currentt\@clsextension\let\LoadClass\@twoloadclasserror\fi
957 <latexrelease> \@popfilename
958 <latexrelease> \@resetoptions}%
959 <latexrelease> \reserved@a}
960 <latexrelease>
961 <latexrelease>\let \load@onefile@withoptions \undefined
962 <latexrelease>\let \@missing@onefile@withoptions \undefined
963 <latexrelease>
964 <latexrelease>\EndIncludeInRelease
965 <*2ekernel>

```

(End definition for \@fileswithoptions and others.)

\@@fileswith@pti@ns Save the definition (for error checking).

```

966 \let\@@fileswith@pti@ns\@fileswith@pti@ns
967 \@onlypreamble\@@fileswith@pti@ns

```

(End definition for \@@fileswith@pti@ns.)

\@resetoptions Reset the default option, and clear lists of declared options.

```

968 \def\@resetoptions{%
969 \global\ifx\@currentt\@clsextension
970 \let\default@ds\OptionNotUsed
971 \else

```

```

972 \let\default@ds\@unknownoptionerror
973 \fi
974 \global\let\ds@\@empty
975 \global\let\@declaredoptions\@empty}
976 \@onlypreamble\@reset@ptions

```

(End definition for \@reset@ptions.)

4.1 Hooks

Allow code to be saved to be executed at specific later times.

Save things in macros, I considered using toks registers, (and \addto@hook from the NFSS code, that would require stacking the contents in the case of required packages, so just generate a new macro for each package.

\@begindocumenthook Stuff to appear at the beginning or end of the document.

```

\@enddocumenthook
977 \ifx\@begindocumenthook\@undefined
978 \let\@begindocumenthook\@empty
979 \fi
980 \let\@enddocumenthook\@empty

```

(End definition for \@begindocumenthook and \@enddocumenthook.)

\AtEndOfPackage The access functions.

```

\AtEndOfClass
\AtBeginDocument
\AtEndDocument
981 \def\AtEndOfPackage{%
982 \expandafter\g@addto@macro\csname\@currname.\@current-h@@k\endcsname}
983 \let\AtEndOfClass\AtEndOfPackage
984 \@onlypreamble\AtEndOfPackage
985 \@onlypreamble\AtEndOfClass
986 </2ekernel>
987 <*2ekernel | latexrelease>
988 <latexrelease>\IncludeInRelease{2020/10/01}%
989 <latexrelease> \AtBeginDocument}{Use hook system}%
990 \DeclareRobustCommand\AtBeginDocument{\AddToHook{begindocument}}
991 \DeclareRobustCommand\AtEndDocument {\AddToHook{enddocument}}
992 %\DeclareRobustCommand\AtEndDocument {\AddToHook{env/document/end}} % alternative impl
993 </2ekernel | latexrelease>
994 <latexrelease>\EndIncludeInRelease
995 <latexrelease>\IncludeInRelease{0000/00/00}%
996 <latexrelease> \AtBeginDocument}{Use hook system}%
997 <latexrelease>
998 <latexrelease>\DeclareRobustCommand\AtBeginDocument{\g@addto@macro\@begindocumenthook}
999 <latexrelease>\DeclareRobustCommand\AtEndDocument{\g@addto@macro\@enddocumenthook}
1000 <latexrelease>
1001 <latexrelease>\EndIncludeInRelease
1002 <*2ekernel>
1003 \@onlypreamble\AtBeginDocument

```

(End definition for \AtEndOfPackage and others.)

`\@cls@pkg` The current file type.

```
1004 \def\@cls@pkg{%
1005   \ifx\@current\@clsextension
1006     document class%
1007   \else
1008     package%
1009   \fi}
1010 \onlypreamble\@cls@pkg

(End definition for \@cls@pkg.)
```

`\@unknownoptionerror` Bad option.

```
1011 \def\@unknownoptionerror{%
1012   \@latex@error
1013     {Unknown option ‘\CurrentOption’ for \@cls@pkg\space‘\@currname’}%
1014     {The option ‘\CurrentOption’ was not declared in
1015       \@cls@pkg\space‘\@currname’, perhaps you\MessageBreak
1016         misspelled its name.
1017       Try typing \space <return>
1018       \space to proceed.}}
1019 \onlypreamble\@unknownoptionerror

(End definition for \@unknownoptionerror.)
```

`\@unprocessedoptions` Declare an error for each option, unless a `\ProcessOptions` occurred.

```
1020 \def\@unprocessedoptions{%
1021   \ifx\@current\@pkgextension
1022     \edef\@curroptions{\@optionlist{\@currname.\@current}}%
1023     \@for\CurrentOption:=\@curroptions\do{%
1024       \ifx\CurrentOption\@empty\else\@unknownoptionerror\fi}%
1025   \fi}
1026 \onlypreamble\@unprocessedoptions
1027 \onlypreamble\@unprocessedoptions

(End definition for \@unprocessedoptions.)
```

`\@badrequireerror` `\RequirePackage` or `\LoadClass` occurs in the options section.

```
1028 \def\@badrequireerror#1[#2]#3[#4]{%
1029   \@latex@error
1030     {\noexpand\RequirePackage or \noexpand\LoadClass
1031       in Options Section}%
1032     {The \@cls@pkg\space ‘\@currname’ is defective.\MessageBreak
1033       It attempts to load ‘#3’ in the options section, i.e.,\MessageBreak
1034       between \noexpand\DeclareOption and \string\ProcessOptions.}}
1035 \onlypreamble\@badrequireerror

(End definition for \@badrequireerror.)
```

`\@twoloadclasserror` Two `\LoadClass` in a class.

```
1036 \def\@twoloadclasserror{%
1037   \@latex@error
1038     {Two \noexpand\LoadClass commands}%
1039     {You may only use one \noexpand\LoadClass in a class file}}
1040 \onlypreamble\@twoloadclasserror
```


(End definition for \twoloadclasserror.)

\twoclasseserror Two \documentclass or \documentstyle.

```
1041 \def\twoclasseserror#1{%
1042   \latex@error
1043     {Two \noexpand\documentclass or \noexpand\documentstyle commands}%
1044     {The document may only declare one class.}\@gobble}
1045 \@onlypreamble\twoclasseserror
```

(End definition for \twoclasseserror.)

4.2 Providing shipment

\two@digits Prefix a number less than 10 with ‘0’.

```
1046 \def\two@digits#1{\ifnum#1<10 0\fi\number#1}
```

(End definition for \two@digits.)

\filecontents This environment implements inline files. The star-form does not write extra comments into the file.

\endfilecontents

```
1047 </2kernel>
1048 <*2kernel | latexrelease>
1049 <latexrelease>\IncludeInRelease{2020/10/01}%
1050 <latexrelease>          {\filec@ntents}{Define \q@curr@file directly (gh/220)}%
1051 %
```

We use @tempswa to mean no preamble writing and reuse @filesw to indicate no overwriting:

```
1052 \def\filecontents{\@tempswatrue\@fileswtrue
1053   \ifnextchar[\filec@ntents@opt\filec@ntents
1054 }
1055 \@namedef{filecontents*}{\@tempswafalse\@fileswtrue
1056   \ifnextchar[\filec@ntents@opt\filec@ntents
1057 }
```

To handle the optional argument we execute for each option the command \filec@ntents@OPTION if it exist or complain about unknown option.

```
1058 \def\filec@ntents@opt[#1]{%
1059   \edef\@fortmp{\zap@space#1 \@empty}%
1060   \@for\reserved@a:=\@fortmp\do{%
1061     \ifcsname filec@ntents@\reserved@a\endcsname
1062       \csname filec@ntents@\reserved@a\endcsname
1063     \else
1064       \latex@error{Unknown filecontents option \reserved@a}%
1065       {Valid options are force (or overwrite), nosearch, noheader}%
1066       \fi}%
1067   \filec@ntents
1068 }
```

Option force (or overwrite) changes the overwriting switch

```
1069 \let\filec@ntents@force\@fileswfalse
1070 \let\filec@ntents@overwrite\@fileswfalse % alternative name
```

and option `noheader` the preamble switch (which is equivalent to using the star form of the environment).

```
1071 \let\filec@tents@noheader\@tempswafalse
```

Option `nosearch` only checks the current directory not the whole \TeX tree for the existence of the file to write.

```
1072 \def\filec@tents@nosearch{%
1073   \let\filec@tents@checkdir\@currdir
1074   \def\filec@tents@where{in current directory}}
```

By default we search the whole tree:

```
1075 \let\filec@tents@checkdir\@empty
1076 \def\filec@tents@where{exists on the system}

1077 \begingroup%
1078 \@tempcnta=1
1079 \loop
1080   \catcode\@tempcnta=12 %
1081   \advance\@tempcnta\@ne %
1082   \ifnum\@tempcnta<32 %
1083   \repeat %
1084   \catcode\@* =11 %
1085   \catcode\^^M\active%
1086   \catcode\^^L\active\let^^L\relax%
1087   \catcode\^^I\active%

1088 \gdef\filec@tents#1{%
1089   \set@curr@file{\filec@tents@checkdir#1}%
1090   \edef\q@curr@file{"\@curr@file"}%
```

Lua \TeX has more writes (and 18 is safe here).

```
1091 \chardef\reserved@c\ifx\directlua\@undefined 15 \else 127 \fi%
1092 \openin\@inputcheck\q@curr@file \space %
1093 \ifeof\@inputcheck%
1094   \@latex@warning@no@line%
1095     {Writing file '\@currdir\@curr@file'}%
1096   \ch@ck7\reserved@c\write\relax%
1097   \immediate\openout\reserved@c\q@curr@file\relax%
1098 \else%

1099   \if@filesw%
1100     \@latex@warning@no@line%
1101       {File '\@curr@file' already \filec@tents@where.\MessageBreak%
1102         Not generating it from this source}%
1103     \let\write\@gobbletwo%
1104     \let\closeout\@gobble%
1105   \else%
```

If we are overwriting, we try to make sure that the user is not by mistake overwriting the input file (`\jobname`). Of course, this only works for input files ending in `.tex`. If a different extension is used there is no way to see that we are overwriting ourselves!

```
1106   \edef\reserved@b{\detokenize\expandafter{\jobname}}%
1107   \ifx\@curr@file\reserved@b%
1108     \@fileswtrue%
1109   \else%
```

```

1110      \edef\reserved@b{\reserved@b\detokenize{.tex}}%
1111      \ifx\@curr@file\reserved@b
1112        \@fileswtrue%
1113      \fi%
1114    \fi%

```

We allocate a write channel but we open it only if it is (hopefully) safe. If not opened that means we are going to write on the terminal.

```

1115      \ch@ck7\reserved@c\write\relax%
1116      \if@filesw% % Foul ... trying to overwrite \jobname!
1117      \@latex@error{Trying to overwrite '\jobname.tex'}{You can't %
1118        write to the file you are reading from!\MessageBreak%
1119        Data is written to screen instead.}%
1120      \else%
1121        \@latex@warning@no@line%
1122          {Writing or overwriting file '@curr@dir\@curr@file'}%
1123        \immediate\openout\reserved@c\q@curr@file\relax%
1124      \fi%
1125    \fi%
1126  \fi%

```

Closing the \@inputcheck is done here to avoid having to do this in each branch.

```

1127  \closein\@inputcheck%
1128  \if@tempw%
1129    \immediate\write\reserved@c{%
1130      \@percentchar\@percentchar\space%
1131      \expandafter\@gobble\string\LaTeX2e file '@curr@file'^~J%
1132      \@percentchar\@percentchar\space generated by the %
1133      '@currentenv' \expandafter\@gobblefour\string\newenvironment^~J%
1134      \@percentchar\@percentchar\space from source '\jobname' on %
1135      \number\year/\two@digits\month/\two@digits\day.^~J%
1136      \@percentchar\@percentchar}%
1137  \fi%
1138  \let\do\@makeother\dospecials%

```

If there are active characters in the upper half (e.g., from inputenc there would be confusion so we render everything harmless.

```

1139  \count@ 128\relax%
1140  \loop%
1141    \catcode\count@ 11\relax%
1142    \advance\count@ \@ne%
1143    \ifnum\count@<\@ccclvi%
1144  \repeat%
1145  \edef\E{\@backslashchar end\string{\@currentenv\string}}%
1146  \edef\reserved@b{%
1147    \def\noexpand\reserved@b%
1148      #####1\E####2\E####3\relax}%
1149  \reserved@b{%
1150    \ifx\relax##3\relax%

```

There was no \end{filecontents}

```

1151    \immediate\write\reserved@c{##1}%
1152  \else%

```

There was a `\end{filecontents}`, so stop this time.

```

1153      \edef^^M{\noexpand\end{\@currentvir}}%
1154      \ifx\relax##1\relax%
1155      \else%

```

Text before the `\end`, write it with a warning.

```

1156      \@latex@warning{Writing text ‘##1’ before %
1157      \string\end{\@currentvir}\MessageBreak
1158      as last line of \@curr@file}%
1159      \immediate\write\reserved@c{##1}%
1160      \fi%
1161      \ifx\relax##2\relax%
1162      \else%

```

Text after the `\end`, ignore it with a warning.

```

1163      \@latex@warning{%
1164      Ignoring text ‘##2’ after \string\end{\@currentvir}}%
1165      \fi%
1166      \fi%
1167      ^^M}%

```

```

1168      \catcode'\^^L\active%
1169      \let\L\@undefined%
1170      \def^^L{\expandafter\ifx\csname L\endcsname\relax\fi ^^J^^J}%
1171      \catcode'\^^I\active%
1172      \let\I\@undefined%
1173      \def^^I{\expandafter\ifx\csname I\endcsname\relax\fi\space}%
1174      \catcode'\^^M\active%
1175      \edef^^M##1^^M{%
1176      \noexpand\reserved@b##1\E\E\relax}}%
1177      \endgroup%

1178      </2ekernel | latexrelease>
1179      <latexrelease>\EndIncludeInRelease
1180      <latexrelease>\IncludeInRelease{2019/10/01}%
1181      <latexrelease>      {\filec@ntents}{Spaces in file names + optional arg}%
1182      <latexrelease>
1183      <latexrelease>\def\filec@ntents{\@tempwatrue\@fileswtrue
1184      <latexrelease>  \ifnextchar[\filec@ntents@opt\filec@ntents
1185      <latexrelease>}\}
1186      <latexrelease>\@namedef{filec@ntents*}{\@tempwafalse\@fileswtrue
1187      <latexrelease>  \ifnextchar[\filec@ntents@opt\filec@ntents
1188      <latexrelease>}\}
1189      <latexrelease>\def\filec@ntents@opt[#1]{%
1190      <latexrelease>  \edef\@fortmp{\zap@space#1 \@empty}%
1191      <latexrelease>  \@for\reserved@a:=\@fortmp\do{%
1192      <latexrelease>    \ifcsname filec@ntents@\reserved@a\endcsname
1193      <latexrelease>      \csname filec@ntents@\reserved@a\endcsname
1194      <latexrelease>    \else
1195      <latexrelease>      \@latex@error{Unknown filec@ntents option \reserved@a}%
1196      <latexrelease>      {Valid options are force (or overwrite), nosearch, noheader}%
1197      <latexrelease>    \fi}%
1198      <latexrelease>  \filec@ntents
1199      <latexrelease>}\}
1200      <latexrelease>\let\filec@ntents@force\@fileswfalse

```

```

1201 <latexrelease>\let\filec@ntents@overwrite\@fileswfalse % alternative name
1202 <latexrelease>\let\filec@ntents@noheader\@tempswafalse
1203 <latexrelease>\def\filec@ntents@nosearch{%
1204 <latexrelease> \let\filec@ntents@checkdir\@currdir
1205 <latexrelease> \def\filec@ntents@where{in current directory}}
1206 <latexrelease>\let\filec@ntents@checkdir\@empty
1207 <latexrelease>\def\filec@ntents@where{exists on the system}
1208 <latexrelease>\begingroup%
1209 <latexrelease>\@tempcnta=1
1210 <latexrelease>\loop
1211 <latexrelease> \catcode\@tempcnta=12 %
1212 <latexrelease> \advance\@tempcnta\@ne %
1213 <latexrelease>\ifnum\@tempcnta<32 %
1214 <latexrelease>\repeat %
1215 <latexrelease>\catcode'\*=11 %
1216 <latexrelease>\catcode'\^~M\active%
1217 <latexrelease>\catcode'\^~L\active\let^~L\relax%
1218 <latexrelease>\catcode'\^~I\active%
1219 <latexrelease>\gdef\filec@ntents#1{%
1220 <latexrelease> \set@curr@file{\filec@ntents@checkdir#1}%
1221 <latexrelease> \edef\q@curr@file{\expandafter\quote@name\expandafter{\@curr@file}}%
1222 <latexrelease> \chardef\reserved@c\ifx\directlua\@undefined 15 \else 127 \fi%
1223 <latexrelease> \openin\@inputcheck\q@curr@file \space %
1224 <latexrelease> \ifeof\@inputcheck%
1225 <latexrelease> \@latex@warning@no@line%
1226 <latexrelease> {Writing file '\@currdir\@curr@file'}%
1227 <latexrelease> \ch@ck7\reserved@c\write\relax%
1228 <latexrelease> \immediate\openout\reserved@c\q@curr@file\relax%
1229 <latexrelease> \else%
1230 <latexrelease> \if@filesw%
1231 <latexrelease> \@latex@warning@no@line%
1232 <latexrelease> {File '\@curr@file' already \filec@ntents@where.\MessageBreak%
1233 <latexrelease> Not generating it from this source}%
1234 <latexrelease> \let\write\@gobbles%
1235 <latexrelease> \let\closeout\@gobble%
1236 <latexrelease> \else%
1237 <latexrelease> \edef\reserved@a{#1}%
1238 <latexrelease> \edef\reserved@a{\detokenize\expandafter{\reserved@a}}%
1239 <latexrelease> \edef\reserved@b{\detokenize\expandafter{\jobname}}%
1240 <latexrelease> \ifx\reserved@a\reserved@b%
1241 <latexrelease> \@fileswtrue%
1242 <latexrelease> \else%
1243 <latexrelease> \edef\reserved@b{\reserved@b\detokenize{.tex}}%
1244 <latexrelease> \ifx\reserved@a\reserved@b
1245 <latexrelease> \@fileswtrue%
1246 <latexrelease> \fi%
1247 <latexrelease> \fi%
1248 <latexrelease> \ch@ck7\reserved@c\write\relax%
1249 <latexrelease> \if@filesw% % Foul ... trying to overwrite \jobname!
1250 <latexrelease> \@latex@error{Trying to overwrite '\jobname.tex'}{You can't %
1251 <latexrelease> write to the file you are reading from!\MessageBreak%
1252 <latexrelease> Data is written to screen instead.}%
1253 <latexrelease> \else%
1254 <latexrelease> \@latex@warning@no@line%

```

```

1255 <latexrelease>          {Writing or overwriting file ‘\@currdir\@curr@file’}%
1256 <latexrelease>          \immediate\openout\reserved@c\q\curr@file\relax%
1257 <latexrelease>          \fi%
1258 <latexrelease>          \fi%
1259 <latexrelease> \fi%
1260 <latexrelease> \closein\@inputcheck%
1261 <latexrelease> \if@tempwa%
1262 <latexrelease>          \immediate\write\reserved@c{%
1263 <latexrelease>          \@percentchar\@percentchar\space%
1264 <latexrelease>          \expandafter\@gobble\string\LaTeX2e file ‘\@curr@file’^^J%
1265 <latexrelease>          \@percentchar\@percentchar\space generated by the %
1266 <latexrelease>          ‘\@currentvir’ \expandafter\@gobblefour\string\newenvironment^^J%
1267 <latexrelease>          \@percentchar\@percentchar\space from source ‘\jobname’ on %
1268 <latexrelease>          \number\year/\two@digits\month/\two@digits\day.^^J%
1269 <latexrelease>          \@percentchar\@percentchar}%
1270 <latexrelease> \fi%
1271 <latexrelease> \let\do\@makeoother\dospecials%
1272 <latexrelease> \count@ 128\relax%
1273 <latexrelease> \loop%
1274 <latexrelease>          \catcode\count@ 11\relax%
1275 <latexrelease>          \advance\count@ \@ne%
1276 <latexrelease>          \ifnum\count@<\@ccclvi%
1277 <latexrelease> \repeat%
1278 <latexrelease> \edef\E{\@backslashchar end\string{\@currentvir\string}}%
1279 <latexrelease> \edef\reserved@b{%
1280 <latexrelease>          \def\noexpand\reserved@b%
1281 <latexrelease>          #####1\E####2\E####3\relax}%
1282 <latexrelease> \reserved@b{%
1283 <latexrelease>          \ifx\relax##3\relax%
1284 <latexrelease>          \immediate\write\reserved@c{##1}%
1285 <latexrelease>          \else%
1286 <latexrelease>          \edef^^M{\noexpand\end{\@currentvir}}%
1287 <latexrelease>          \ifx\relax##1\relax%
1288 <latexrelease>          \else%
1289 <latexrelease>          \@latex@warning{Writing text ‘##1’ before %
1290 <latexrelease>          \string\end{\@currentvir}\MessageBreak as last line of \@curr@file}%
1291 <latexrelease>          \immediate\write\reserved@c{##1}%
1292 <latexrelease>          \fi%
1293 <latexrelease>          \ifx\relax##2\relax%
1294 <latexrelease>          \else%
1295 <latexrelease>          \@latex@warning{%
1296 <latexrelease>          Ignoring text ‘##2’ after \string\end{\@currentvir}}%
1297 <latexrelease>          \fi%
1298 <latexrelease>          \fi%
1299 <latexrelease>          ^^M}%
1300 <latexrelease> \catcode‘^^L\active%
1301 <latexrelease> \let\L\@undefined%
1302 <latexrelease> \def^^L{\expandafter\ifx\csname L\endcsname\relax\fi ^^J^^J}%
1303 <latexrelease> \catcode‘^^I\active%
1304 <latexrelease> \let\I\@undefined%
1305 <latexrelease> \def^^I{\expandafter\ifx\csname I\endcsname\relax\fi\space}%
1306 <latexrelease> \catcode‘^^M\active%
1307 <latexrelease> \edef^^M##1^^M{%
1308 <latexrelease>          \noexpand\reserved@b##1\E\E\relax}}%

```

```

1309 <latexrelease>\endgroup%
1310 <latexrelease>\EndIncludeInRelease
1311 <latexrelease>\IncludeInRelease{0000/00/00}%
1312 <latexrelease>    {\filec@ntents}{Spaces in file names + optional arg}%
1313 <latexrelease>
1314 <latexrelease>\let\filec@ntents@opt      \@undefined
1315 <latexrelease>\let\filec@ntents@force    \@undefined
1316 <latexrelease>\let\filec@ntents@overwrite \@undefined
1317 <latexrelease>\let\filec@ntents@noheader \@undefined
1318 <latexrelease>\let\filec@ntents@nosearch \@undefined
1319 <latexrelease>\let\filec@ntents@checkdir \@undefined
1320 <latexrelease>\let\filec@ntents@where    \@undefined
1321 <latexrelease>
1322 <latexrelease>\begingroup%
1323 <latexrelease>\@tempcnta=1
1324 <latexrelease>\loop
1325 <latexrelease>  \catcode\@tempcnta=12 %
1326 <latexrelease>  \advance\@tempcnta\@ne %
1327 <latexrelease>\ifnum\@tempcnta<32 %
1328 <latexrelease>\repeat %
1329 <latexrelease>\catcode'\*=11 %
1330 <latexrelease>\catcode'\^^M\active%
1331 <latexrelease>\catcode'\^^L\active\let^^L\relax%
1332 <latexrelease>\catcode'\^^I\active%
1333 <latexrelease>
1334 <latexrelease>\gdef\filec@ntents#1{%
1335 <latexrelease>  \openin\@inputcheck#1 %
1336 <latexrelease>  \ifeof\@inputcheck%
1337 <latexrelease>    \@latex@warning@no@line%
1338 <latexrelease>      {Writing file '\@currdir#1'}%
1339 <latexrelease>  \chardef\reserved@c15 %
1340 <latexrelease>  \ch@ck7\reserved@c\write%
1341 <latexrelease>  \immediate\openout\reserved@c#1\relax%
1342 <latexrelease> \else%
1343 <latexrelease>  \closein\@inputcheck%
1344 <latexrelease>  \@latex@warning@no@line%
1345 <latexrelease>    {File '#1' already exists on the system.\MessageBreak%
1346 <latexrelease>      Not generating it from this source}%
1347 <latexrelease>  \let\write\@gobbletwo%
1348 <latexrelease>  \let\closeout\@gobble%
1349 <latexrelease> \fi%
1350 <latexrelease> \if@tempswa%
1351 <latexrelease>  \immediate\write\reserved@c{%
1352 <latexrelease>    \@percentchar\@percentchar\space%
1353 <latexrelease>    \expandafter\@gobble\string\LaTeX2e file '#1'^J%
1354 <latexrelease>    \@percentchar\@percentchar\space generated by the %
1355 <latexrelease>    '\@currenvir' \expandafter\@gobblefour\string\newenvironment^^J%
1356 <latexrelease>    \@percentchar\@percentchar\space from source '\jobname' on %
1357 <latexrelease>    \number\year/\two@digits\month/\two@digits\day.^J%
1358 <latexrelease>    \@percentchar\@percentchar}%
1359 <latexrelease> \fi%
1360 <latexrelease> \let\do\@makeother\dospecials%
1361 <latexrelease> \count@ 128\relax%
1362 <latexrelease> \loop%

```

```

1363 <latexrelease> \catcode\count@ 11\relax%
1364 <latexrelease> \advance\count@ \@ne%
1365 <latexrelease> \ifnum\count@<\@cclvi%
1366 <latexrelease> \repeat%
1367 <latexrelease> \edef\E{\@backslashchar end\string{\@currenvir\string}}%
1368 <latexrelease> \edef\reserved@b{%
1369 <latexrelease> \def\noexpand\reserved@b%
1370 <latexrelease> #####1\E####2\E####3\relax}%
1371 <latexrelease> \reserved@b{%
1372 <latexrelease> \ifx\relax##3\relax%
1373 <latexrelease> \immediate\write\reserved@c{##1}%
1374 <latexrelease> \else%
1375 <latexrelease> \edef^^M{\noexpand\end{\@currenvir}}%
1376 <latexrelease> \ifx\relax##1\relax%
1377 <latexrelease> \else%
1378 <latexrelease> \@latex@warning{Writing text ‘##1’ before %
1379 <latexrelease> \string\end{\@currenvir}\MessageBreak as last line of #1}%
1380 <latexrelease> \immediate\write\reserved@c{##1}%
1381 <latexrelease> \fi%
1382 <latexrelease> \ifx\relax##2\relax%
1383 <latexrelease> \else%
1384 <latexrelease> \@latex@warning{%
1385 <latexrelease> Ignoring text ‘##2’ after \string\end{\@currenvir}}%
1386 <latexrelease> \fi%
1387 <latexrelease> \fi%
1388 <latexrelease> ^^M}%
1389 <latexrelease>
1390 <latexrelease> \catcode‘^^L\active%
1391 <latexrelease> \let\L\@undefined%
1392 <latexrelease> \def^^L{\expandafter\ifx\csname L\endcsname\relax\fi ^^J^^J}%
1393 <latexrelease> \catcode‘^^I\active%
1394 <latexrelease> \let\I\@undefined%
1395 <latexrelease> \def^^I{\expandafter\ifx\csname I\endcsname\relax\fi\space}%
1396 <latexrelease> \catcode‘^^M\active%
1397 <latexrelease> \edef^^M##1^^M{%
1398 <latexrelease> \noexpand\reserved@b##1\E\E\relax}}%
1399 <latexrelease>\endgroup%
1400 <latexrelease>\EndIncludeInRelease
1401 (*2ekernel)

1402 \begingroup
1403 \catcode‘|= \catcode‘\%
1404 \catcode‘\%=12
1405 \catcode‘\*=11
1406 \gdef\@percentchar{%}
1407 \gdef\endfilecontents{|
1408 \immediate\closeout\reserved@c
1409 \def\T##1##2##3{|
1410 \ifx##1\@undefined\else
1411 \@latex@warning@no@line{##2 has been converted to Blank ##3e}|
1412 \fi}|
1413 \T\L{Form Feed}{Lin}|
1414 \T\I{Tab}{Spac}|
1415 \immediate\write\@unused{}}
1416 \global\let\endfilecontents*\endfilecontents

```


We no longer prevent the code to be used after begin document (no rollback needed for this change).

```

1417 %\onlypreamble\filecontents
1418 %\onlypreamble\endfilecontents
1419 %\onlypreamble\filecontents*
1420 %\onlypreamble\endfilecontents*
1421 \endgroup
1422 %\onlypreamble\filecontents
(End definition for \filecontents and \endfilecontents.)

```

5 Package/class rollback mechanism

```

1423 </2ekernel>
1424 <*2ekernel | latexreleasefirst>

```

`\pkgcls@debug` For testing we have a few extra lines of code that by default do nothing but one can set `\pkgcls@debug` to `\typeout` to get extra info. Sometime in the future this will be dropped.

```

1425 <*tracerollback>
1426 %\let\pkgcls@debug\typeout
1427 \let\pkgcls@debug@gobble
1428 </tracerollback>

```

(End definition for `\pkgcls@debug`.)

`\requestedLaTeXdate` The macro (!) `\requestedLaTeXdate` holds the globally requested rollback date (via `latexrelease`) or zero if no such request was made.

```

1429 \def\requestedLaTeXdate{0}

```

(End definition for `\requestedLaTeXdate`.)

`\pkgcls@targetdate` If a rollback for a package or class is requested then `\pkgcls@targetdate` holds the requested date as a number YYYYMMDD (if there was one, otherwise the value of `\requestedLaTeXdate`) and `\pkgcls@targetlabel` will be empty. If there was a request for a named version then `\pkgcls@targetlabel` holds the version name and `\pkgcls@targetdate` is set to 1.

`\pkgcls@targetdate=0` is used to indicate that there was no rollback request. While loading an old release `\pkgcls@targetdate` is also reset to zero so that `\DeclareRelease` declarations are bypassed.

In contrast `\pkgcls@innerdate` will always hold the requested date (in a macro not a counter) if there was one, otherwise, e.g., if there was no request or a request to a version name it will contain T_EX largest legal number. While loading a file this can be used to provide conditionals that select code based on the request.

```

1430 \ifx\pkgcls@targetdate\@undefined
1431   \newcount\pkgcls@targetdate
1432 \fi
1433 \let\pkgcls@targetlabel\@empty
1434 \def\pkgcls@innerdate{\maxdimen}

```

(End definition for `\pkgcls@targetdate`, `\pkgcls@targetlabel`, and `\pkgcls@innerdate`.)

`\pkgcls@candidate` When looping through the `\DeclareRelease` declarations we record if the release is the best candidate we have seen so far. This is recorded in `\pkgcls@candidate` and we update it whenever we see a better one.

In `\pkgcls@releasedate` we keep track of the release date of that candidate.

```
1435 \let\pkgcls@candidate\@empty
1436 \let\pkgcls@releasedate\@empty
```

(End definition for `\pkgcls@candidate` and `\pkgcls@releasedate`.)

`\load@onefilewithoptions` the best place to add the rollback code is at the point where `\@onefilewithoptions` is called to load a single class or package.

To make things easy we save the old definition as `\load@onefilewithoptions` and then provide a new interface.

Important: as this code is also unconditionally placed into `latexrelease` we can only do this name change once otherwise both macros will contain the same code.

```
1437 \ifx\load@onefilewithoptions\undefined
1438 \let\load@onefilewithoptions\@onefilewithoptions
1439 \def\@onefilewithoptions#1[#2][#3]#4{%
```

First a bit of tracing normally disabled.

```
1440 \tracingrequest
1441 \pkgcls@debug{--- File loaded request (\noexpand\usepackage or ...)}%
1442 \pkgcls@debug{\@spaces 1: #1}%
1443 \pkgcls@debug{\@spaces 2: #2}%
1444 \pkgcls@debug{\@spaces 3: #3}%
1445 \pkgcls@debug{\@spaces 4: #4}%
1446 \tracingrequest
```

Two of the arguments are needed later on in error/warning messages so we save them.

```
1447 \def\pkgcls@name{#1}% % for info message
1448 \def\pkgcls@arg {#3}% % for info message
```

then we parse the final optional argument to determine if there is a specific rollback request for the current file. This will set `\pkgcls@targetdate`, `\pkgcls@targetlabel` and `\pkgcls@mindate`.

```
1449 \pkgcls@parse@date@arg{#3}%
```

When determining the correct release to load we keep track of candidates in `\pkgcls@candidate` and initially we don't have any:

```
1450 \let\pkgcls@candidate\@empty
```

If we had a rollback request then `#3` may contain data but not necessarily a “minimal date” so instead of passing it on we pass on the content of `\pkgcls@mindate`. We need to pass the value not the command, otherwise nested packages may pick up the wrong information.

```
1451 \begingroup
1452 \edef\reserved@a{%
1453 \endgroup
1454 \unexpanded{\load@onefilewithoptions#1[#2]}%
1455 [\pkgcls@mindate]%
1456 \unexpanded{#4}}%
1457 \reserved@a
1458 }
1459 \fi
```

(End definition for `\load@onefilewithoptions` and `\@onefilewithoptions`.)

`\pkgcls@parse@date@arg` The `\pkgcls@parse@date@arg` command parses the second optional argument of `\usepackage`, `\RequirePackage` or `\documentclass` for a rollback request setting the values of `\pkgcls@targetdate` and `\pkgcls@targetlabel`.

This optional argument has a dual purpose: If it just contains a date string then this means that the package should have at least that date (to ensure that a certain feature is actually available, or a certain bug has been fixed). When the package gets loaded the information in `\Provides...` will then be checked against this request.

But if it starts with an equal sign followed by a date string or followed by a version name then this means that we should roll back to the state of the package at that date or to the version with the requested name.

If there was no optional argument or the optional argument does not start with “=” then the `\pkgcls@targetdate` is set to the date of the overall rollback request (via `latexrelease`) or if that was not given it is set to 0. In either case `\pkgcls@targetlabel` will be made empty.

If the argument doesn’t start with “=” then it is supposed to be a “minimal date” and we therefore save the value in `\pkgcls@mindate`, otherwise this macro is made empty.

So in summary we have:

Input	<code>\pkgcls@targetdate</code>	<code>\pkgcls@targetlabel</code>	<code>\pkgcls@mindate</code>
<code><empty></code>	<code><global-rollbackdate-as-number></code>	<code><empty></code>	<code><empty></code>
<code><date></code>	<code><global-rollbackdate-as-number></code>	<code><empty></code>	<code><date></code>
<code>=<date></code>	<code><date-as-number></code>	<code><empty></code>	<code><empty></code>
<code>=<version></code>	1	<code><version></code>	<code><empty></code>
<code><other></code>	<code><global-rollbackdate-as-number></code>	<code><empty></code>	<code><other></code>

where `<global-rollbackdate-as-number>` is a date request given via `latexrelease` or if there wasn’t one 0.

```
1460 \def\pkgcls@parse@date@arg #1{%
```

If the argument is empty we use the rollback date from `latexrelease` which has the value of zero if there was no rollback request. The label and the minimal date is made empty in that case.

```
1461 \ifx\@nil#1\@nil
1462 \pkgcls@targetdate\requestedLaTeXdate\relax
1463 \let\pkgcls@targetlabel\@empty
1464 \let\pkgcls@mindate\@empty
```

Otherwise we parse the argument further, checking for a = as the first character. We append a = at the end so that there is at least one such character in the argument.

```
1465 \else
1466 \pkgcls@parse@date@arg#1=\@nil\relax
1467 \fi
1468 }
```

The actual parsing work then happens in `\pkgcls@parse@date@arg@`:

```
1469 \def\pkgcls@parse@date@arg@#1=#2\@nil{%
```

We set `\pkgcls@targetdate` depending on the parsing result; the code is expandable so we can do the parsing as part of the assignment.

```
1470 \pkgcls@targetdate
```

If a = was in first position then #1 will be empty. In that case #2 will be the original argument with a = appended.

This can be parsed with \@parse@version, the trailing character is simply ignored. This macro returns the parsed date as a number (or zero if it wasn't a date) and accepts both YYYY/MM/DD and YYYY-MM-DD formats.

```
1471 \ifx\@nil#1\@nil
1472 \@parse@version0#2//00\@nil\relax
```

Whatever is returned is thus assigned to \pkgcls@targetdate and therefore we can now test its value. If the value is zero we assume that the remaining argument string represents a version and change \pkgcls@targetdate and set \pkgcls@targetlabel to the version name (after stripping off the trailing =.

```
1473 \ifnum \pkgcls@targetdate=\z@
1474 \pkgcls@targetdate\@ne
1475 \def\pkgcls@innerdate{\maxdimen}%
1476 \pkgcls@parse@date@arg@version#2%
1477 \else
1478 \edef\pkgcls@innerdate{\the\pkgcls@targetdate}%
1479 \fi
1480 \let\pkgcls@mindate\@empty
1481 \else
```

If #1 was not empty then there wasn't a = character in first position so we are dealing either with a "minimum date" or with some incorrect data. We assume the former and make the following assignments (the first one finishing the assignment of \pkgcls@targetdate):

```
1482 \requestedLaTeXdate\relax
1483 \let\pkgcls@targetlabel\@empty
1484 \def\pkgcls@innerdate{\maxdimen}%
1485 \def\pkgcls@mindate{#1}%
```

If the min-date is after the requested rollback date (if there is any, i.e., if it is not zero) then we have a conflict and therefore issue a warning.

```
1486 \ifnum \pkgcls@targetdate > \z@
1487 \ifnum \@parse@version0#1//00\@nil > \pkgcls@targetdate
1488 \@latex@warning@no@line{Suspicious rollback/min-date date given\MessageBreak
1489 A minimal date of #1 has been specified for
1490 \@cls@pkg\MessageBreak '\pkgcls@name'.\MessageBreak
1491 But this is in conflict
1492 with a rollback request to \requestedpatchdate}
1493 \fi
1494 \fi
1495 \fi
1496 }
```

Strip off the trailing = and assign the version name to \pkgcls@targetlabel.

```
1497 \def\pkgcls@parse@date@arg@version#1={%
1498 \def\pkgcls@targetlabel{#1}}
```

(End definition for \pkgcls@parse@date@arg.)

\DeclareRelease First argument is the "name" of the release and it can be left empty if one doesn't like to give a name to the release. The second argument is that from which on this release was available (or should be used in case of minor updates). The final argument is the external

file name of this release, by convention this should be $\langle pkg/cls-name \rangle - \langle date \rangle . \langle extension \rangle$ but this is not enforced and through this argument one can overwrite it.

```

1499 \def\DeclareRelease#1#2#3{%
1500   \ifnum\pkgcls@targetdate>\z@ % some sort of rollback request
1501   \tracerollback
1502   \pkgcls@debug{---\string\DeclareRelease:}%
1503   \pkgcls@debug{\@spaces 1: #1}%
1504   \pkgcls@debug{\@spaces 2: #2}%
1505   \pkgcls@debug{\@spaces 3: #3}%
1506 }

```

If the date argument #2 is empty we are dealing with a special release that should be only accessible via its name; a typical use case would be a “beta” release. So if we are currently processing a date request we ignore it and otherwise we check if we can match the name and if so load the corresponding release file.

```

1507   \ifx\@nil#2\@nil
1508     \ifnum\pkgcls@targetdate=\@ne % named request
1509     \def\reserved@a{#1}%
1510     \ifx\pkgcls@targetlabel\reserved@a
1511     \pkgcls@use@this@release{#3}{}%
1512   \tracerollback
1513   \else
1514     \pkgcls@debug{Label doesn't match}%
1515   \tracerollback
1516   \fi
1517 \tracerollback
1518 \else
1519   \pkgcls@debug{Date request: ignored}%
1520 \tracerollback
1521 \fi
1522 \else

```

If the value of $\backslash\text{pkgcls@targetdate}$ is greater than 1 (or in reality greater than something like 19930101) we are dealing with a rollback request to a specific date.

```

1523   \ifnum\pkgcls@targetdate>\@ne % a real request

```

So we parse the date of this release to check if it is before or after the request date.

```

1524   \ifnum\@parse@version#2//00\@nil
1525   >\pkgcls@targetdate

```

If it is after we have to distinguish between two cases: If there was an earlier candidate we use that one because the other is too late, but if there wasn’t one (i.e., if current release is the oldest that exists) we use it as the best choice. However in that case something is wrong (as there shouldn’t be a rollback to a date where a package used doesn’t yet exists. So we make a complained to the user.

```

1526   \ifx\pkgcls@candidate\@empty
1527   \pkgcls@rollbackdate@error{#2}%
1528   \pkgcls@use@this@release{#3}{#2}%
1529   \else
1530   \pkgcls@use@this@release\pkgcls@candidate
1531   \pkgcls@releasedate
1532   \fi
1533 \else

```

Otherwise, if the release date of this version is before the target rollback and we record it as a candidate. But we don't use it yet as there may be another release which is still before the target rollback.

```

1534         \def\pkgcls@candidate{#3}%
1535         \def\pkgcls@releasedate{#2}%
1536 (*tracereollback)
1537         \pkgcls@debug{New candidate: #3}%
1538 (/tracereollback)
1539         \fi
1540     \else

```

If we end up in this branch we have a named version request. So we check if `\pkgcls@targetlabel` matches the current name and if yes we use this release immediately, otherwise we do nothing as a later declaration may match it.

```

1541         \def\reserved@a{#1}%
1542         \ifx\pkgcls@targetlabel\reserved@a
1543         \pkgcls@use@this@release{#3}{#2}%
1544 (*tracereollback)
1545         \else
1546         \pkgcls@debug{Label doesn't match}%
1547 (/tracereollback)
1548         \fi
1549     \fi
1550 \fi
1551 \fi
1552 }

```

(End definition for `\DeclareRelease`.)

`\pkgcls@use@this@release` If a certain release has been selected (stored in the external file given in #1) we need to input it and afterwards stop reading the current file.

```

1553 \def\pkgcls@use@this@release#1#2{%

```

Before that we record the selection made inside the transcript.

```

1554     \pkgcls@show@selection{#1}{#2}%

```

We then set the `\pkgcls@targetdate` to zero so that any `\DeclareRelease` or `\DeclareCurrentRelease` in the file we now load are bypassed³¹ and then we finally load the correct release.

After loading that file we need to stop reading the current file so we issue `\endinput`. Note that the `\relax` before that is essential to ensure that the `\endinput` is only happening after the file has been fully processed, otherwise it would act after the first line of the `\@@input`!

```

1555     \pkgcls@targetdate\z@
1556     \@@input #1\relax
1557     \endinput
1558 }

```

(End definition for `\pkgcls@use@this@release`.)

³¹The older release may also have such declarations inside if it was a simply copy of the `.sty` or `.cls` file current at that date. Removing these declarations would make the file load a tiny bit faster, but this way it works in any case.

`\pkgcls@show@selection` This command records what selection was made. As that is needed in two places (and it is rather lengthy) it was placed in a separate command. The first argument is the name of the external file that is being loaded and is only needed for debugging. The second argument is the date that corresponds to this file and it is used as part of the message.

```

1559 \def\pkgcls@show@selection#1#2{%
1560   (*traceroollback)
1561   \pkgcls@debug{Result: use #1}%
1562 }/traceroollback)
1563 \GenericInfo
1564 {\@spaces\@spaces\space}{Rollback for
1565   \@cls@pkg\space'\@currname' requested ->
1566   \ifnum\pkgcls@targetdate>\@ne
1567     date
1568     \ifnum\requestedLaTeXdate=\pkgcls@targetdate
1569       \requestedpatchdate
1570     \else
1571       \expandafter\@gobble\pkgcls@arg
1572     \fi.\MessageBreak

```

Instead of “best approximation” we could say that we have been able to exactly match the date (if it is exact), but that would mean extra tests without much gain, so not done.

```

1573   Best approximation is
1574   \else
1575     version '\pkgcls@targetlabel'.\MessageBreak
1576     This corresponds to
1577   \fi
1578   \ifx\@nil#2\@nil
1579     a special release%
1580   \else
1581     the release introduced on #2%
1582   \fi
1583   \@gobble}%
1584 }

```

(End definition for `\pkgcls@show@selection`.)

`\pkgcls@rollbackdate@error` This is called if the requested rollback date is earlier than the earliest known release of a package or class.

A similar error is given if global rollback date and min-date on a specific package conflict with each other, but that case is happens only once so it is inlined.

```

1585 \def\pkgcls@rollbackdate@error#1{%
1586   \@latex@error{Suspicious rollback date given}%
1587   {The \@cls@pkg\space'\@currname' has no rollback data
1588     before #1 which\MessageBreak
1589     is after your requested rollback date --- so
1590     something may be wrong here.\MessageBreak
1591     Continue and we use the earliest known release.}}

```

(End definition for `\pkgcls@rollbackdate@error`.)

`\DeclareCurrentRelease` This declares the date (and possible name) of the current version of a package or class.

```

1592 \def\DeclareCurrentRelease#1#2{%

```

First we test if `\pkgcls@targetdate` is greater than zero, otherwise this code is bypassed (as there is no rollback request).

```

1593 \ifnum\pkgcls@targetdate>\z@ % some sort of rollback request
1594 (*traceroollback)
1595 \pkgcls@debug{---DeclareCurrentRelease}%
1596 \pkgcls@debug{ 1: #1}%
1597 \pkgcls@debug{ 2: #2}%
1598 (/traceroollback)

```

If the value is greater than 1 we have to deal with a date request, so we parse #2 as a date and compare it with `\pkgcls@targetdate`.

```

1599 \ifnum\pkgcls@targetdate>\@ne % a date request
1600 \ifnum\@parse@version#2//00\@nil
1601 >\pkgcls@targetdate

```

If it is greater that means the release date if this file is later than the requested rollback date. Again we have two cases: If there was a previous candidate release we use that one as the current release is too young, but if there wasn't we have to use this release nevertheless as there isn't any alternative.

However this case can only happen if there is a `\DeclareCurrentRelease` but no declared older releases (so basically the use of the declaration is a bit dubious).

```

1602 \ifx\pkgcls@candidate\@empty
1603 \pkgcls@rollbackdate@error{#2}%
1604 \else
1605 \pkgcls@use@this@release\pkgcls@candidate
1606 \pkgcls@releasedate
1607 \fi

```

Otherwise the current file is the right release, so we record that in the transcript and then carry on.

```

1608 \else
1609 \pkgcls@show@selection{current version}{#2}%
1610 \fi
1611 \else % a label request

```

Otherwise we have a rollback request to a named version so we check if that fits the current name and if not give an error as this was the last possible opportunity.

```

1612 \def\reserved@a{#1}%
1613 \ifx\pkgcls@targetlabel\reserved@a
1614 \pkgcls@show@selection{current version}{#2}%
1615 \else
1616 \@latex@error{Requested version '\pkgcls@targetlabel' for
1617 \cls@pkg\space'\@currname' is unknown}\@ehc
1618 \fi
1619 \fi
1620 \fi
1621 }

```

(End definition for `\DeclareCurrentRelease`.)

\IfTargetDateBefore This enables a simple form of conditional code inside a class or package file. If there is a date request and the request date is earlier than the first argument the code in the second argument is processed otherwise the code in the third argument is processed. If there was no date request then we also execute the third argument, i.e., we will get the “latest” version of the file.

Most often the second argument (before-date-code) will be empty.

```

1622 \DeclareRobustCommand\IfTargetDateBefore[1]{%
1623   \ifnum\pkgcls@innerdate <%
1624     \expandafter\@parse@version\expandafter0#1//00\@nil
1625     \typeout{Exclude code introduced on #1}%
1626     \expandafter\@firstoftwo
1627   \else
1628     \typeout{Include code introduced on #1}%
1629     \expandafter\@secondoftwo
1630   \fi
1631 }

```

(End definition for \IfTargetDateBefore.)

```

1632 </2ekernel | latexreleasefirst>

```

6 After Preamble

Finally we declare a package that allows all the commands declared above to be `\@onlypreamble` to be used after `\begin{document}`.

```

1633 <*afterpreamble>
1634 \NeedsTeXFormat{LaTeX2e}
1635 \ProvidesPackage{pkgindoc}
1636   [2020-08-08 v1.3m Package Interface in Document (DPC)]
1637 \def\reserved@a#1\do\@classoptionslist#2\do\filecontents#3\relax{%
1638   \gdef\@preamblecmds{#1#3}}
1639 \expandafter\reserved@a\@preamblecmds\relax
1640 </afterpreamble>

```

File T

ltfilehook.dtx

Contents

1 Introduction

1.1 Provided hooks

The code offers a number of hooks into which packages (or the user) can add code to support different use cases. Many hooks are offered as pairs (i.e., the second hook is reversed). Also important to know is that these pairs are properly nested with respect to other pairs of hooks.

There are hooks that are executed for all files of a certain type (if they contain code), e.g., for all “include files” or all “packages”, and there are also hooks that are specific to a single file, e.g., do something after the package `foo.sty` has been loaded.

1.2 General hooks for file reading

There are four hooks that are called for each file that is read using document-level commands such as `\input`, `\include`, `\usepackage`, etc. They are not called for files read using internal low-level methods, such as `\@input` or `\openin`.

<code>file/before</code>
<code>file/before/...</code>
<code>file/after/...</code>
<code>file/after</code>

These are:

file/before, **file/before/⟨file-name⟩** These hooks are executed in that order just before the file is loaded for reading. The code of the first hook is used with every file, while the second is executed only for the file with matching `⟨file-name⟩` allowing you to specify code that only applies to one file.

file/after/⟨file-name⟩, **file/after** These hooks are after the file with name `⟨file-name⟩` has been fully consumed. The order is swapped (the specific one comes first) so that the **before** and **after** hooks nest properly, which is important if any of them involve grouping (e.g., contain environments, for example). Furthermore both hooks are reversed hooks to support correct nesting of different packages adding code to both `/before` and `/after` hooks.

So the overall sequence of hook processing for any file read through the user interface commands of L^AT_EX is:

```
\UseHook{⟨file/before⟩}
\UseHook{⟨file/before/⟨file name⟩⟩}
  ⟨file contents⟩
\UseHook{⟨file/after/⟨file name⟩⟩}
\UseHook{⟨file/after⟩}
```

The file hooks only refer to the file by its name and extension, so the $\langle file\ name \rangle$ should be the file name as it is on the filesystem with extension (if any) and without paths. Different from `\input` and similar commands, the `.tex` extension is not assumed in hook $\langle file\ name \rangle$, so `.tex` files must be specified with their extension to be recognized. Files within subfolders should also be addressed by their name and extension only.

Extensionless files also work, and should then be given without extension. Note however that \TeX prioritizes `.tex` files, so if two files `foo` and `foo.tex` exist in the search path, only the latter will be seen.

When a file is input, the $\langle file\ name \rangle$ is available in `\CurrentFile`, which is then used when accessing the `file/before/` $\langle file\ name \rangle$ and `file/after/` $\langle file\ name \rangle.$

`\CurrentFile`

The name of the file about to be read (or just finished) is available to the hooks through `\CurrentFile` (there is no `expl3` name for it for now). The file is always provided with its extension, i.e., how it appears on your hard drive, but without any specified path to it. For example, `\input{sample}` and `\input{app/sample.tex}` would both have `\CurrentFile` being `sample.tex`.

`\CurrentFilePath`

The path to the current file (complement to `\CurrentFile`) is available in `\CurrentFilePath` if needed. The paths returned in `\CurrentFilePath` are only user paths, given through `\input@path` (or `expl3`'s equivalent `\l_file_search_path_seq`) or by directly typing in the path in the `\input` command or equivalent. Files located by `kpsewhich` get the path added internally by the \TeX implementation, so at the macro level it looks as if the file were in the current folder, so the path in `\CurrentFilePath` is empty in these cases (package and class files, mostly).

`\CurrentFileUsed`
`\CurrentFilePathUsed`

In normal circumstances these are identical to `\CurrentFile` and `\CurrentFilePath`. They will differ when a file substitution has occurred for `\CurrentFile`. In that case, `\CurrentFileUsed` and `\CurrentFilePathUsed` will hold the actual file name and path loaded by \LaTeX , while `\CurrentFile` and `\CurrentFilePath` will hold the names that were *asked for*. Unless doing very specific work on the file being read, `\CurrentFile` and `\CurrentFilePath` should be enough.

1.3 Hooks for package and class files

Commands to load package and class files (e.g., `\usepackage`, `\RequirePackage`, `\LoadPackageWithOptions`, etc.) offer the hooks from section 1.2 when they are used to load a package or class file, e.g., `file/after/array.sty` would be called after the `array` package got loaded. But as packages and classes form as special group of files, there are some additional hooks available that only apply when a package or class is loaded.

<hr/> package/before	These are:
package/after	
package/before/...	package/before, package/after These hooks are called for each package being loaded.
package/after/...	
class/before	package/before/⟨name⟩, package/after/⟨name⟩ These hooks are additionally called if
class/after	the package name is ⟨name⟩ (without extension).
class/before/...	
class/after/...	class/before, class/after These hooks are called for each class being loaded.
<hr/>	
	class/before/⟨name⟩, class/after/⟨name⟩ These hooks are additionally called if the
	class name is ⟨name⟩ (without extension).

All `/after` hooks are implemented as reversed hooks.
The overall sequence of execution for `\usepackage` and friends is therefore:

```

\UseHook{⟨package/before⟩}
\UseHook{⟨package/before/⟨package name⟩⟩}

  \UseHook{⟨file/before⟩}
  \UseHook{⟨file/before/⟨package name⟩.sty⟩}
    ⟨package contents⟩
  \UseHook{⟨file/after/⟨package name⟩.sty⟩}
  \UseHook{⟨file/after⟩}

  code from \AtEndOfPackage if used inside the package

\UseHook{⟨package/after/⟨package name⟩⟩}
\UseHook{⟨package/after⟩}

```

and similar for class file loading, except that `package/` is replaced by `class/` and `\AtEndOfPackage` by `\AtEndOfClass`.

If a package or class is not loaded (or it was loaded before the hooks were set) none of the hooks are executed!

1.4 Hooks for `\include` files

To manage `\include` files, L^AT_EX issues a `\clearpage` before and after loading such a file. Depending on the use case one may want to execute code before or after these `\clearpages` especially for the one that is issued at the end.

Executing code before the final `\clearpage`, means that the code is processed while the last page of the included material is still under construction. Executing code after it means that all floats from inside the include file are placed (which might have added further pages) and the final page has finished.

Because of these different scenarios we offer hooks in three places.³² None of the hooks are executed when an `\include` file is bypassed because of an `\includeonly` declaration. They are, however, all executed if L^AT_EX makes an attempt to load the `\include` file (even if it doesn't exist and all that happens is “No file `⟨filename⟩.tex`”).

³²If you want to execute code before the first `\clearpage` there is no need to use a hook—you can write it directly in front of the `\include`.

```

include/before
include/before/...
include/end
include/end/...
include/after
include/after/...

```

These are:

include/before, include/before/⟨name⟩ These hooks are executed (in that order) after the initial `\clearpage` and after `.aux` file is changed to use `⟨name⟩.aux`, but before the `⟨name⟩.tex` file is loaded. In other words they are executed at the very beginning of the first page of the `\include` file.

include/end/⟨name⟩, include/end These hooks are executed (in that order) after `ℒTEX` has stopped reading from the `\include` file, but before it has issued a `\clearpage` to output any deferred floats.

include/after/⟨name⟩, include/after These hooks are executed (in that order) after `ℒTEX` has issued the `\clearpage` but before it has switched back writing to the main `.aux` file. Thus technically we are still inside the `\include` and if the hooks generate any further typeset material including anything that writes to the `.aux` file, then it would be considered part of the included material and bypassed if it is not loaded because of some `\includeonly` statement.³³

1.5 High-level interfaces for `ℒTEX`

We do not provide any additional wrappers around the hooks (like `filehook` or `scrfile` do) because we believe that for package writers the high-level commands from the hook management, e.g., `\AddToHook`, etc. are sufficient and in fact easier to work with, given that the hooks have consistent naming conventions.

1.6 Internal interfaces for `ℒTEX`

```

\declare@file@substitution    \declare@file@substitution    {\file}\{replacement-file\}
\undecclare@file@substitution \undecclare@file@substitution {\file}

```

If `⟨file⟩` is requested for loading replace it with `⟨replacement-file⟩`. `\CurrentFile` remains pointing to `⟨file⟩` but `\CurrentFileUsed` will show the file actually loaded.

The main use case for this declaration is to provide a corrected version of a package that can't be changed (due to its license) but no longer functions because of `ℒTEX` kernel changes, for example, or to provide a version that makes use of new kernel functionality while the original package remains available for use with older releases.

The `\undecclare@file@substitution` declaration undoes a substitution made earlier.

Please do not misuse this functionality and replace a file with another unless if really needed and only if the new version is implementing the same functionality as the original one!

³³For that reason another `\clearpage` is executed after these hooks which normally does nothing, but starts a new page if further material got added this way.

```
\disable@package@load {\package}} {\alternate-code}}
\reenable@package@load {\package}}
```

If $\langle package \rangle$ is requested do not load it but instead run $\langle alternate-code \rangle$ which could issue a warning, error or any other code.

The main use case is for classes that want to restrict the set of supported packages or contain code that make the use of some packages impossible. So rather than waiting until the document breaks they can set up informative messages why certain packages are not available.

The function is only implemented for packages not for arbitrary files.

1.7 A sample package for structuring the log output

As an application we provide the package `structuredlog` that adds lines to the `.log` when a file is opened and closed for reading keeping track of nesting level as well. For example, for the current document it adds the lines

```
= (LEVEL 1 START) t1lmr.fd
= (LEVEL 1 STOP) t1lmr.fd
= (LEVEL 1 START) supp-pdf.mkii
= (LEVEL 1 STOP) supp-pdf.mkii
= (LEVEL 1 START) nameref.sty
== (LEVEL 2 START) refcount.sty
== (LEVEL 2 STOP) refcount.sty
== (LEVEL 2 START) gettitlestring.sty
== (LEVEL 2 STOP) gettitlestring.sty
= (LEVEL 1 STOP) nameref.sty
= (LEVEL 1 START) ltfilehook-doc.out
= (LEVEL 1 STOP) ltfilehook-doc.out
= (LEVEL 1 START) ltfilehook-doc.out
= (LEVEL 1 STOP) ltfilehook-doc.out
= (LEVEL 1 START) ltfilehook-doc.hd
= (LEVEL 1 STOP) ltfilehook-doc.hd
= (LEVEL 1 START) ltfilehook.dtx
== (LEVEL 2 START) ot1lmr.fd
== (LEVEL 2 STOP) ot1lmr.fd
== (LEVEL 2 START) omllmm.fd
== (LEVEL 2 STOP) omllmm.fd
== (LEVEL 2 START) omslmsy.fd
== (LEVEL 2 STOP) omslmsy.fd
== (LEVEL 2 START) omxlmex.fd
== (LEVEL 2 STOP) omxlmex.fd
== (LEVEL 2 START) umsa.fd
== (LEVEL 2 STOP) umsa.fd
== (LEVEL 2 START) umsb.fd
== (LEVEL 2 STOP) umsb.fd
== (LEVEL 2 START) ts1lmr.fd
== (LEVEL 2 STOP) ts1lmr.fd
== (LEVEL 2 START) t1lmss.fd
== (LEVEL 2 STOP) t1lmss.fd
= (LEVEL 1 STOP) ltfilehook.dtx
```

Thus if you inspect an issue in the .log it is easy to figure out in which file it occurred, simply by searching back for LEVEL and if it is a STOP then remove 1 from the level value and search further for LEVEL with that value which should then be the START level of the file you are in.

2 The Implementation

```
1 <*2ekernel>
2 <@@=filehook>
```

2.1 Document and package-level commands

\CurrentFile User-level macros that hold the current file name and file path. These are used internally as well because the code takes care to protect against a possible redefinition of these macros in the loaded file (it's necessary anyway to make hooks work with nested \input). The versions \...Used hold the *actual* file name and path that is loaded by L^AT_EX, whereas the other two hold the name as requested. They will differ in case there's a file substitution.

```
3 </2ekernel>
4 <*2ekernel | latexrelease>
5 <latexrelease>\IncludeInRelease{2020/10/01}%
6 <latexrelease>                {\CurrentFile}{Hook management file}%
7 \ExplSyntaxOn
8 \tl_new:N \CurrentFile
9 \tl_new:N \CurrentFilePath
10 \tl_new:N \CurrentFileUsed
11 \tl_new:N \CurrentFilePathUsed
12 \ExplSyntaxOff
13 </2ekernel | latexrelease>
14 <latexrelease>\EndIncludeInRelease
15 <latexrelease>\IncludeInRelease{0000/00/00}%
16 <latexrelease>                {\CurrentFile}{Hook management file}%
17 <latexrelease>
18 <latexrelease>\let \CurrentFile      \@undefined
19 <latexrelease>\let \CurrentFilePath  \@undefined
20 <latexrelease>\let \CurrentFileUsed  \@undefined
21 <latexrelease>\let \CurrentFilePathUsed \@undefined
22 <latexrelease>
23 <latexrelease>\EndIncludeInRelease
24 <*2ekernel>
```

(End definition for \CurrentFile and others. These functions are documented on page 804.)

2.2 expl3 helpers

```
25 </2ekernel>
26 <*2ekernel | latexrelease>
27 <latexrelease>\IncludeInRelease{2020/10/01}%
28 <latexrelease>                {\_filehook_file_parse_full_name:nN}{File helpers}%
29 \ExplSyntaxOn
```

_filehook_file_parse_full_name:nN A utility macro to trigger expl3's file-parsing and lookup, and return a normalized representation of the file name. If the queried file doesn't exist, no normalization takes place.

_filehook_full_name:nn

The output of `__filehook_file_parse_full_name:nN` is passed on to the #2—a 3-argument macro that takes the *<path>*, *<base>*, and *<ext>* parts of the file name.

```

30 \cs_new:Npn \__filehook_file_parse_full_name:nN #1
31 {
32   \exp_args:Nf \file_parse_full_name_apply:nN
33   {
34     \exp_args:Nf \__filehook_full_name:nn
35     { \file_full_name:n {#1} } {#1}
36   }
37 }
38 \cs_new:Npn \__filehook_full_name:nn #1 #2
39 {
40   \tl_if_empty:nTF {#1}
41   { \tl_trim_spaces:n {#2} }
42   { \tl_trim_spaces:n {#1} }
43 }

```

(End definition for `__filehook_file_parse_full_name:nN` and `__filehook_full_name:nn`.)

Some actions depend on whether the file extension was explicitly given, and sometimes the extension has to be removed. The macros below use `__filehook_file_parse_full_name:nN` to split up the file name and either check if *<ext>* (#3) is empty, or discard it.

```

44 \cs_new:Npn \__filehook_if_no_extension:nTF #1
45 {
46   \exp_args:Ne \tl_if_empty:nTF
47   { \file_parse_full_name_apply:nN {#1} \use_iii:nnn }
48 }
49 \cs_new_protected:Npn \__filehook_drop_extension:N #1
50 {
51   \tl_gset:Nx #1
52   {
53     \exp_args:NV \__filehook_file_parse_full_name:nN #1
54     \__filehook_drop_extension_aux:nnn
55   }
56 }
57 \cs_new:Npn \__filehook_drop_extension_aux:nnn #1 #2 #3
58 { \tl_if_empty:nF {#1} { #1 / } #2 }

```

(End definition for `__filehook_if_no_extension:nTF` and `__filehook_drop_extension:N`.)

Yet another stack, to keep track of `\CurrentFile` and `\CurrentFilePath` with nested `\inputs`. At the beginning of `\InputIfFileExists`, the current value of `\CurrentFilePath` and `\CurrentFile` is pushed to `\g__filehook_input_file_seq`, and at the end, it is popped and the value reassigned. Some other places don't use `\InputIfFileExists` directly (`\include`) or need `\CurrentFile` earlier (`\@onefilewithoptions`), so these are manually used elsewhere as well.

```

59 \tl_new:N \l__filehook_internal_tl
60 \seq_if_exist:NF \g__filehook_input_file_seq
61 { \seq_new:N \g__filehook_input_file_seq }
62 \cs_new_protected:Npn \__filehook_file_push:
63 {
64   \seq_gpush:Nx \g__filehook_input_file_seq

```



```

65     {
66         { \CurrentFilePathUsed } { \CurrentFileUsed }
67         { \CurrentFilePath      } { \CurrentFile      }
68     }
69 }
70 \cs_new_protected:Npn \__filehook_file_pop:
71 {
72     \seq_gpop:NNTF \g__filehook_input_file_seq \l__filehook_internal_tl
73     { \exp_after:wN \__filehook_file_pop_assign:nnnn \l__filehook_internal_tl }
74     {
75         \msg_error:nnn { kernel } { should-not-happen }
76         { Tried-to-pop-from-an-empty-file-name-stack. }
77     }
78 }
79 \cs_new_protected:Npn \__filehook_file_pop_assign:nnnn #1 #2 #3 #4
80 {
81     \tl_set:Nn \CurrentFilePathUsed {#1}
82     \tl_set:Nn \CurrentFileUsed     {#2}
83     \tl_set:Nn \CurrentFilePath     {#3}
84     \tl_set:Nn \CurrentFile         {#4}
85 }
86 \ExplSyntaxOff

```

(End definition for `\g__filehook_input_file_seq` and others.)

```

87 </2ekernel|latexrelease>
88 <latexrelease>\EndIncludeInRelease

```

When rolling forward the following `expl3` functions may not be defined. If we roll back the code does nothing.

```

89 <latexrelease>\IncludeInRelease{2020/10/01}%
90 <latexrelease>          {\file_parse_full_name_apply:nN}{Roll forward help}%
91 <latexrelease>
92 <latexrelease>\ExplSyntaxOn
93 <latexrelease>\cs_if_exist:NF\file_parse_full_name_apply:nN
94 <latexrelease>{
95 <latexrelease>\cs_new:Npn \file_parse_full_name_apply:nN #1
96 <latexrelease> {
97 <latexrelease>     \exp_args:Ne \__file_parse_full_name_auxi:nN
98 <latexrelease>     { \__kernel_file_name_sanitiz:n {#1} }
99 <latexrelease> }
100 <latexrelease>\cs_new:Npn \__file_parse_full_name_auxi:nN #1
101 <latexrelease> {
102 <latexrelease>     \__file_parse_full_name_area:nw { } #1
103 <latexrelease>     / \s__file_stop
104 <latexrelease> }
105 <latexrelease>\cs_new:Npn \__file_parse_full_name_area:nw #1 #2 / #3 \s__file_stop
106 <latexrelease> {
107 <latexrelease>     \tl_if_empty:nTF {#3}
108 <latexrelease>     { \__file_parse_full_name_base:nw { } #2 . \s__file_stop {#1} }
109 <latexrelease>     { \__file_parse_full_name_area:nw { #1 / #2 }
110 <latexrelease>         #3 \s__file_stop }
111 <latexrelease> }
112 <latexrelease>\cs_new:Npn \__file_parse_full_name_base:nw #1 #2 . #3 \s__file_stop
113 <latexrelease> {

```

```

114 <latexrelease> \tl_if_empty:nTF {#3}
115 <latexrelease> {
116 <latexrelease> \tl_if_empty:nTF {#1}
117 <latexrelease> {
118 <latexrelease> \tl_if_empty:nTF {#2}
119 <latexrelease> { \__file_parse_full_name_tidy:nnnN { } { } }
120 <latexrelease> { \__file_parse_full_name_tidy:nnnN { .#2 } { } }
121 <latexrelease> }
122 <latexrelease> { \__file_parse_full_name_tidy:nnnN {#1} { .#2 } }
123 <latexrelease> }
124 <latexrelease> { \__file_parse_full_name_base:nw { #1 . #2 }
125 <latexrelease> #3 \s__file_stop }
126 <latexrelease> }
127 <latexrelease>\cs_new:Npn \__file_parse_full_name_tidy:nnnN #1 #2 #3 #4
128 <latexrelease> {
129 <latexrelease> \exp_args:Nee #4
130 <latexrelease> {
131 <latexrelease> \str_if_eq:nnF {#3} { / } { \use_none:n }
132 <latexrelease> #3 \prg_do_nothing:
133 <latexrelease> }
134 <latexrelease> { \use_none:n #1 \prg_do_nothing: }
135 <latexrelease> {#2}
136 <latexrelease> }
137 <latexrelease>}
138 <latexrelease>\ExplSyntaxOff
139 <latexrelease>
140 <latexrelease>\EndIncludeInRelease
141 <*2ekernel>
142 <@@=>

```

2.3 Declaring the file-related hooks

All hooks starting with `file/`, `include/`, `class/` or `package/` are generic and will be allocated if code is added to them. Thus there is no need to explicitly declare any hook in the code below.

Furthermore, those named `.../after` or `.../end` are automatically declared as reversed hooks if filled with code, so this is also automatically taken care of.

2.4 Patching L^AT_EX's `\InputIfFileExists` command

Most of what we have to do is adding `\UseHook` into several L^AT_EX 2_ε core commands, because of some circular dependencies in the kernel we do this only now and not in `ltxfiles`.

```

\InputIfFileExists \InputIfFileExists loads any file if it is available so we have to add the hooks
\input@file@exists@with@hooks file/before and file/after in the right places. If the file doesn't exist no hooks
\unqu@tefilef@und should be executed.
143 </2ekernel>
144 <latexrelease>\IncludeInRelease{2020/10/01}%
145 <latexrelease> {\InputIfFileExists}{Hook management (files)}%
146 <*2ekernel | latexrelease>

```

```

147 \let\InputIfFileExists\@undefined
148 \DeclareRobustCommand \InputIfFileExists[2]{%
149   \IfFileExists{#1}%
150   {%
151     \@expl@@@filehook@file@push@@
152     \@filehook@set@CurrentFile

```

We pre-expand `\@filef@und` so that in case another file is loaded in the true branch of `\InputIfFileExists`, these don't change their value meanwhile. This isn't a worry with `\CurrentFile...` because they are kept in a stack.

```

153   \expandafter\@swaptwoargs\expandafter
154   {\expandafter\@input@file@exists@with@hooks
155   \expandafter{\@filef@und}}%
156   {#2}%
157   \@expl@@@filehook@file@pop@@
158 }%
159 }
160 \def\@input@file@exists@with@hooks#1{%

```

If the file exists then `\CurrentFile` holds its name. But we can't rely on that still being true after the file has been processed. Thus for using the name in the file hooks we need to preserve the name and then restore it for the `file/after/...` hook.

The hook always refers to the file requested by the user. The hook is *always* loaded for `\CurrentFile` which usually is the same as `\CurrentFileUsed`. In the case of a file replacement, the `\CurrentFileUsed` holds the actual file loaded. In any case the file names are normalized so that the hooks work on the real file name, rather than what the user typed in.

`expl3's \file_full_name:n` normalizes the file name (to factor out differences in the `.tex` extension), and then does a file lookup to take into account a possible path from `\l_file_search_path_seq` and `\input@path`. However only the file name and extension are returned so that file hooks can refer to the file by their name only. The path to the file is returned in `\CurrentFilePath`.

```

161 \edef\reserved@a{%
162   \@expl@@@filehook@file@pop@assign@@nnnn
163   {\CurrentFilePathUsed}%
164   {\CurrentFileUsed}%
165   {\CurrentFilePath}%
166   {\CurrentFile}}%
167 \expandafter\@swaptwoargs\expandafter{\reserved@a}%

```

Before adding to the file list we need to make all (letter) characters catcode 11, because several packages use constructions like

```

\filename@parse{<filename>}
\ifx\filename@ext\@clsextension
...
\fi

```

and that doesn't work if `\filename@ext` is `\detokenized`. Making `\@clsextension` a string doesn't help much because some packages define their own `\<prefix>@someextension` with normal catcodes. This is not entirely correct because packages loaded (somehow) with catcode 12 alphabetic tokens (say, as the result of a `\string` or `\detokenize` command, or from a T_EX string like `\jobname`) will have these character tokens incorrectly

turned into letter tokens. This however is rare, so we'll go for the all-letters approach (grepping the packages in T_EX Live didn't bring up any obvious candidate for breaking with this catcode change).

```

168 {\edef\reserved@a{\unqu@tefilef@und#1\@nil}%
169 \addtofilelist{\string@makeletter\reserved@a}%
170 \UseHook{file/before}%

```

The current file name is available in `\CurrentFile` so we use that in the specific hook.

```

171 \UseHook{file/before/\CurrentFile}%
172 \@@input #1% <- trailing space comes from \@filef@und
173 }%

```

And here, `\CurrentFile` is restored (by `\@expl@@@filehook@file@pop@assign@@nnnn`) so we can use it once more.

```

174 \UseHook{file/after/\CurrentFile}%
175 \UseHook{file/after}}
176 \def\unqu@tefilef@und"#1" \@nil{#1}
177 \<latexrelease>\EndIncludeInRelease
178 \</2ekernel | latexrelease>

```

Now define `\InputIfFileExists` to input #1 if it seems to exist. Immediately prior to the input, #2 is executed. If the file #1 does not exist, execute ‘#3’.

```

179 \<latexrelease>\IncludeInRelease{2019/10/01}%
180 \<latexrelease> \InputIfFileExists}{Hook management (files)}%
181 \<latexrelease>
182 \<latexrelease>\DeclareRobustCommand \InputIfFileExists[2]{%
183 \<latexrelease> \IfFileExists{#1}%
184 \<latexrelease> {%
185 \<latexrelease> \expandafter\@swaptwoargs\expandafter
186 \<latexrelease> {\@filef@und}{#2\@addtofilelist{#1}\@@input}}%
187 \<latexrelease>\let\@input@file@exists@with@hooks\@undefined
188 \<latexrelease>\let\unqu@tefilef@und\@undefined
189 \<latexrelease>\EndIncludeInRelease
190 \<latexrelease>\IncludeInRelease{0000/00/00}%
191 \<latexrelease> \InputIfFileExists}{Hook management (files)}%
192 \<latexrelease>\long\def \InputIfFileExists#1#2{%
193 \<latexrelease> \IfFileExists{#1}%
194 \<latexrelease> {#2\@addtofilelist{#1}\@@input \@filef@und}}

```

Also undo the internal command as some packages unfortunately test for their existence instead of using `\IfFormatAtLeastTF`.

```

195 \<latexrelease>\expandafter\let\csname InputIfFileExists \endcsname\@undefined
196 \<latexrelease>\let\@input@file@exists@with@hooks\@undefined
197 \<latexrelease>\let\unqu@tefilef@und\@undefined
198 \<latexrelease>\EndIncludeInRelease
199 \<*/2ekernel>

```

(End definition for `\InputIfFileExists`, `\@input@file@exists@with@hooks`, and `\unqu@tefilef@und`.)

2.5 Declaring a file substitution

```

200 <@=filehook>
201 </2ekernel>
202 <*2ekernel | latexrelease>
203 <latexrelease>\IncludeInRelease{2020/10/01}%
204 <latexrelease>          {\_filehook_subst_add:nn}{Declaring file substitution}%
205 \ExplSyntaxOn

\_filehook_subst_add:nn \_filehook_subst_add:nn declares a file substitution by doing a (global) definition
\_filehook_subst_remove:n of the form \def\@file-subst@<file>{\<replacement>}. The file names are properly
\_filehook_subst_file_normalize:Nn sanitised, and normalized with the same treatment done for the file hooks. That is, a
\_filehook_subst_empty_name_chk:NN file replacement is declared by using the file name (and extension, if any) only, and the
file path should not be given. If a file name is empty it is replaced by .tex (the empty
cname is used to check that).

206 \cs_new_protected:Npn \_filehook_subst_add:nn #1 #2
207 {
208   \group_begin:
209     \cs_set:cpx { } { \exp_not:o { \cs:w\cs_end: } }
210     \int_set:Nn \tex_escapechar:D { -1 }
211     \cs_gset:cpx
212       {
213         \@file-subst@
214         \_filehook_subst_file_normalize:Nn \use_ii_iii:nnn {#1}
215       }
216     { \_filehook_subst_file_normalize:Nn \_filehook_file_name_compose:nnn
217       {#2} }
218   \group_end:
219 }
220 \cs_new_protected:Npn \_filehook_subst_remove:n #1
221 {
222   \group_begin:
223     \cs_set:cpx { } { \exp_not:o { \cs:w\cs_end: } }
224     \int_set:Nn \tex_escapechar:D { -1 }
225     \cs_undefine:c
226       {
227         \@file-subst@
228         \_filehook_subst_file_normalize:Nn \use_ii_iii:nnn {#1}
229       }
230   \group_end:
231 }
232 \cs_new:Npn \_filehook_subst_file_normalize:Nn #1 #2
233 {
234   \exp_after:wN \_filehook_subst_empty_name_chk:NN
235   \cs:w \exp_after:wN \cs_end:
236   \cs:w \_filehook_file_parse_full_name:nN {#2} #1 \cs_end:
237 }
238 \cs_new:Npn \_filehook_subst_empty_name_chk:NN #1 #2
239 { \if_meaning:w #1 #2 .tex \else: \token_to_str:N #2 \fi: }

(End definition for \_filehook_subst_add:nn and others.)

```

\use_ii_iii:nnn A variant of \use_... to discard the first of three arguments.

Todo: this should move to expl3

```
240 \cs_gset:Npn \use_ii_iii:nnn #1 #2 #3 {#2 #3}
```

(End definition for \use_ii_iii:nnn.)

```
241 \ExplSyntaxOff
242 </2ekernel | latexrelease>
243 <latexrelease>\EndIncludeInRelease
244 <*2ekernel>
```

\declare@file@substitution For two internals we provide L^AT_EX 2_ε names so that we can use them elsewhere in the kernel (and so that they can be used in packages if really needed, e.g., `scrfile`).

\undeclare@file@substitution

```
245 </2ekernel>
246 <*2ekernel | latexrelease>
247 <latexrelease>\IncludeInRelease{2020/10/01}%
248 <latexrelease>          {\declare@file@substitution}{File substitution}%
249 \ExplSyntaxOn
250 \cs_new_eq:NN \declare@file@substitution  \__filehook_subst_add:nn
251 \cs_new_eq:NN \undeclare@file@substitution \__filehook_subst_remove:n
252 \ExplSyntaxOff
253 </2ekernel | latexrelease>
254 <latexrelease>\EndIncludeInRelease
```

We are not fully rolling back the file substitutions in case a rollback encounters a package that contains them, but is itself not setup for rollback. So we just bypass them and hope for the best.

```
255 <latexrelease>\IncludeInRelease{0000/00/00}%
256 <latexrelease>          {\declare@file@substitution}{File substitution}%
257 <latexrelease>
258 <latexrelease>\let \declare@file@substitution \@gobbletwo
259 <latexrelease>\let \undeclare@file@substitution \@gobble
260 <latexrelease>
261 <latexrelease>\EndIncludeInRelease
262 <*2ekernel>
```

(End definition for \declare@file@substitution and \undeclare@file@substitution. These functions are documented on page 806.)

```
263 <@@=>
264 \ExplSyntaxOff
```

2.6 Selecting a file (\set@curr@file)

\set@curr@file Now we hook into \set@curr@file to resolve a possible file substitution, and add **\curr@file** \@expl@@@filehook@set@curr@file@@nNN at the end, after \curr@file is set.

\curr@file@reqd

A file name is built using \expandafter\string\csname<filename>\endcsname to avoid expanding utf8 active characters. The \csname expands the normalization machinery and the routine to resolve a file substitution, returning a control sequence with the same name as the file.

It happens that when <filename> is empty, the generated control sequence is \csname\endcsname, and doing \string on that results in the file `csname\endcsname.tex`. To guard against that we \ifx-compare the generated control sequence with the empty csname. To do so, \csname\endcsname has to be defined, otherwise it would be equal to \relax and we would have false positives. Here we define \csname\endcsname to expand to itself to avoid it matching the definition of some other control sequence.

```
265 </2ekernel>
```

```

266 <*2ekernel | latexrelease>
267 <latexrelease>\IncludeInRelease{2021/06/01}%
268 <latexrelease>          {\set@curr@file}{Setting current file name}%
269 \def\set@curr@file#1{%
270   \begingroup
271     \escapechar\m@ne
272     \let\protect\string
273     \edef~{\string~}%
274     \expandafter\def\csname\expandafter\endcsname
275       \expandafter{\csname\endcsname}%

```

Two file names are set here: `\@curr@file@reqd` which is the file requested by the user, and `\@curr@file` which should be the same, except when we have a file substitution, in which case it holds the actual loaded file. `\@curr@file` is resolved first, to check if a substitution happens. If it doesn't, `\@expl@@@filehook@if@file@replaced@@TF` short-cuts and just copies `\@curr@file`, otherwise the full normalization procedure is executed.

At this stage the file name is parsed and normalized, but if the input doesn't have an extension, the default `.tex` is *not* added to `\@curr@file` because for applications other than `\input` (graphics, for example) the default extension may not be `.tex`. First check if the input has an extension, then if the input had no extension, call `\@expl@@@filehook@drop@extension@@N`. In case of a file substitution, `\@curr@file` will have an extension.

```

276   \@expl@@@filehook@if@no@extension@@nTF{#1}%
277   {\@tempwattrue}{\@tempwafalse}%
278   \@kernel@make@file@csname\@curr@file
279   \@expl@@@filehook@resolve@file@subst@@w {#1}%
280   \@expl@@@filehook@if@file@replaced@@TF
281   {\@kernel@make@file@csname\@curr@file@reqd
282     \@expl@@@filehook@normalize@file@name@@w{#1}%
283     \if@tempswa \@expl@@@filehook@drop@extension@@N\@curr@file@reqd \fi}%
284   {\if@tempswa \@expl@@@filehook@drop@extension@@N\@curr@file \fi
285     \global\let\@curr@file@reqd\@curr@file}%
286   \@expl@@@filehook@clear@replacement@flag@@
287   \endgroup}
288 </2ekernel | latexrelease>
289 <latexrelease>\EndIncludeInRelease

290 <latexrelease>\IncludeInRelease{2020/10/01}%
291 <latexrelease>          {\set@curr@file}{Setting current file name}%
292 <latexrelease>\def\set@curr@file#1{%
293 <latexrelease>  \begingroup
294 <latexrelease>    \escapechar\m@ne
295 <latexrelease>    \expandafter\def\csname\expandafter\endcsname
296 <latexrelease>      \expandafter{\csname\endcsname}%
297 <latexrelease>    \@expl@@@filehook@if@no@extension@@nTF{#1}%
298 <latexrelease>    {\@tempwattrue}{\@tempwafalse}%
299 <latexrelease>    \@kernel@make@file@csname\@curr@file
300 <latexrelease>    \@expl@@@filehook@resolve@file@subst@@w {#1}%
301 <latexrelease>    \@expl@@@filehook@if@file@replaced@@TF
302 <latexrelease>    {\@kernel@make@file@csname\@curr@file@reqd
303 <latexrelease>      \@expl@@@filehook@normalize@file@name@@w{#1}%
304 <latexrelease>      \if@tempswa \@expl@@@filehook@drop@extension@@N\@curr@file@reqd \fi}%
305 <latexrelease>    {\if@tempswa \@expl@@@filehook@drop@extension@@N\@curr@file \fi

```

```

306 <latexrelease> \global\let\@curr@file@reqd\@curr@file}%
307 <latexrelease> \@expl@@@filehook@clear@replacement@flag@@
308 <latexrelease> \endgroup}
309 <latexrelease>\EndIncludeInRelease

310 <latexrelease>\IncludeInRelease{2019/10/01}%
311 <latexrelease> {\set@curr@file}{Setting current file name}%
312 <latexrelease>\def\set@curr@file#1{%
313 <latexrelease> \begingroup
314 <latexrelease> \escapechar\m@ne
315 <latexrelease> \xdef\@curr@file{%
316 <latexrelease> \expandafter\expandafter\expandafter\unquote@name
317 <latexrelease> \expandafter\expandafter\expandafter{%
318 <latexrelease> \expandafter\string
319 <latexrelease> \csname\@firstofone#1\@empty\endcsname}}%
320 <latexrelease> \endgroup
321 <latexrelease>}
322 <latexrelease>\EndIncludeInRelease

323 <latexrelease>\IncludeInRelease{0000/00/00}%
324 <latexrelease> {\set@curr@file}{Setting current file name}%
325 <latexrelease>\let\set@curr@file\@undefined
326 <latexrelease>\EndIncludeInRelease
327 <*2ekernel>

```

(End definition for \set@curr@file, \@curr@file, and \@curr@file@reqd.)

```

\@filehook@set@CurrentFile
\@kernel@make@file@csname
\@set@curr@file@aux

```

Todo: This should get internalized using @expl@ names

```

328 </2ekernel>
329 <*2ekernel | latexrelease>
330 <latexrelease>\IncludeInRelease{2020/10/01}%
331 <latexrelease> {\@kernel@make@file@csname}{Make file csname}%
332 \def\@kernel@make@file@csname#1#2#3{%
333 \xdef#1{\expandafter\@set@curr@file@aux
334 \csname\expandafter#2\@firstofone#3\@nil\endcsname}}

```

This auxiliary compares \<filename> with \csname\endcsname to check if the empty .tex file was requested.

```

335 \def\@set@curr@file@aux#1{%
336 \expandafter\ifx\csname\endcsname#1%
337 .tex\else\string#1\fi}

```

Then we call \@expl@@@filehook@set@curr@file@@nNN once for \@curr@file to set \CurrentFile(Path)Used and once for \@curr@file@reqd to set \CurrentFile(Path). Here too the slower route is only used if a substitution happened, but here \@expl@@@filehook@if@file@replaced@@TF can't be used because the flag is reset at the \endgroup above, so we check if \@curr@file and \@curr@file@reqd differ. This macro is issued separate from \set@curr@file because it changes \CurrentFile, and side-effects would quickly get out of control.

```

338 \def\@filehook@set@CurrentFile{%
339 \@expl@@@filehook@set@curr@file@@nNN{\@curr@file}%
340 \CurrentFileUsed\CurrentFilePathUsed
341 \ifx\@curr@file@reqd\@curr@file
342 \let\CurrentFile\CurrentFileUsed
343 \let\CurrentFilePath\CurrentFilePathUsed

```



```

344 \else
345 \expl@@@filehook@set@curr@file@@nNN{\@curr@file@reqd}%
346 \CurrentFile\CurrentFilePath
347 \fi}
348 </2ekernel|latexrelease>
349 <latexrelease>\EndIncludeInRelease
350 <*2ekernel>

(End definition for \@filehook@set@CurrentFile, \@kernel@make@file@csname, and
\@set@curr@file@aux.)

```

`\@@_set_curr_file:nNN` When inputting a file, `\set@curr@file` does a file lookup (in `\input@path` and `\l_file_search_path_seq`) and returns the actual file name (`<base>` plus `<ext>`) in `\CurrentFileUsed`, and in case there's a file substitution, the requested file in `\CurrentFile` (otherwise both are the same). Only the base and extension are returned, regardless of the input (both `path/to/file.tex` and `file.tex` end up as `file.tex` in `\CurrentFile`). The path is returned in `\CurrentFilePath`, in case it's needed.

```

351 </2ekernel>
352 <*2ekernel|latexrelease>
353 <latexrelease>\IncludeInRelease{2020/10/01}%
354 <latexrelease> \@@_set_curr_file:nNN}{Set curr file}%
355 \ExplSyntaxOn
356 <@@=filehook>
357 \cs_new_protected:Npn \__filehook_set_curr_file:nNN #1
358 {
359 \exp_args:Nf \__filehook_file_parse_full_name:nN {#1}
360 \__filehook_set_curr_file_assign:nnnNN
361 }
362 \cs_new_protected:Npn \__filehook_set_curr_file_assign:nnnNN #1 #2 #3 #4 #5
363 {
364 \str_set:Nn #5 {#1}
365 \str_set:Nn #4 {#2#3}
366 }
367 \ExplSyntaxOff
368 </2ekernel|latexrelease>
369 <latexrelease>\EndIncludeInRelease
370 <*2ekernel>

(End definition for \@@_set_curr_file:nNN and \@@_set_curr_file_assign:nnnNN.)

```

2.7 Replacing a file and detecting loops

`__filehook_resolve_file_subst:w` Start by sanitizing the file with `__filehook_file_parse_full_name:nN` then do `__filehook_file_subst_begin:nnn{<path>}{<name>}{<ext>}`.
`__filehook_normalize_file_name:w`
`__filehook_file_name_compose:nnn`

```

371 </2ekernel>
372 <*2ekernel|latexrelease>
373 <latexrelease>\IncludeInRelease{2020/10/01}%
374 <latexrelease> {\__filehook_resolve_file_subst:w}{Replace files detect loops}%
375 \ExplSyntaxOn
376 \cs_new:Npn \__filehook_resolve_file_subst:w #1 \@nil
377 { \__filehook_file_parse_full_name:nN {#1} \__filehook_file_subst_begin:nnn }
378 \cs_new:Npn \__filehook_normalize_file_name:w #1 \@nil
379 { \__filehook_file_parse_full_name:nN {#1} \__filehook_file_name_compose:nnn }
380 \cs_new:Npn \__filehook_file_name_compose:nnn #1 #2 #3

```

```
381 { \tl_if_empty:nF {#1} { #1 / } #2#3 }
```

Since the file replacement is done expandably in a `\csname`, use a flag to remember if a substitution happened. We use this in `\set@curr@file` to short-circuit some of it in case no substitution happened (by far the most common case, so it's worth optimizing). The flag raised during the file substitution algorithm must be explicitly cleared after the `__filehook_if_file_replaced:TF` conditional is no longer needed, otherwise further uses of `__filehook_if_file_replaced:TF` will wrongly return true.

```
382 \flag_new:n { __filehook_file_replaced }
383 \cs_new:Npn \__filehook_if_file_replaced:TF #1 #2
384 { \flag_if_raised:nTF { __filehook_file_replaced } {#1} {#2} }
385 \cs_new_protected:Npn \__filehook_clear_replacement_flag:
386 { \flag_clear:n { __filehook_file_replaced } }
```

First off, start by checking if the current file ($\langle name \rangle + \langle ext \rangle$) has a declared substitution. If not, then just put that as the name (including a possible $\langle path \rangle$ in this case): this is the default case with no substitutions, so it's the first to be checked. The auxiliary `__filehook_file_subst_tortoise_hare:nn` sees that there's no replacement for `#2#3` and does nothing else.

```
387 \cs_new:Npn \__filehook_file_subst_begin:nnn #1 #2 #3
388 {
389   \__filehook_file_subst_tortoise_hare:nn { #2#3 } { #2#3 }
390   { \__filehook_file_name_compose:nnn {#1} {#2} {#3} }
391 }
392 \ExplSyntaxOff
393 </2ekernel|latexrelease>
394 <latexrelease>\EndIncludeInRelease
395 <*2ekernel>
```

2.7.1 The Tortoise and Hare algorithm

If there is a substitution ($\langle true \rangle$ in the first `\cs_if_exist:cTF` below), then first check if there is no substitution down the line: this should be the second most common case, of one file replaced by another. In that case just leave the substitution there and the job is done. If any substitution happens, then the `\flag __filehook_file_replaced` is raised (conditionally, because checking if a flag is raised is much faster than raising it over and over again).

If, however there are more substitutions, then we need to check for a possible loop in the substitutions, which would otherwise put `TeX` in an infinite loop if just an exhaustive expansion was used.

To detect a loop, the *Tortoise and Hare* algorithm is used. The name of the algorithm is an analogy to Aesop's fable, in which the Hare outruns a Tortoise. The two pointers here are the csnames which contains each file replacement, both of which start at the position zero, which is the file requested. In the inner part of the macro below, `__filehook_file_subst_loop:cc` is called with `\@file-subst@<file>` and `\@file-subst@\@file-subst@<file>`; that is, the substitution of $\langle file \rangle$ and the substitution of that substitution: the Tortoise walks one step while the Hare walks two.

Within `__filehook_file_subst_loop:NN` the two substitutions are compared, and if they lead to the same file it means that there is a loop in the substitutions. If there's no loop, `__filehook_file_subst_tortoise_hare:nn` is called again with the Tortoise at position 1 and the hare at 2. Again, the substitutions are checked ahead of the Hare

pointer to check that it won't run too far; in case there is no loop in the declarations, eventually one of the `\cs_if_exist:cTF` below will go *false* and the algorithm will end; otherwise it will run until the Hare reaches the same spot as the tortoise and a loop is detected.

```

396 </2ekernel>
397 <*2ekernel|latexrelease>
398 <latexrelease>\IncludeInRelease{2020/10/01}%
399 <latexrelease> {\__filehook_file_subst_tortoise_hare:nn}{Tortoise and Hare}%
400 \ExplSyntaxOn
401 \cs_new:Npn \__filehook_file_subst_tortoise_hare:nn #1 #2 #3
402 {
403   \cs_if_exist:cTF { @file-subst@ #2 }
404   {
405     \flag_if_raised:nF { __filehook_file_replaced }
406     { \flag_raise:n { __filehook_file_replaced } }
407     \cs_if_exist:cTF { @file-subst@ \use:c { @file-subst@ #2 } }
408     {
409       \__filehook_file_subst_loop:cc
410       { @file-subst@ #1 }
411       { @file-subst@ \use:c { @file-subst@ #2 } }
412     }
413     { \use:c { @file-subst@ #2 } }
414   }
415   { #3 }
416 }

```

This is just an auxiliary to check if a loop was found, and continue the algorithm otherwise. If a loop is found, the `.tex` file is used as fallback and `__filehook_file_subst_cycle_error:cN` is called to report the error.

```

417 \cs_new:Npn \__filehook_file_subst_loop:NN #1 #2
418 {
419   \token_if_eq_meaning:NNTF #1 #2
420   {
421     .tex
422     \__filehook_file_subst_cycle_error:cN { @file-subst@ #1 } #1
423   }
424   { \__filehook_file_subst_tortoise_hare:nn {#1} {#2} {#2} }
425 }
426 \cs_generate_variant:Nn \__filehook_file_subst_loop:NN { cc }

```

Showing this type of error expandably is tricky, as we have a very limited amount of characters to show and a potentially large list. As a work around, several errors are printed, each showing one step of the loop, until all the error messages combined show the loop.

```

427 \cs_new:Npn \__filehook_file_subst_cycle_error:NN #1 #2
428 {
429   \__kernel_msg_expandable_error:nnff { kernel } { file-cycle }
430   {#1} { \use:c { @file-subst@ #1 } }
431   \token_if_eq_meaning:NNTF #1 #2
432   { \__filehook_file_subst_cycle_error:cN { @file-subst@ #1 } #2 }
433 }
434 \cs_generate_variant:Nn \__filehook_file_subst_cycle_error:NN { c }

```

And the error message:

```

435 \__kernel_msg_new:nnn { kernel } { file-cycle }
436 { File~loop!~#1~replaced~by~#2... }

(End definition for \__filehook_resolve_file_subst:w and others.)

437 \ExplSyntaxOff
438 \</2ekernel | latexrelease>
439 \<latexrelease>\EndIncludeInRelease
440 \<*2ekernel>
441 \<@@=>

```

2.8 Preventing a package from loading

We support the use case of preventing a package from loading but not any other type of files (e.g., classes).

```

\disable@package@load \disable@package@load defines \@pkg-disable@<package> to expand to some code #2
\reenable@package@load instead of loading the package.
\@disable@packageload@do
442 \</2ekernel>
443 \<*2ekernel | latexrelease>
444 \<latexrelease>\IncludeInRelease{2020/10/01}%
445 \<latexrelease>          {\disable@package@load}{Disable packages}%
446 \def\disable@package@load#1#2{%
447   \global\@namedef{\@pkg-disable@#1.\@pkgextension}{#2}}
448 \def\@disable@packageload@do#1#2{%
449   \ifundefined{\@pkg-disable@#1}{#2}%
450   {\@nameuse{\@pkg-disable@#1}}}

\reenable@package@load undefines \@pkg-disable@<package> to realow loading
a package.

451 \def\reenable@package@load#1{%
452   \global\expandafter\let
453   \csname \@pkg-disable@#1.\@pkgextension \endcsname \@undefined}
454 \</2ekernel | latexrelease>
455 \<latexrelease>\EndIncludeInRelease
456 \<latexrelease>\IncludeInRelease{0000/00/00}%
457 \<latexrelease>          {\disable@package@load}{Disable packages}%
458 \<latexrelease>
459 \<latexrelease>\let\disable@package@load \@undefined
460 \<latexrelease>\let\@disable@packageload@do\@undefined
461 \<latexrelease>\let\reenable@package@load \@undefined
462 \<latexrelease>\EndIncludeInRelease
463 \<*2ekernel>

(End definition for \disable@package@load, \reenable@package@load, and
\@disable@packageload@do. These functions are documented on page 807.)

```

2.9 High-level interfaces for L^AT_EX

None so far and the general feeling for now is that the hooks are enough. Packages like filehook, etc., may use them to set up their interfaces (samples are given below) but for the now the kernel will not provide any.

2.10 Internal commands needed elsewhere

Here we set up a few horrible (but consistent) $\text{\LaTeX} 2_{\epsilon}$ names to allow for internal commands to be used outside this module (and in parts that still use $\text{\LaTeX} 2_{\epsilon}$ syntax. We have to unset the @@ since we want double “at” sign in place of double underscores.

```

464 <@@=)
465 </2ekernel>
466 <*2ekernel | latexrelease>
467 <latexrelease>\IncludeInRelease{2020/10/01}%
468 <latexrelease>    {\@expl@@@filehook@if@no@extension@@nTF}{2e tmp interfaces}%
469 \ExplSyntaxOn

470 \cs_new_eq:NN \@expl@@@filehook@if@no@extension@@nTF
471                \__filehook_if_no_extension:nTF

472 \cs_new_eq:NN \@expl@@@filehook@set@curr@file@@nNN
473                \__filehook_set_curr_file:nNN

474 \cs_new_eq:NN \@expl@@@filehook@resolve@file@subst@@w
475                \__filehook_resolve_file_subst:w

476 \cs_new_eq:NN \@expl@@@filehook@normalize@file@name@@w
477                \__filehook_normalize_file_name:w

478 \cs_new_eq:NN \@expl@@@filehook@if@file@replaced@@TF
479                \__filehook_if_file_replaced:TF

480 \cs_new_eq:NN \@expl@@@filehook@clear@replacement@flag@@
481                \__filehook_clear_replacement_flag:

482 \cs_new_eq:NN \@expl@@@filehook@drop@extension@@N
483                \__filehook_drop_extension:N

484 \cs_new_eq:NN \@expl@@@filehook@file@push@@
485                \__filehook_file_push:

486 \cs_new_eq:NN \@expl@@@filehook@file@pop@@
487                \__filehook_file_pop:

488 \cs_new_eq:NN \@expl@@@filehook@file@pop@assign@@nnnn
489                \__filehook_file_pop_assign:nnnn

490 \ExplSyntaxOff

```

This one specifically has to be undefined because it is left over in the input stream from `\InputIfFileExists` and executed when `latexrelease` is loaded. It cannot be `\let` to `\@undefined` otherwise it would error as well, so it is `\let` to `\relax` to be silently ignored when loading `\latexrelease`.

```

491 </2ekernel | latexrelease>
492 <latexrelease>\EndIncludeInRelease
493 <latexrelease>
494 <latexrelease>\IncludeInRelease{0000/00/00}%
495 <latexrelease>    {\@expl@@@filehook@if@no@extension@@nTF}{2e tmp interfaces}%
496 <latexrelease>\let\@expl@@@filehook@file@pop@@\relax
497 <latexrelease>\EndIncludeInRelease
498 <*2ekernel>

```

This ends the kernel code in this file.

```

499 </2ekernel>

```

3 A sample package for structuring the log output

```

500 <*structuredlog>
501 <@@=filehook>

502 \ProvidesExplPackage
503   {structuredlog}{\ltfilehookdate}{\ltfilehookversion}
504   {Structuring the TeX transcript file}

\g_filehook_nesting_level_int Stores the current package nesting level.
505 \int_new:N \g_filehook_nesting_level_int
Initialise the counter with the number of files in the \@currnamestack (the number of
items divided by 3) minus one, because this package is skipped when printing to the log.
506 \int_gset:Nn \g_filehook_nesting_level_int
507   { ( \tl_count:N \@currnamestack ) / 3 - 1 }

(End definition for \g_filehook_nesting_level_int.)

\__filehook_log_file_record:n This macro is responsible for increasing and decreasing the file nesting level, as well as
printing to the log. The argument is either STOPTART or STOP and the action it takes on
the nesting integer depends on that.
508 \cs_new_protected:Npn \__filehook_log_file_record:n #1
509   {
510     \str_if_eq:nnT {#1} {START} { \int_gincr:N \g_filehook_nesting_level_int }
511     \iow_term:x
512     {
513       \prg_replicate:nn { \g_filehook_nesting_level_int } { = } ~
514       ( LEVEL ~ \int_use:N \g_filehook_nesting_level_int \c_space_tl #1 ) ~
515       \CurrentFileUsed
If there was a file replacement, show that as well:
516       \str_if_eq:nnF \CurrentFileUsed \CurrentFile
517       { ~ ( \CurrentFile \c_space_tl requested ) }
518       \iow_newline:
519     }
520     \str_if_eq:nnT {#1} {STOP} { \int_gdecr:N \g_filehook_nesting_level_int }
521   }

Now just hook the macro above in the generic file/before...
522 \AddToHook{file/before}{ \__filehook_log_file_record:n { START } }
...and file/after hooks. We don't want to install the file/after hook immediately,
because that would mean it is the first time executed when the package finishes. We
therefore put the declaration inside \AddToHookNext so that it gets only installed when
we have left this package.
523 \AddToHookNext{file/after}
524   { \AddToHook{file/after}{ \__filehook_log_file_record:n { STOP } } }

(End definition for \__filehook_log_file_record:n.)

525 <@@=
526 </structuredlog>

```

4 Package emulations

4.1 Package `atveryend` emulation

With the new hook management and the hooks in `\enddocument` all of `atveryend` is taken care of. We can make an emulation only here after the substitution functionality is available:

```
527 \*2kernel)
528 \declare@file@substitution{atveryend.sty}{atveryend-ltx.sty}
529 \*2kernel)
```

Here is the package file we point to:

```
530 \*atveryend-ltx)
531 \ProvidesPackage{atveryend-ltx}
532 [2020/08/19 v1.0a
533 Emulation of the original atvery package^^Jwith kernel methods]
```

Here are new definitions for its interfaces now pointing to the hooks in `\enddocument`

```
534 \newcommand\AfterLastShipout {\AddToHook{enddocument/afterlastpage}}
535 \newcommand\AtVeryEndDocument {\AddToHook{enddocument/afteraux}}
```

Next one is a bit of a fake, but the result should normally be as expected. If not, one needs to add a rule to sort the code chunks in `enddocument/info`.

```
536 \newcommand\AtEndAfterFileList{\AddToHook{enddocument/info}}
537 \newcommand\AtVeryVeryEnd {\AddToHook{enddocument/end}}
```

`\BeforeClearDocument` This one is the only one we don't implement or rather don't have a dedicated hook in the code.

```
538 \ExplSyntaxOn
539 \newcommand\BeforeClearDocument[1]
540 { \AtEndDocument{#1}
541   \atveryend@DEPRECATED{BeforeClearDocument \tl_to_str:n{#1}}
542 }
543 \cs_new:Npn\atveryend@DEPRECATED #1
544 {\iow_term:x{=====~DEPRECATED~USAGE~#1~=====}}
545 \ExplSyntaxOff
```

(End definition for `\BeforeClearDocument`.)

```
546 \*atveryend-ltx)
```

File U

ltshipout.dtx

Contents

1 Introduction

The code provides an interface to the `\shipout` primitive of T_EX which is called when a finished pages is finally “shipped out” to the target output file, e.g., the `.dvi` or `.pdf` file. A good portion of the code is based on ideas by Heiko Oberdiek implemented in his packages `atbegshi` and `atenddvi` even though the interfaces are somewhat different.³⁴

1.1 Overloading the `\shipout` primitive

`\shipout`

With this implementation T_EX’s `shipout` primitive is no longer available for direct use. Instead `\shipout` is running some (complicated) code that picks up the box to be shipped out regardless of how that is done, i.e., as a constructed `\vbox` or `\hbox` or as a box register.

It then stores it in a named box register. This box can then be manipulated through a set of hooks after which it is shipped out for real.

Each `shipout` that actually happens (i.e., where the material is not discarded for one or the other reason) is recorded and the total number is available in a readonly variable and in a L^AT_EX counter.

`\RawShipout`

This command implements a simplified `shipout` that bypasses the foreground and background hooks, e.g., only `shipout/firstpage` and `shipout/lastpage` are executed and the total `shipout` counters are incremented.

The command doesn’t use `\ShipoutBox` but its own private box register so that it can be used inside of `shipout` hooks to do some additional `shipouts` while already in the output routine with the current page being stored in `\ShipoutBox`. It does have access to `\ShipoutBox` if it is used in `shipout/before` (or `shipout/after`) and can use its content.

It is safe to use it in `shipout/before` or `shipout/after` but not necessarily in the other `shipout/...` hooks as they are intended for special processing.

³⁴Heiko’s interfaces are emulated by the kernel code, if a document requests his packages, so older documents will continue to work.

`\ShipoutBox`
`\l_shipout_box`

This box register is called `\ShipoutBox` (alternatively available via the L3 name `\l_shipout_box`).

This box is a “local” box and assignments to it should be done only locally. Global assignments (as done by some packages with older code where this box is known as 255) may work but they are conceptually wrong and may result in errors under certain circumstances.

During the execution of `shipout/before` this box contains the accumulated material for the page, but not yet any material added by other shipout hooks. During execution of `shipout/after`, i.e., after the shipout has happened, the box also contains any background or foreground material.

Material from the hooks `shipout/firstpage` or `shipout/lastpage` is not included (but only used during the actual shipout) to facilitate reuse of the box data (e.g., `shipout/firstpage` material should never be added to a later page of the output).

`\l_shipout_box_ht_dim`
`\l_shipout_box_dp_dim`
`\l_shipout_box_wd_dim`
`\l_shipout_box_ht_plus_dp_dim`

The shipout box dimensions are available in the L3 registers `\l_shipout_box_ht_dim`, etc. (there are no L^AT_EX 2_ε names).³⁵ These variables can be used inside the hook code for `shipout/before`, `shipout/foreground` and `shipout/background` if needed.

1.2 Provided hooks

`shipout/before`
`shipout/after`
`shipout/foreground`
`shipout/background`
`shipout/firstpage`
`shipout/lastpage`

The code for `\shipout` offers a number of hooks into which packages (or the user) can add code to support different use cases. These are:

shipout/before This hook is executed after the finished page has been stored in `\ShipoutBox` / `\l_shipout_box`. It can be used to alter that box content or to discard it completely (see `\DiscardShipoutBox` below).

You can use `\RawShipout` inside this hook for special use cases. It can make use of `\ShipoutBox` (which doesn’t yet include the background and foreground material).

Note: It is not possible (or say advisable) to try and use this hook to typeset material with the intention to return it to main vertical list, it will go wrong and give unexpected results in many cases—for starters it will appear after the current page not before or it will vanish or the vertical spacing will be wrong!

shipout/background This hook adds a picture environment into the background of the page with the (0,0) coordinate in the top-left corner using a `\unitlength` of 1pt.

It should therefore only receive `\put` commands or other commands suitable in a `picture` environment and the vertical coordinate values would normally be negative.

³⁵Might need changing, but HO’s version as strings is not really helpful I think).

Technically this is implemented by adding a zero-sized `\hbox` as the very first item into the `\ShipoutBox` containing that `picture` environment. Thus the rest of the box content will overprint what ever is typeset by that hook.

shipout/foreground This hook adds a picture environment into the foreground of the page with the (0,0) coordinate in the top-left corner using a `\unitlength` of 1pt.

Technically this is implemented by adding a zero-sized `\hbox` as the very last item into the `\ShipoutBox` and raising it up so that it still has its (0,0) point in the top-left corner. But being placed after the main box content it will be typeset later and thus overprints it (i.e., is in the foreground).

shipout/firstpage The material from this hook is executed only once at the very beginning of the first output page that is shipped out (i.e., not discarded at the last minute). It should only contain `\special` or similar commands needed to direct post processors handling the .dvi or .pdf output.³⁶

This hook is added to the very first page regardless of how it is shipped out (i.e., with `\shipout` or `\RawShipout`).

shipout/lastpage The corresponding hook to add `\specials` at the very end of the output file. It is only executed on the very last page of the output file — or rather on the page that L^AT_EX believes is the last one. Again it is executed regardless of the shipout method.

It may not be possible for L^AT_EX to correctly determine which page is the last one without several reruns. If this happens and the hook is non-empty then L^AT_EX will add an extra page to place the material and also request a rerun to get the correct placement sorted out.

shipout/after This hook is executed after a shipout has happened. If the shipout box is discarded this hook is not looked at.

You can use `\RawShipout` inside this hook for special use cases and the main `\ShipoutBox` is still available at this point (but in contrast to `shipout/before` it now includes the background and foreground material).

Note: Just like `shipout/before` this hook is not meant to be used for adding adding typeset material back to the main vertical list—it might vanish or the vertical spacing will be wrong!

As mentioned above the hook `shipout/before` is executed first and can manipulate the prepared shipout box stored in `\ShipoutBox` or set things up for use in `\write` during the actual shipout. It is even run if there was a `\DiscardShipoutBox` request in the document.

The other hooks (except `shipout/after`) are added inside hboxes to the box being shipped out in the following order:

<code>shipout/firstpage</code>	only on the first page
<code>shipout/background</code>	
<code><boxed content of \ShipoutBox></code>	
<code>shipout/foreground</code>	
<code>shipout/lastpage</code>	only on the last page

³⁶In L^AT_EX 2_ε that was already existing, but implemented using a box register with the name `\@begindvibox`.

If any of the hooks has no code then that particular no box is added at that point.

Once the (page) box has been shipped out the `shipout/after` hook is called (while you are still inside the output routine). It is not called if the shipout box was discarded.

In a document that doesn't produce pages, e.g., only makes `\typeouts`, none of the hooks are ever executed (as there is no `\shipout`) not even the `shipout/lastpage` hook.

If `\RawShipout` is used instead of `\shipout` then only the hooks `shipout/firstpage` and `shipout/lastpage` are executed (on the first or last page), all others are bypassed.

1.3 Legacy L^AT_EX commands

`\AtBeginDvi`
`\AtEndDvi`

`\AtBeginDvi` is the existing L^AT_EX 2_ε interface to fill the `shipout/firstpage` hook. This is not really a good name as it is not just supporting `.dvi` but also `.pdf` output or `.xdv`.

`\AtEndDvi` is the counterpart that was not available in the kernel but only through the package `atenddvi`. It fills the `shipout/lastpage` hook.

As these two wrappers have been available for a long time we continue offering them. However, for new code we suggest using the high-level hook management commands directly instead of “randomly-named” wrappers. This will lead to code that is easier to understand and to maintain. For this reason we do not provide any other wrapper commands for the above hooks in the kernel.

1.4 Special commands for use inside the hooks

`\DiscardShipoutBox`
`\shipout_discard:`

`\AddToHookNext {shipout/before} {...\DiscardShipoutBox...}`

The `\DiscardShipoutBox` declaration (L³ name `\shipout_discard:`) requests that on the next shipout the page box is thrown away instead of being shipped to the `.dvi` or `.pdf` file.

Typical applications wouldn't do this unconditionally, but have some processing logic that decides to use or not to use the page.

Note that if this declaration is used directly in the document it may depend on the placement to which page it applies, given that L^AT_EX output routine is called in an asynchronous manner! Thus normally one would use this only as part of the `shipout/before` code.

Todo: Once we have a new mark mechanism available we can improve on that and make sure that the declaration applies to the page that contains it.

In the `atbegshi` package there are a number of additional commands for use inside the `shipout/before` hook. They should normally not be needed any more as one can instead simply add code to the hooks `shipout/before`, `shipout/background` or `shipout/foreground`.³⁷ If `atbegshi` gets loaded then those commands become available as public functions with their original names as given below.

³⁷If that assumption turns out to be wrong it would be trivial to change them to public functions (right now they are private).

1.5 Provided LuaTeX callbacks

`pre_shipout_filter`

Under LuaTeX the `pre_shipout_filter` Lua callback is provided which gets called immediately before the shipout primitive gets invoked. The signature is

```
function(<node> head)
  return true
end
```

The `head` is the list node corresponding to the box to be shipped out. The return value should always be `true`.

1.6 Information counters

`\ReadonlyShipoutCounter`
`\g_shipout_readonly_int`

```
\ifnum\ReadonlyShipoutCounter=...
\int_use:N \g_shipout_readonly_int % expl3 usage
```

This integer holds the number of pages shipped out up to now (including the one to be shipped out when inside the output routine). More precisely, it is incremented only after it is clear that a page will be shipped out, i.e., after the `shipout/before` hook (because that might discard the page)! In contrast `shipout/after` sees the incremented value.

Just like with the `page` counter its value is only accurate within the output routine. In the body of the document it may be off by one as the output routine is called asynchronously!

Also important: it *must not* be set, only read. There are no provisions to prevent that restriction, but if you manipulate it, chaos will be the result. To emphasize this fact it is not provided as a L^AT_EX counter but as a T_EX counter (i.e., a command), so `\Alph{\ReadonlyShipoutCounter}` etc, would not work.

`totalpages`
`\g_shipout_totalpages_int`

```
\arabic{totalpages}
\int_use:N \g_shipout_totalpages_int % expl3 usage
```

In contrast to `\ReadonlyShipoutCounter`, the `totalpages` counter is a L^AT_EX counter and incremented for each shipout attempt including those pages that are discarded for one or the other reason. Again `shipout/before` sees the counter before it is incremented. In contrast `shipout/after` sees the incremented value.

Furthermore, while it is incremented for each page, its value is never used by L^AT_EX. It can therefore be freely reset or changed by user code, for example, to additionally count a number of pages that are not build by L^AT_EX but are added in a later part of the process, e.g., cover pages or picture pages made externally.

Important: as this is a page-related counter its value is only reliable inside the output routine!

`\PreviousTotalPages`

```
\thetotalpages/\PreviousTotalPages
```

Command that expands to the number of total pages from the previous run. If there was no previous run or if used in the preamble it expands to 0. Note that this is a command and not a counter, so in order to display the number in, say, Roman numerals you have to assign its value to a counter and then use `\Roman` on that counter.

1.7 Debugging shipout code

<code>\DebugShipoutsOn</code>	<code>\DebugShipoutsOn</code>
<code>\DebugShipoutsOff</code>	
<code>\shipout_debug_on:</code>	Turn the debugging of shipout code on or off. This displays changes made to the shipout data structures.
<code>\shipout_debug_off:</code>	

Todo: This needs some rationalizing and may not stay this way.

2 Emulating commands from other packages

The packages in this section are no longer necessary, but as they are used by other packages, they are emulated when they are explicitly loaded with `\usepackage` or `\RequirePackage`.

Please note that the emulation only happens if the package is explicitly requested, i.e., the commands documented below are not automatically available in the L^AT_EX kernel! If you write a new package we suggest to use the appropriate kernel hooks directly instead of loading the emulation.

2.1 Emulating atbegshi

<code>\AtBeginShipoutUpperLeft</code>	<code>\AddToHook {shipout/before}</code>
<code>\AtBeginShipoutUpperLeftForeground</code>	<code>{... \AtBeginShipoutUpperLeft{<code>}...}</code>

This adds a `picture` environment into the background of the shipout box expecting `<code>` to contain `picture` commands. The same effect can be obtained by simply using kernel features as follows:

`\AddToHook{shipout/background}{<code>}`

There is one technical difference: if `\AtBeginShipoutUpperLeft` is used several times each invocation is put into its own box inside the shipout box whereas all `<code>` going into `shipout/background` ends up all in the same box in the order it is added or sorted based on the rules for the hook chunks.

`\AtBeginShipoutUpperLeftForeground` is similar with the difference that the `picture` environment is placed in the foreground. To model it with the kernel functions use the hook `shipout/foreground` instead.

<code>\AtBeginShipoutAddToBox</code>	<code>\AddToHook {shipout/before} {... \AtBeginShipoutAddToBox{<code>}...}</code>
<code>\AtBeginShipoutAddToBoxForeground</code>	

These work like `\AtBeginShipoutUpperLeft` and `\AtBeginShipoutUpperLeftForeground` with the difference that `<code>` is directly placed into an `\hbox` inside the shipout box and not surrounded by a `picture` environment.

To emulate them using `shipout/background` or `shipout/foreground` you may have to wrap `<code>` into a `\put` statement but if the code is not doing any typesetting just adding it to the hook should be sufficient.

<code>\AtBeginShipoutBox</code>	This is the name of the shipout box as <code>atbegshi</code> knows it.
---------------------------------	--

`\AtBeginShipoutOriginalShipout`

This is the name of the `\shipout` primitive as `atbegshi` knows it. This bypasses all the mechanisms set up by the \LaTeX kernel and there are various scenarios in which it can therefore fail. It should only be used to run existing legacy `atbegshi` code but not in newly developed applications.

The kernel alternative is `\RawShipout` which is integrated with the \LaTeX mechanisms and updates, for example, the `\ReadonlyShipoutCounter` counter. Please use `\RawShipout` for new code if you want to bypass the before, foreground and background hooks.

`\AtBeginShipoutInit`

By default `atbegshi` delayed its action until `\begin{document}`. This command was forcing it in an earlier place. With the new concept it does nothing.

`\AtBeginShipout`

`\AtBeginShipout{<code>} \equiv \AddToHook{shipout/before}{<code>}`

`\AtBeginShipoutNext`

`\AtBeginShipoutNext{<code>} \equiv \AddToHookNext{shipout/before}{<code>}`

This is equivalent to filling the `shipout/before` hook by either using `\AddToHook` or `\AddToHookNext`, respectively.

`\AtBeginShipoutFirst`

`\AtBeginShipoutDiscard`

The `atbegshi` names for `\AtBeginDvi` and `\DiscardShipoutBox`.

2.2 Emulating `everyshi`

The `everyshi` package is providing commands to run arbitrary code just before the `shipout` starts. One point of difference: in the new `shipout` hooks the page is available as `\ShipoutBox` for inspection of change, one should not manipulate box 255 directly inside `shipout/before`, so old code doing this would change to use `\ShipoutBox` instead of 255 or `\@cc1v`.

`\EveryShipout`

`\EveryShipout{<code>} \equiv \AddToHook{shipout/before}{<code>}`

`\AtNextShipout`

`\AtNextShipout{<code>} \equiv \AddToHookNext{shipout/before}{<code>}`

However, most use cases for `everyshi` are attempts to put some picture or text into the background or foreground of the page and that can be done today simply by using the `shipout/background` and `shipout/foreground` hooks without any need to coding.

2.3 Emulating `atenddvi`

The `atenddvi` package implemented only a single command: `\AtEndDvi` and that is now available out of the box so the emulation makes the package a no-op.

2.4 Emulating everypage

This package patched the original `\@begindvi` hook and replaced it with its own version. Its functionality is now covered by the hooks offered by the kernel so that there is no need for such patching any longer.

<code>\AddEverypageHook</code>	<code>\AddEverypageHook{<code>} ≡</code> <code>\AddToHook{shipout/background}{\put(1in,-1in){<code>}}</code>
--------------------------------	---

`\AddEverypageHook` is adding something into the background of every page at a position of 1in to the right and 1in down from the top left corner of the page. By using the kernel hook directly you can put your material directly to the right place, i.e., use other coordinates in the `\put` statement above.

<code>\AddThispageHook</code>	<code>\AddThispageHook{<code>} ≡</code> <code>\AddToHookNext{shipout/background}{\put(1in,-1in){<code>}}</code>
-------------------------------	--

The `\AddThispageHook` wrapper is similar but uses `\AddToHookNext`.

3 The Implementation

1 `<@@=shipout>`

At the moment the whole module rolls back in one go, but if we make any modifications in later releases this will then need splitting.

2 `<*2ekernel | latexrelease>`
3 `<latexrelease>\IncludeInRelease{2020/10/01}%`
4 `<latexrelease> \shipout}{Hook management (shipout))}%`
5 `\ExplSyntaxOn`

3.1 Debugging

<code>\g__shipout_debug_bool</code>	Holds the current debugging state. 6 <code>\bool_new:N \g__shipout_debug_bool</code>
-------------------------------------	---

(End definition for `\g__shipout_debug_bool`.)

<code>\shipout_debug_on:</code> <code>\shipout_debug_off:</code> <code>__shipout_debug:n</code> <code>__shipout_debug_gset:</code>	Turns debugging on and off by redefining <code>__shipout_debug:n</code> . 7 <code>\cs_new_eq:NN __shipout_debug:n \use_none:n</code> 8 <code>\cs_new_protected:Npn \shipout_debug_on:</code> 9 <code>{</code> 10 <code>\bool_gset_true:N \g__shipout_debug_bool</code> 11 <code>__shipout_debug_gset:</code> 12 <code>}</code> 13 <code>\cs_new_protected:Npn \shipout_debug_off:</code> 14 <code>{</code> 15 <code>\bool_gset_false:N \g__shipout_debug_bool</code> 16 <code>__shipout_debug_gset:</code> 17 <code>}</code> 18 <code>\cs_new_protected:Npn __shipout_debug_gset:</code> 19 <code>{</code> 20 <code>\cs_gset_protected:Npx __shipout_debug:n ##1</code> 21 <code>{ \bool_if:NT \g__shipout_debug_bool {##1} }</code> 22 <code>}</code>
---	--

(End definition for `\shipout_debug_on:` and others. These functions are documented on page 830.)

`\ShipoutBox` The box filled with the page to be shipped out (both L3 and L^AT_EX 2_ε name).
`\l_shipout_box`

```
23 \box_new:N \l_shipout_box
24 \cs_set_eq:NN \ShipoutBox \l_shipout_box
```

(End definition for `\ShipoutBox` and `\l_shipout_box`. These functions are documented on page 826.)

`\l__shipout_raw_box` The `\RawShipout` gets its own box but it is internal as there is no hook manipulation for it.

```
25 \box_new:N \l__shipout_raw_box
```

(End definition for `\l__shipout_raw_box`.)

`__shipout_finalize_box:` For Lua_T_EX invoke the `pre_shipout_filter` callback.

```
26 \sys_if_engine luatex:TF
27 {
28   \newluafunction \__shipout_finalize_box:
29   \exp_args:Nx \everyjob {
30     \exp_not:V \everyjob
31     \exp_not:N \lua_now:n {
32       luatexbase.create_callback('pre_shipout_filter', 'list')
33       local~call, getbox, setbox = luatexbase.call_callback, tex.getbox, tex.setbox~
34       lua.get_functions_table()[\the \__shipout_finalize_box:] = function()
35         local~result = call('pre_shipout_filter', getbox(\the \l_shipout_box))
36         if~not (result == true) then~
37           setbox(\the \l_shipout_box, result~or~nil)
38         end~
39       end
40     }
41   }
42   \protected \luadef \__shipout_finalize_box: \the \__shipout_finalize_box:
43 } {
44   \cs_set_eq:NN \__shipout_finalize_box: \scan_stop:
45 }
```

(End definition for `__shipout_finalize_box:.`)

`__shipout_execute:` This is going to be the code run by `\shipout`. The code follows closely the ideas from `atbegshi`, so not documenting that here for now.

```
46 \cs_set_protected:Npn \__shipout_execute: {
47   \tl_set:Nx \l__shipout_group_level_tl
48   { \int_value:w \tex_currentgrouplevel:D }
49   \tex_afterassignment:D \__shipout_execute_test_level:
50   \tex_setbox:D \l_shipout_box
51 }
```

(End definition for `__shipout_execute:.`)

`\shipout` Overloading the `\shipout` primitive:

```
52 \cs_gset_eq:NN \shipout \__shipout_execute:
```

(End definition for `\shipout`. This function is documented on page 825.)

`\l__shipout_group_level_tl` Helper token list to record the group level at which `__shipout_execute:` is encountered.

```
53 \tl_new:N \l__shipout_group_level_tl
```

(End definition for `\l__shipout_group_level_tl`.)

`_shipout_execute_test_level:` If the group level has changed then we are still constructing `\l_shipout_box` and to continue we need to wait until the current group has finished, hence the `\tex_aftergroup:D`.

```
54 \cs_new:Npn \__shipout_execute_test_level: {
55   \int_compare:nNnT
56     \l__shipout_group_level_tl < \tex_currentgrouplevel:D
57     \tex_aftergroup:D \__shipout_execute_cont:
58 }
```

(End definition for `_shipout_execute_test_level:.`)

`__shipout_execute_cont:` This does the actual shipout running several hooks as part of it. The code for them is passed as argument #2 to #4 to `__shipout_execute_main_cont:Nnnn`; the first argument is the box to be shipped out.

```
59 \cs_new:Npn \__shipout_execute_cont: {
60   \__shipout_execute_main_cont:Nnnn
61   \l_shipout_box
62   { \hook_use:n {shipout/before} }
63   { \hook_if_empty:nF {shipout/foreground}
64     { \__shipout_add_foreground_picture:n
65       { \hook_use:n {shipout/foreground} } } }
```

If the user hook for the background (`shipout/background`) has no code, there might still code in the kernel hook so we need to test for this too. We only test for the `\@kernel@before@shipout@background` though. If the `\@kernel@after@shipout@background` needs executing even if the user hook is empty then we can add another test (or the kernel could put something into the before hook).

```
66   \bool_lazy_and:nnF
67     { \hook_if_empty_p:n {shipout/background} }
68     { \tl_if_empty_p:N \@kernel@before@shipout@background }
69     { \__shipout_add_background_picture:n
70       { \@kernel@before@shipout@background
71         \hook_use:n {shipout/background}
72         \@kernel@after@shipout@background }
73     }
74   }
75   { \hook_use:n {shipout/after} }
76 }
```

(End definition for `__shipout_execute_cont:.`)

`_shipout_execute_main_cont:Nnnn` When we have reached this point the shipout box has been processed and is available in `\l_shipout_box` and ready for real ship out (unless it gets discarded during the process).

The three arguments hold hook code that is executed just before the actual shipout (#1), within the shipout adding background and foreground material (#2) and after the shipout has happened (#3). These are passed as arguments because the same code without those hooks is also used when doing a “raw” shipout implemented by `\RawShipout`. The only hook that is always executed is that for the very last page, i.e., `shipout/lastpage`.

First we quickly check if it is void (can't happen in the standard L^AT_EX output routine but `\shipout` might be called from a package that has some special processing logic). If it is void we aren't shipping anything out and processing ends.³⁸

```

77 \cs_new:Npn \__shipout_execute_main_cont:Nnnn #1#2#3#4 {
78   \box_if_empty:NTF #1
79   { \@latex@warning{Ignoring~ void~ shipout~ box} }
80   {

```

Otherwise we assume that we will ship something and prepare for final adjustments (in particular setting the state of `\protect` while we are running the hook code). We also save the current `\protect` state to restore it later.

```

81 %       \bool_gset_false:N \g_shipout_discard_bool % setting this would disable
82                                     % \DiscardShipoutBox on doc-level
83       \cs_set_eq:NN \__shipout_saved_protect: \protect
84       \set@typeset@protect

```

We also store the current shipout box dimension in registers, so that they can be used in the hook code.³⁹

```

85       \__shipout_get_box_size:N #1

```

Then we execute the `shipout/before` hook (or nothing in case of `\RawShipout`).

```

86       #2

```

In `\g_shipout_totalpages_int` we count all shipout attempts so we increment that counter already here (the other one is incremented later when we know for sure that we do a `\shipout`).

We increment it after running the above hook so that the values for `\g_shipout_totalpages_int` and `\g_shipout_readonly_int` are in sync while the hook is executed (in the case that `totalpages` isn't manually altered or through discarding pages that is).

```

87       \int_gincr:N \g_shipout_totalpages_int

```

The above hook might contain code that requests the page to be discarded so we now test for it.

```

88       \bool_if:NTF \g_shipout_discard_bool
89       { \@latex@info@no@line{Completed~ page~ discarded}
90       \bool_gset_false:N \g_shipout_discard_bool

```

As we are discarding the page box and not shipping anything out, we need to do some house cleaning and reset T_EX's deadcycles so that it doesn't complain about too many calls to the OR without any shipout.

```

91       \tex_deadcycles:D \c_zero_int

```

Todo: In `atbegshi` the box was dropped but is that actually needed? Or the resetting of `\protect` to its kernel value?

```

92 %       \group_begin:
93 %       \box_set_eq_drop:NN #1 #1
94 %       \group_end:
95 %       \cs_set_eq:NN \protect \exp_not:N
96       }

```

³⁸In that case we don't reset the deadcycles, that would be up to the OR processing logic to do.

³⁹This is not really necessary as the code could access them via `\box_ht:N`, etc., but it is perhaps convenient.

Even if there was no explicit request to discard the box it is possible that the code for the hook `shipout/before` has voided the box (by mistake or deliberately). We therefore test once more but this time make it a warning, because the best practice way is to use the request mechanism.

```

97      { \box_if_empty:NTF #1
98        { \@latex@warning{
99          Shipout~ box~ was~ voided~ by~ hook,\MessageBreak
100          ignoring~ shipout~ box }
101        }

```

Finally, if the box is still non-empty we are nearly ready to ship it out. First we increment the total page counter so that we can later test if we have reached the final page according to our available information.⁴⁰

```

102      {
103        \int_gincr:N \g_shipout_readonly_int
104        \__shipout_debug:n {
105          \typeout{Absolute~ page~ =~ \int_use:N \g_shipout_readonly_int
106                  \space (target:~ \@abspage@last)}
107        }

```

Then we store the box sizes again (as they may have changed) and then look at the hooks `shipout/foreground` and `shipout/background`. If either or both are non-empty we add a `picture` environment to the box (in the foreground and/or in the background) and execute the hook code inside that environment.

```

108      \__shipout_get_box_size:N #1

```

The first hook we run is the `shipout/firstpage` hook. This is only done once, then the `__shipout_run_firstpage_hook:` command redefines itself to do nothing. If the hook contains `\specials` for integration at the top of the page they will be temporarily stored in a safe place and added later with `__shipout_add_firstpage_specials:`.

```

109      \__shipout_run_firstpage_hook:

```

Run the hooks for background and foreground or, if this is called by `\RawShipout`, copy the box `\l__shipout_raw_box` to `\l_shipout_box` so that firstpage and lastpage material gets added if necessary (that is always done to `\l_shipout_box`).

```

110      #3

```

We then run `__shipout_add_firstpage_specials:` that adds the content of the hook `shipout/firstpage` to the start of the first page (if non-empty). It is then redefined to do nothing on later pages.

```

111      \__shipout_add_firstpage_specials:

```

Then we check if we have to add the `shipout/lastpage` hook or the corresponding kernel hook because we have reached the last page. This test will be false for all but one (and hopefully the correct) page.

```

112      \int_compare:nNnT \@abspage@last = \g_shipout_readonly_int
113      { \bool_lazy_and:nnF
114        { \hook_if_empty_p:n {shipout/lastpage} }
115        { \tl_if_empty_p:N \@kernel@after@shipout@lastpage }
116        { \__shipout_debug:n { \typeout{Executing~ lastpage~ hook~
117                                on~ page~ \int_use:N \g_shipout_readonly_int } }
118        \__shipout_add_foreground_box:n

```

⁴⁰Doing that earlier would be wrong because we might end up with the last page counted but discard and then we have no place to add the final objects into the output file.

```

119             { \UseHook{shipout/lastpage}
120               \@kernel@after@shipout@lastpage }
121           }
122           \bool_gset_true:N \g__shipout_lastpage_handled_bool
123         }
124         \__shipout_finalize_box:

```

Finally we run the actual \TeX primitive for shipout. As that will expand delayed \write statements inside the page in which protected commands should not expand we first change \protect to the appropriate definition for that case.

```

125         \cs_set_eq:NN \protect \exp_not:N
126         \tex_shipout:D \box_use:N \l_shipout_box

```

The \l_shipout_box may contain the firstpage material if this was the very first shipout. That makes it unsuitable for reuse in another shipout, so as a safety measure the next command resets \l_shipout_box to its earlier state if that is necessary. On later pages this is then a no-op.

```

127         \__shipout_drop_firstpage_specials:

```

The \shipout/after hook (if in #4) needs to run with \protected commands again being executed, because that hook will “typeset” material added at the top of the next page.

```

128         \set@typeset@protect
129         #4
130       }
131     }

```

Restore the value of \protect in case \shipout is called outside of the output routine (where it is automatically restored because of the implicit group).

```

132     \cs_set_eq:NN \protect \__shipout_saved_protect:
133   }
134 }

```

(End definition for $\text{__shipout_execute_main_cont:Nnnn}$.)

$\text{__shipout_execute_raw:}$ This implements the “raw” shipout which bypasses the before, foreground, background and after hooks. It follows the same pattern than $\text{__shipout_execute_raw:}$ except that it finally calls $\text{__shipout_execute_main_cont:Nnnn}$ with three empty arguments. instead of the hook code.

```

135 \cs_set_protected:Npn \__shipout_execute_raw: {
136   \tl_set:Nx \l__shipout_group_level_tl
137   { \int_value:w \tex_currentgrouplevel:D }
138   \tex_afterassignment:D \__shipout_execute_test_level_raw:
139   \tex_setbox:D \l__shipout_raw_box
140 }
141 \cs_new:Npn \__shipout_execute_test_level_raw: {
142   \int_compare:nNnT
143     \l__shipout_group_level_tl < \tex_currentgrouplevel:D
144     \tex_aftergroup:D \__shipout_execute_nohooks_cont:
145 }

```

Well, not totally empty arguments, we add some debugging if we are actually doing a shipout.

```

146 \cs_new:Npn \__shipout_execute_nohooks_cont: {
147   \__shipout_execute_main_cont:Nnnn \l__shipout_raw_box
148   {} { \__shipout_debug:n{ \typeout{Doing~ raw~ shipout~ ...} } }

```

```

149         \box_set_eq:NN \l_shipout_box \l__shipout_raw_box } {}
150     }

(End definition for \__shipout_execute_raw: and \__shipout_execute_test_level_raw:.)

```

\RawShipout The interface name for raw shipout.

```

151 \cs_gset_eq:NN \RawShipout \__shipout_execute_raw:

(End definition for \RawShipout. This function is documented on page 825.)

```

__shipout_saved_protect: Remember the current `\protect` state.

```

152 \cs_new_eq:NN \__shipout_saved_protect: \protect

(End definition for \__shipout_saved_protect:.)

```

shipout/before Declaring all hooks for the shipout code.

```

shipout/after 153 \hook_new:n{shipout/before}
shipout/foreground 154 \hook_new:n{shipout/after}
shipout/background 155 \hook_new:n{shipout/foreground}
shipout/firstpage 156 \hook_new:n{shipout/background}
shipout/lastpage 157 \hook_new:n{shipout/firstpage}
158 \hook_new:n{shipout/lastpage}

```

(End definition for `shipout/before` and others. These functions are documented on page 826.)

`\@kernel@after@shipout@lastpage` And here are the internal kernel hooks going before or after the public ones where needed.

```

\@kernel@before@shipout@background
\@kernel@after@shipout@background
159 \let\@kernel@after@shipout@lastpage\@empty
160 \let\@kernel@before@shipout@background\@empty
161 \let\@kernel@after@shipout@background\@empty

```

(End definition for `\@kernel@after@shipout@lastpage`, `\@kernel@before@shipout@background`, and `\@kernel@after@shipout@background`.)

__shipout_run_firstpage_hook: There are three commands to handle the `shipout/firstpage` hook: `__shipout_run_firstpage_hook:`, `__shipout_add_firstpage_specials:` and `__shipout_drop_firstpage_specials:`.

That hook is supposed to contain `\specials` and similar material to be placed at the very beginning of the output page and so it needs careful placing to avoid that anything else gets in front of it. And this means we have to wait with this until other hooks such as `shipout/background` have added their bits. It is also important that such `\specials` show up only on the very first page, so if this page gets saved before `\shipout` for later reuse, we have to make sure that they aren't in the saved version.

In addition the hook may also contain code to be executed “first”, e.g., visible from code in `shipout/background` and this conflicts with adding the `\specials` late.

Therefore the processing is split into different parts: `__shipout_run_firstpage_hook:` is done early and checks if there is any material in the hook.

```

162 \cs_new:Npn \__shipout_run_firstpage_hook: {
163     \hook_if_empty:nTF {shipout/firstpage}

```

If not then we define the other two commands to do nothing.

```

164     {
165         \cs_gset_eq:NN \__shipout_add_firstpage_specials: \prg_do_nothing:
166         \cs_gset_eq:NN \__shipout_drop_firstpage_specials: \prg_do_nothing:
167     }

```

If there is material we execute inside a box, which means any `\special` will end up in that box and any other code is executed and can have side effects (as long as they are global).

```

168     {
169         \hbox_set:Nn \l__shipout_firstpage_box { \UseHook{shipout/firstpage} }
170     }

```

Once we are here we change the definition to do nothing next time and we also change the command used to implement `\AtBeginDvi` to become a warning and not add further material to a hook that is never used again.

```

171 \cs_gset_eq:NN \__shipout_run_firstpage_hook: \prg_do_nothing:
172 \cs_gset:Npn \__shipout_add_firstpage_material:Nn ##1 ##2 {
173     \@latex@warning{
174         First~ page~ is~ already~ shipped~ out,~ ignoring\MessageBreak
175         \string##1 }
176     }
177 }

```

(End definition for `__shipout_run_firstpage_hook:.`)

`__shipout_add_firstpage_specials:`
`__shipout_drop_firstpage_specials:`

The `__shipout_add_firstpage_specials:` then adds the `\specials` stored in `\l__shipout_firstpage_box` to the page to be shipped out when the time is ready. Note that if there was no material in the `shipout/firstpage` hook then this command gets redefined to do nothing. But for most documents there is something, e.g., some PostScript header, or some meta data declaration, etc. so by default we assume there is something to do.

```

178 \cs_new:Npn \__shipout_add_firstpage_specials: {

```

First we make a copy of the `\l_shipout_box` that we can restore it later on.

```

179 \box_set_eq:NN \l__shipout_raw_box \l_shipout_box

```

Adding something to the beginning means adding it to the background as that layer is done first in the output.

```

180 \__shipout_add_background_box:n { \hbox_unpack_drop:N \l__shipout_firstpage_box }

```

After the actual shipout `__shipout_drop_firstpage_specials:` is run to restore the earlier content of `\l_shipout_box` and then redefines itself again to do nothing.

As a final act we change the definition to do nothing next time.

```

181 \cs_gset_eq:NN \__shipout_add_firstpage_specials: \prg_do_nothing:
182 }

```

The `__shipout_drop_firstpage_specials:` is run after the shipout has occurred but before the `shipout/afterpage` hook is executed. That is the point where we have to restore the `\ShipoutBox` to its state without the `shipout/firstpage` material.

```

183 \cs_new:Npn \__shipout_drop_firstpage_specials: {
184     \box_set_eq:NN \l_shipout_box \l__shipout_raw_box

```

If there was no such material then `__shipout_run_firstpage_hook:` will have changed the definition to a no-op already. Otherwise this is what we do here.

```

185     \cs_gset_eq:NN \__shipout_drop_firstpage_specials: \prg_do_nothing:
186 }

```

(End definition for `__shipout_add_firstpage_specials:` and `__shipout_drop_firstpage_specials:.`)

`\l__shipout_firstpage_box` The box to hold any firstpage `\specials`.

```

187 \box_new:N \l__shipout_firstpage_box
(End definition for \l__shipout_firstpage_box.)

```

`\g__shipout_lastpage_handled_bool` A boolean to signal if we have already handled the `shipout/lastpage` hook.

```

188 \bool_new:N \g__shipout_lastpage_handled_bool
(End definition for \g__shipout_lastpage_handled_bool.)

```

`__shipout_add_firstpage_material:Nn` This command adds material to the `shipout/firstpage` hook. It is used in `\AtBeginDvi`, etc. The first argument is the command through which it is called. Initially this is ignored but once we are passed the first page it can be used to generate a warning message mentioning the right user command.

```

189 \cs_new:Npn \__shipout_add_firstpage_material:Nn #1#2 {
190   \AddToHook{shipout/firstpage}{#2}
191 }
(End definition for \__shipout_add_firstpage_material:Nn.)

```

`__shipout_get_box_size:N` Store the box dimensions in dimen registers.

Todo: This could/should perhaps be generalized to set height depth and width given an arbitrary box.

```

192 \cs_new:Npn \__shipout_get_box_size:N #1 {
193   \dim_set:Nn \l_shipout_box_ht_dim { \box_ht:N #1 }
194   \dim_set:Nn \l_shipout_box_dp_dim { \box_dp:N #1 }
195   \dim_set:Nn \l_shipout_box_wd_dim { \box_wd:N #1 }
196   \dim_set:Nn \l_shipout_box_ht_plus_dp_dim
197     { \l_shipout_box_ht_dim + \l_shipout_box_dp_dim }
198 }
(End definition for \__shipout_get_box_size:N.)

```

`\l_shipout_box_ht_dim`
`\l_shipout_box_dp_dim`
`\l_shipout_box_wd_dim`
`\l_shipout_box_ht_plus_dp_dim`

And here are the variables set by `__shipout_get_box_size:N`.

```

199 \dim_new:N \l_shipout_box_ht_dim
200 \dim_new:N \l_shipout_box_dp_dim
201 \dim_new:N \l_shipout_box_wd_dim
202 \dim_new:N \l_shipout_box_ht_plus_dp_dim
(End definition for \l_shipout_box_ht_dim and others. These functions are documented on page 826.)

```

`\g__shipout_discard_bool` Indicate whether or not the current page box should be discarded

```

203 \bool_new:N \g__shipout_discard_bool
(End definition for \g__shipout_discard_bool.)

```

`\l__shipout_tmp_box`
`\l__shipout_saved_badness_tl`

We need a box for the background and foreground material and a token register to remember badness settings as we disable them during the buildup below.

```

204 \box_new:N \l__shipout_tmp_box
205 \tl_new:N \l__shipout_saved_badness_tl
(End definition for \l__shipout_tmp_box and \l__shipout_saved_badness_tl.)

```

`_shipout_add_background_box:n` In standard L^AT_EX the shipout box is always a `\vbox` but here we allow for other usage as well, in case some package has its own output routine.

```
206 \cs_new:Npn \_shipout_add_background_box:n #1
207 { \_shipout_get_box_size:N \l_shipout_box
```

But we start testing for a vertical box as that should be the normal case.

```
208 \box_if_vertical:NTF \l_shipout_box
209 {
```

Save current values of `\vfuzz` and `\vbadness` then change them to allow box manipulations without warnings.

```
210 \tl_set:Nx \l__shipout_saved_badness_tl
211 { \vfuzz=\the\vfuzz\relax
212 \vbadness=\the\vbadness\relax }
213 \vfuzz=\c_max_dim
214 \vbadness=\c_max_int
```

Then we reconstruct `\l_shipout_box` ...

```
215 \vbox_set_to_ht:Nnn \l_shipout_box \l_shipout_box_ht_plus_dp_dim
216 {
```

... the material in `#1` is placed into a horizontal box with zero dimensions.

```
217 \hbox_set:Nn \l__shipout_tmp_box
218 { \l__shipout_saved_badness_tl #1 }
219 \box_set_wd:Nn \l__shipout_tmp_box \c_zero_dim
220 \box_set_ht:Nn \l__shipout_tmp_box \c_zero_dim
221 \box_set_dp:Nn \l__shipout_tmp_box \c_zero_dim
```

Then we typeset that box followed by whatever was in `\l_shipout_box` before (unpacked).

```
222 \skip_zero:N \baselineskip
223 \skip_zero:N \lineskip
224 \skip_zero:N \lineskiplimit
225 \box_use:N \l__shipout_tmp_box
226 \vbox_unpack:N \l_shipout_box
```

The `\kern` ensures that the box has no depth which is afterwards explicitly corrected.

```
227 \kern \c_zero_dim
228 }
229 \box_set_ht:Nn \l_shipout_box \l_shipout_box_ht_dim
230 \box_set_dp:Nn \l_shipout_box \l_shipout_box_dp_dim
```

Todo: The whole boxing maneuver looks a bit like overkill to me, but for the moment I leave.

```
231 \l__shipout_saved_badness_tl
232 }
233 {
```

A horizontal box is handled in a similar way. The last case would be a void box in which case we do nothing hence the missing F branch.

```
234 \box_if_horizontal:NT \l_shipout_box
235 {
236 \tl_set:Nx \l__shipout_saved_badness_tl
237 { \hfuzz=\the\hfuzz\relax
238 \hbadness=\the\hbadness\relax }
239 \hfuzz=\c_max_dim
```



```

240         \hbadness=\c_max_int
241         \hbox_set_to_wd:Nnn \l_shipout_box \l_shipout_box_wd_dim
242         {
243             \hbox_set:Nn \l__shipout_tmp_box
244                 { \l__shipout_saved_badness_tl #1 }
245             \box_set_wd:Nn \l__shipout_tmp_box \c_zero_dim
246             \box_set_ht:Nn \l__shipout_tmp_box \c_zero_dim
247             \box_set_dp:Nn \l__shipout_tmp_box \c_zero_dim
248             \box_move_up:nn
249                 \l_shipout_box_ht_dim
250             { \box_use:N \l__shipout_tmp_box }
251             \hbox_unpack:N \l_shipout_box
252         }
253         \l__shipout_saved_badness_tl
254     }
255 }
256 }

```

(End definition for __shipout_add_background_box:n.)

__shipout_add_foreground_box:n Foreground boxes are done in the same way, only the order and placement of boxes has to be done differently.

```

257 \cs_new:Npn \__shipout_add_foreground_box:n #1
258 {
259     \box_if_vertical:NTF \l_shipout_box
260     {
261         \tl_set:Nx \l__shipout_saved_badness_tl
262             { \vfuzz=\the\vfuzz\relax
263               \vbadness=\the\vbadness\relax }
264         \vfuzz=\c_max_dim
265         \vbadness=\c_max_int
266         \vbox_set_to_ht:Nnn \l_shipout_box \l_shipout_box_ht_plus_dp_dim
267         {
268             \hbox_set:Nn \l__shipout_tmp_box
269                 { \l__shipout_saved_badness_tl #1 }
270             \box_set_wd:Nn \l__shipout_tmp_box \c_zero_dim
271             \box_set_ht:Nn \l__shipout_tmp_box \c_zero_dim
272             \box_set_dp:Nn \l__shipout_tmp_box \c_zero_dim
273             \skip_zero:N \baselineskip
274             \skip_zero:N \lineskip
275             \skip_zero:N \lineskiplimit
276             \vbox_unpack:N \l_shipout_box
277             \kern -\l_shipout_box_ht_plus_dp_dim
278             \box_use:N \l__shipout_tmp_box
279             \kern \l_shipout_box_ht_plus_dp_dim
280         }
281         \l__shipout_saved_badness_tl
282         \box_set_ht:Nn \l_shipout_box \l_shipout_box_ht_dim
283         \box_set_dp:Nn \l_shipout_box \l_shipout_box_dp_dim
284     }
285     {
286         \box_if_horizontal:NT \l_shipout_box
287         {
288             \tl_set:Nx \l__shipout_saved_badness_tl

```

```

289         { \hfuzz=\the\hfuzz\relax
290           \hbadness=\the\hbadness\relax }
291       \hfuzz=\c_max_dim
292       \hbadness=\c_max_int
293       \hbox_set_to_wd:Nnn \l_shipout_box \l_shipout_box_wd_dim
294       {
295         \hbox_unpack:N \l_shipout_box
296         \kern -\box_wd:N \l_shipout_box
297         \hbox_set:Nn \l__shipout_tmp_box
298           { \l__shipout_saved_badness_tl #1 }
299         \box_set_wd:Nn \l__shipout_tmp_box \c_zero_dim
300         \box_set_ht:Nn \l__shipout_tmp_box \c_zero_dim
301         \box_set_dp:Nn \l__shipout_tmp_box \c_zero_dim
302         \box_move_up:nn { \box_ht:N \l_shipout_box }
303           { \box_use:N \l__shipout_tmp_box }
304         \kern \box_wd:N \l_shipout_box
305       }%
306       \l__shipout_saved_badness_tl
307     }
308   }
309 }

```

(End definition for `__shipout_add_foreground_box:n`.)

```

\__shipout_init_page_origins:
\c__shipout_horigin_tl
\c__shipout_vorigin_tl

```

Two constants holding the offset of the top-left with respect to the media box.

Setting the constants this way is courtesy of Bruno.

We delay setting the constants to the last possible place as there might be updates in the preamble or even in the `begindocument` hook that affects their setup.

```

310 \cs_new:Npn \__shipout_init_page_origins: {
311   \tl_const:Nx \c__shipout_horigin_tl
312   {
313     \cs_if_exist_use:NTF \pdfvariable { horigin }
314     { \cs_if_exist_use:NF \pdfhorigin { 1in } }
315   }
316   \tl_const:Nx \c__shipout_vorigin_tl
317   {
318     \cs_if_exist_use:NTF \pdfvariable { vorigin }
319     { \cs_if_exist_use:NF \pdfvorigin { 1in } }
320   }

```

After the constants have been set there is no need to execute this command again, in fact it would raise an error, so we redefine it to do nothing.

```

321   \cs_gset_eq:NN \__shipout_init_page_origins: \prg_do_nothing:
322 }

```

(End definition for `__shipout_init_page_origins:`, `\c__shipout_horigin_tl`, and `\c__shipout_vorigin_tl`.)

```
\__shipout_picture_overlay:n
```

Put the argument into a `picture` environment that doesn't take up any size and uses `1pt` for `\unitlength`.

Todo: Could perhaps be generalized as it might be useful elsewhere. For now it is not.

```

323 \cs_new:Npn \__shipout_picture_overlay:n #1 {

```

The very first time this is executed we have to initialize (and freeze) the origins.

```

324   \__shipout_init_page_origins:
325   \kern -\c__shipout_horigin_tl \scan_stop:
326   \vbox_to_zero:n {
327     \kern -\c__shipout_vorigin_tl \scan_stop:
328     \unitlength 1pt \scan_stop:

```

This mimics a simple zero-sized picture environment. The `\hss` is need in case there is horizontal material (without using `\put` with a positive width.

```

329     \hbox_set_to_wd:Nnn \l__shipout_tmp_box \c_zero_dim
330                        { \ignorespaces #1 \hss }
331     \box_set_ht:Nn \l__shipout_tmp_box \c_zero_dim
332     \box_set_dp:Nn \l__shipout_tmp_box \c_zero_dim
333     \box_use:N \l__shipout_tmp_box
334     \tex_vss:D
335   }
336 }

```

(End definition for `__shipout_picture_overlay:n`.)

`__shipout_add_background_picture:n` Put a `picture` env in the background of the shipout box with its reference point in the top-left corner.

```

337 \cs_new:Npn \__shipout_add_background_picture:n #1 {
338   \__shipout_add_background_box:n { \__shipout_picture_overlay:n {#1} }
339 }

```

(End definition for `__shipout_add_background_picture:n`.)

`__shipout_add_foreground_picture:n` Put a `picture` env in the foreground of the shipout box with its reference point in the top-left corner.

```

340 \cs_new:Npn \__shipout_add_foreground_picture:n #1 {
341   \__shipout_add_foreground_box:n { \__shipout_picture_overlay:n {#1} }
342 }

```

(End definition for `__shipout_add_foreground_picture:n`.)

`\shipout_discard:` Request that the next shipout box should be discarded. At the moment this is just setting a boolean, but we may want to augment this behavior that the position of the call is taken into account (in case \LaTeX looks ahead and is not using the position for on the next page).

```

343 \cs_new_protected:Npn \shipout_discard: {
344   \bool_gset_true:N \g__shipout_discard_bool
345 }

```

(End definition for `\shipout_discard:`. This function is documented on page 828.)

3.2 Handling the end of job hook

At the moment this is partly solved by using the existing hooks. But rather than putting the code into these hooks it should be moved to the right place directly as we shouldn't prefill hooks with material unless it needs to interact with other code.

`\g_shipout_readonly_int` We count every shipout activity that makes a page (but not those that are discarded) in order to know how many pages got produced.

`\ReadonlyShipoutCounter`

```

346 \int_new:N \g_shipout_readonly_int

```

For L^AT_EX 2_ε it is available as a command (i.e., a T_EX counter only).

```

347 \cs_new_eq:NN \ReadonlyShipoutCounter \g_shipout_readonly_int

```

(End definition for `\g_shipout_readonly_int` and `\ReadonlyShipoutCounter`. These functions are documented on page 829.)

`\g_shipout_totalpages_int` We count every shipout attempt (even those that are discarded) in this counter. It is not used in the code but may get used in user code.

`\c@totalpages`

```

348 \int_new:N \g_shipout_totalpages_int

```

For L^AT_EX 2_ε this is offered as a L^AT_EX counter so can be easily typeset inside the output routine to display things like “`\thepage/\thetotalpages`”, etc.

```

349 \cs_new_eq:NN \c@totalpages \g_shipout_totalpages_int
350 \cs_new:Npn \thetotalpages { \arabic{totalpages} }

```

(End definition for `\g_shipout_totalpages_int` and `\c@totalpages`. These functions are documented on page 829.)

`\@abspage@last` In `\@abspage@last` record the number of pages from the last run. This is written to the `.aux` and this way made available to the next run. In case there is no `.aux` file or the statement is missing from it we initialize it with the largest possible number in T_EX. We use this as the default because then we are inserting the `shipout/lastpage` on the last page (or after the last page) but not on page 1 for a multipage document.

```

351 \xdef\@abspage@last{\number\maxdimen}

```

(End definition for `\@abspage@last`.)

`\enddocument` Instead of using the hooks `enddocument` and `enddocument/afterlastpage` we add this code to private kernel hooks to be 100% sure when it is executed and to avoid cluttering the hooks with data that is always there.

Inside `\enddocument` there is a `\clearpage`. Just before that we execute this code here. There is a good chance that we are on the last page. Therefore, if we don't know the value from the last run, we assume that the current page is the right one. So we set `\@abspage@last` and as a result the next shipout will run the `shipout/lastpage` code. Of course, if there are floats that still need a placement this guess will be wrong but then rerunning the document will give us the correct value next time around.

`\@kernel@after@enddocument`

```

352 \g@addto@macro \@kernel@after@enddocument {
353   \int_compare:nNnT \@abspage@last = \maxdimen
354     {

```

We use L^AT_EX 2_ε coding as `\@abspage@last` is not an L₃ name.

```

355     \xdef\@abspage@last{ \int_eval:n {\g_shipout_readonly_int + 1} }
356   }
357 }

```

Once the `\clearpage` has done its work inside `\enddocument` we know for sure how many pages this document has, so we record that in the `.aux` file for the next run.

```
358 \g@addto@macro \@kernel@after@enddocument@afterlastpage {
```

There is one special case: If no output is produced then there is no point in a) recording the number as 0 will never match the page number of a real page and b) adding an extra page to run the `shipout/lastpage` is pointless as well (as it would remain forever). So we test for this and run the code only if there have been pages.

```
359 \int_compare:nNf \g_shipout_readonly_int = 0
360 {
```

This ends up in the `.aux` so we use L^AT_EX 2_ε names here.

Todo: This needs an interface for `\nofiles` in `expl3`, doesn't at the moment!

```
361 \if@filesw
362 \iow_now:Nx \@auxout {
363 \gdef\string\@abspage@last {\int_use:N \g_shipout_readonly_int}}
364 \fi
```

But we may have guessed wrongly earlier and we still have to run the `shipout/lastpage` even though there is no page to place it into. If that is the case we make a trivial extra page and put it there. This temporary page will then vanish again on the next run but helps to keep pdf viewers happy.

```
365 \bool_if:Nf \g__shipout_lastpage_handled_bool
366 {
```

However, making this extra page in case the hook is actually empty would be forcing a rerun without any reason, so we check that condition and also check if `\@kernel@after@shipout@lastpage` contains any code. If both are empty we omit the page generation.

```
367 \bool_lazy_and:nnF
368 { \hook_if_empty_p:n {shipout/lastpage} }
369 { \tl_if_empty_p:N \@kernel@after@shipout@lastpage }
370 {
371 \tex_shipout:D\ vbox to\textheight
372 {
373 \hbox:n { \UseHook{shipout/lastpage}
374 \@kernel@after@shipout@lastpage }
375 }
```

This extra page could be totally empty except for the hook content, but to help the user understanding why it is there we put some text into it.

```
375 \__shipout_excuse_extra_page:
376 \null
377 }
```

At this point we also signal to L^AT_EX's endgame that a rerun is necessary so that an appropriate message can be shown on the terminal. We do this by simply defining a command used as a flag and tested in `\enddocument`.

```
378 \cs_gset_eq:NN \@extra@page@added \relax
379 }
380 }
381 }
382 }
```

(End definition for `\enddocument`, `\@kernel@after@enddocument`, and `\@kernel@after@enddocument@afterlastpage`.)

`_shipout_excuse_extra_page:` Say mea culpa ...

```

383 \cs_new:Npn \_shipout_excuse_extra_page: {
384   \vfil
385   \begin{center}
386     \bfseries Temporary~ page!
387   \end{center}
388   \LaTeX{}~ was~ unable~ to~ guess~ the~ total~ number~ of~ pages~
389   correctly.~ ~ As~ there~ was~ some~ unprocessed~ data~ that~
390   should~ have~ been~ added~ to~ the~ final~ page~ this~ extra~
391   page~ has~ been~ added~ to~ receive~ it.
392   \par
393   If~ you~ rerun~ the~ document~ (without~ altering~ it)~ this~
394   surplus~ page~ will~ go~ away,~ because~ \LaTeX{}~ now~ knows~
395   how~ many~ pages~ to~ expect~ for~ this~ document.
396   \vfil
397 }
```

(End definition for `_shipout_excuse_extra_page:.`)

`\PreviousTotalPages` In the preamble before the aux file was read `\PreviousTotalPages` is always zero.

`\@kernel@before@begindocument`

```

398 \def\PreviousTotalPages{0}

In the aux file there should be an update for \@abspage@last recording the number of
pages from the previous run. If not that macro holds the value of \maxdimen. So we test
for it and update \PreviousTotalPages if there was a real value. This should happen
just before the begindocument hook is executed so that the value can be used inside that
hook.

399 \g@addto@macro\@kernel@before@begindocument
400   {\ifnum\@abspage@last<\maxdimen
401     \xdef\PreviousTotalPages{\@abspage@last}\fi}
```

(End definition for `\PreviousTotalPages` and `\@kernel@before@begindocument`. These functions are documented on page 829.)

4 Legacy L^AT_EX 2_ε interfaces

`\DiscardShipoutBox` Request that the next shipout box is to be discarded.

```
402 \cs_new_eq:NN \DiscardShipoutBox \shipout_discard:
```

(End definition for `\DiscardShipoutBox`. This function is documented on page 828.)

`\AtBeginDvi` If we roll forward from an earlier kernel `\AtBeginDvi` is defined so we better not use `\cs_new_protected:Npn` here.

```

403 \cs_set_protected:Npn \AtBeginDvi
404   {\_shipout_add_firstpage_material:Nn \AtBeginDvi}
```

(End definition for `\AtBeginDvi`. This function is documented on page 828.)

`\DebugShipoutsOn`

`\DebugShipoutsOff`

```

405 \cs_new_eq:NN \DebugShipoutsOn \shipout_debug_on:
406 \cs_new_eq:NN \DebugShipoutsOff \shipout_debug_off:
```

(End definition for `\DebugShipoutsOn` and `\DebugShipoutsOff`. These functions are documented on page 830.)

5 Internal commands needed elsewhere

These internal commands use double and triple @ signs so we need to stop getting them translated to the module name.

407 <@@= >

\@expl@@@shipout@add@firstpage@material@@@Nn
 \@expl@@@shipout@add@background@box@@n
 \@expl@@@shipout@add@foreground@box@@n
 \@expl@@@shipout@add@background@picture@@n
 \@expl@@@shipout@add@foreground@picture@@n

Some internals needed elsewhere.

408 \cs_set_eq:NN \@expl@@@shipout@add@firstpage@material@@@Nn
 409 __shipout_add_firstpage_material:Nn
 410 \cs_set_eq:NN \@expl@@@shipout@add@background@box@@n
 411 __shipout_add_background_box:n
 412 \cs_set_eq:NN \@expl@@@shipout@add@foreground@box@@n
 413 __shipout_add_foreground_box:n
 414 \cs_set_eq:NN \@expl@@@shipout@add@background@picture@@n
 415 __shipout_add_background_picture:n
 416 \cs_set_eq:NN \@expl@@@shipout@add@foreground@picture@@n
 417 __shipout_add_foreground_picture:n

(End definition for \@expl@@@shipout@add@firstpage@material@@@Nn and others.)

418 \ExplSyntaxOff
 419 </2ekernel | latexrelease >
 420 <latexrelease>\EndIncludeInRelease

Rolling back here doesn't undefine the interface commands as they may be used in packages without rollback functionality. So we just make them do nothing which may or may not work depending on the code usage.

421 <latexrelease>\IncludeInRelease{0000/00/00}%
 422 <latexrelease> \{shipout\}{Hook management (shipout)}%
 423 <latexrelease>

If we roll forward then \tex_shipout:D may not be defined in which case \shipout does have it original definition and so we must not \let it to something else which is \relax!

424 <latexrelease>\ifcsname tex_shipout:D\endcsname
 425 <latexrelease>\expandafter\let\expandafter\shipout
 426 <latexrelease> \csname tex_shipout:D\endcsname
 427 <latexrelease>\fi
 428 <latexrelease>
 429 <latexrelease>\let \RawShipout\@undefined
 430 <latexrelease>\let \ShipoutBox\@undefined
 431 <latexrelease>\let \ReadonlyShipoutCounter \@undefined
 432 <latexrelease>\let \c@totalpages \@undefined
 433 <latexrelease>\let \thetotalpages \@undefined
 434 <latexrelease>
 435 <latexrelease>\let \DiscardShipoutBox \@undefined
 436 <latexrelease>\let \DebugShipoutsOn \@undefined
 437 <latexrelease>\let \DebugShipoutsOff \@undefined
 438 <latexrelease>
 439 <latexrelease>\DeclareRobustCommand \AtBeginDvi [1]{%
 440 <latexrelease> \global \setbox \@begindvibox
 441 <latexrelease> \vbox{\unvbox \@begindvibox #1}%
 442 <latexrelease>}

```

443 <latexrelease>
444 <latexrelease>\let \AtBeginShipout \@undefined
445 <latexrelease>\let \AtBeginShipoutNext \@undefined
446 <latexrelease>
447 <latexrelease>\let \AtBeginShipoutFirst \@undefined
448 <latexrelease>
449 <latexrelease>\let \ShipoutBoxHeight \@undefined
450 <latexrelease>\let \ShipoutBoxDepth \@undefined
451 <latexrelease>\let \ShipoutBoxWidth \@undefined
452 <latexrelease>

```

We do not undo a substitution when rolling back. As the file support gets undone the underlying data is no longer used (and sufficiently obscure that it should not interfere with existing commands) and properly removing it would mean we need to make the `\undeclare@...` and its support macros available in all earlier kernel releases which is pointless (and actually worse).

```

453 <latexrelease>
454 <latexrelease>\let \AtEndDvi \@undefined

```

We do not reenale a disabled package load when rolling back. As the file support gets undone the underlying data is no longer checked (and sufficiently obscure that it should not interfere with existing commands) and properly removing it would mean we need to make the `\reenable@package@load` command available in all earlier kernel releases which is pointless (and actually worse).

```

455 %\reenable@package@load{atenddvi}
456 <latexrelease>
457 <latexrelease>\EndIncludeInRelease
458 <*2ekernel>

```

6 Package emulation for compatibility

6.1 Package `atenddvi` emulation

`\AtEndDvi` This package has only one public command to simulating it is easy and actually sensible to provide as part of the kernel.

```

459 </2ekernel>
460 <*2ekernel | latexrelease>
461 <latexrelease>\IncludeInRelease{2020/10/01}%
462 <latexrelease> \AtEndDvi}{atenddvi emulation}%
463 \ExplSyntaxOn
464 \cs_new_protected:Npn \AtEndDvi {\AddToHook{shipout/lastpage}}
465 \ExplSyntaxOff

```

As the package is integrate we prevent loading (no need to roll that back):

```

466 \disable@package@load{atenddvi}
467 {\PackageWarning{atenddvi}
468 {Functionality of this package is already\MessageBreak
469 provided by LaTeX.\MessageBreak\MessageBreak
470 It is there no longer necessary to load it\MessageBreak
471 and you can safely remove it.\MessageBreak
472 Found on}}
473 </2ekernel | latexrelease>

```



```

474 \latexrelease\EndIncludeInRelease
475 \latexrelease\IncludeInRelease{0000/00/00}%
476 \latexrelease\let \AtEndDvi \atendddvi emulation}%
477 \latexrelease\let \AtEndDvi \@undefined
478 \latexrelease\EndIncludeInRelease
479 \*2kernel)

(End definition for \AtEndDvi. This function is documented on page 828.)

480 \*2kernel)

```

6.2 Package atbegshi emulation

```

481 \*atbegshi-ltx}
482 \ProvidesPackage{atbegshi-ltx}
483 [2021/01/10 v1.0c
484 Emulation of the original atbegshi^^Jpackage with kernel methods]

```

\AtBeginShipoutBox

```

485 \let \AtBeginShipoutBox \ShipoutBox

(End definition for \AtBeginShipoutBox. This function is documented on page 830.)

```

\AtBeginShipoutInit Compatibility only, we aren't delaying ...

```

486 \let \AtBeginShipoutInit \@empty

(End definition for \AtBeginShipoutInit. This function is documented on page 831.)

```

\AtBeginShipout
\AtBeginShipoutNext

Filling hooks

```

487 \protected \def \AtBeginShipout {\AddToHook{shipout/before}}
488 \protected \def \AtBeginShipoutNext {\AddToHookNext{shipout/before}}

```

(End definition for \AtBeginShipout and \AtBeginShipoutNext. These functions are documented on page 831.)

\AtBeginShipoutFirst

Slightly more complex as we need to know the name of the command under which the shipout/firstpage hook is filled.

```

489 \protected \def \AtBeginShipoutFirst
490 {\@expl@@@shipout@add@firstpage@material@@Nn \AtBeginShipoutFirst}

```

(End definition for \AtBeginShipoutFirst. This function is documented on page 831.)

\AtBeginShipoutDiscard

Just a different name.

```

491 \let \AtBeginShipoutDiscard \DiscardShipoutBox

```

(End definition for \AtBeginShipoutDiscard. This function is documented on page 831.)

\AtBeginShipoutAddToBox

We don't expose them.

\AtBeginShipoutAddToBoxForeground

```

492 \let \AtBeginShipoutAddToBox
493 \@expl@@@shipout@add@background@box@@Nn

```

\AtBeginShipoutUpperLeft

\AtBeginShipoutUpperLeftForeground

```

494 \let \AtBeginShipoutAddToBoxForeground
495 \@expl@@@shipout@add@foreground@box@@Nn

496 \let \AtBeginShipoutUpperLeft
497 \@expl@@@shipout@add@background@picture@@Nn
498 \let \AtBeginShipoutUpperLeftForeground
499 \@expl@@@shipout@add@foreground@picture@@Nn

```

(End definition for `\AtBeginShipoutAddToBox` and others. These functions are documented on page 830.)

`\AtBeginShipoutOriginalShipout` This offers the raw `\shipout` primitive of the engine. A page shipped out with this is not counted by `\ReadonlyShipoutCounter` counter and thus the mechanism to place `\specials` at the very end of the output might fail, etc. It should therefore not be used in new applications but is only provided to allow running legacy code. For new code use the commands provided by the kernel instead.

```
500 \ExplSyntaxOn
501 \cs_new_eq:NN \AtBeginShipoutOriginalShipout \tex_shipout:D
```

(End definition for `\AtBeginShipoutOriginalShipout`. This function is documented on page 831.)

`\ShipoutBoxHeight` `\ShipoutBoxWidth` `\ShipoutBoxDepth` This is somewhat different from the original in `atbegshi` where `\ShipoutBoxHeight` etc. only holds the `\the\ht<box>` value. This may have some implications in some use cases and if that is a problem then it might need changing.

```
502 \cs_new:Npn \ShipoutBoxHeight { \dim_use:N \l_shipout_box_ht_dim }
503 \cs_new:Npn \ShipoutBoxDepth { \dim_use:N \l_shipout_box_dp_dim }
504 \cs_new:Npn \ShipoutBoxWidth { \dim_use:N \l_shipout_box_wd_dim }
505 \ExplSyntaxOff
```

(End definition for `\ShipoutBoxHeight`, `\ShipoutBoxWidth`, and `\ShipoutBoxDepth`.)

```
506 </atbegshi-ltx>
```

If the package is requested we substitute the one above:

```
507 <*2kernel>
508 \declare@file@substitution{atbegshi.sty}{atbegshi-ltx.sty}
509 </2kernel>
```

6.3 Package **everyshi** emulation

This is now directly handled in that package so emulation is not necessary any more.

Rather important :-)

```
510 <@@=>
```

File V

ltoutput.dtx

1 Output Routine

1.1 Floats

The ‘2ekernel’ code ensures that a `\usepackage{autoout1}` is essentially ignored if a ‘full’ format is being used that has the autoload file mode already in the format.

```

1 <defx>\begingroup
2 <defx>\makeatletter
3 <defx>\nfss@catcodes
4 <2ekernel>\expandafter\let\csname ver@autoout1.sty\endcsname\fmtversion

```

Historical L^AT_EX 2.09 comments (not necessarily accurate any more):

```

5 <*2ekernel>
6 \message{output,}

*****
*                               *
*                               *
*****

```

PAGE LAYOUT PARAMETERS

```

\topmargin      : Extra space added to top of page.
@twoside        : boolean.  T if two-sided printing
\oddsidemargin  : IF @twoside = T
                  THEN extra space added to left of odd-numbered
                  pages.
                  ELSE extra space added to left of all pages.
\evensidemargin : IF @twoside = T
                  THEN extra space added to left of even-numbered
                  pages.

\headheight     : height of head
\headsep        : separation between head and text
\footskip       : distance separation between baseline of last
                  line of text and baseline of foot.
                  Note difference between \footSKIP and \headSEP.
\textheight     : height of text on page, excluding head and foot
\textwidth      : width of printing on page
\columnsep      : IF @twocolumn = T
                  THEN width of space between columns
\columnseprule  : IF @twocolumn = T
                  THEN width of rule between columns (0 if none).
\columnwidth    : IF @twocolumn = T
                  THEN (\textwidth - \columnsep)/2
                  ELSE \textwidth
                  It is set by the \twocolumn and
                  \onecolumn commands.

```

- `\@textbottom` : Command executed at bottom of vbox holding text of page (including figures). The `\raggedbottom` command almost `\let's` this to `\vfil` (actually sets it to `\vskip \z@ plus.0001fil`). Should have depth 0pt.
- `\@texttop` : Command executed at top of vbox holding text of page (including figures). Used by letter style; can also be used to produce centered pages. Let to `\relax` by `\raggedbottom` and `\flushbottom`.

Page layout must initialize `\@colht` and `\@colroom` to `\textheight`.

PAGE STYLE PARAMETERS:

- `\floatsep` : Space left between floats.
- `\textfloatsep` : Space between last top float or first bottom float and the text.
- `\topfigrule` : Command to place rule (or whatever) between floats at top of page and text. Executed in inner vertical mode right before the `\textfloatsep` skip separating the floats from the text. Must occupy zero vertical space. (See `\footnoterule`.)
- `\botfigrule` : Same as `\topfigrule`, but put after the `\textfloatsep` skip separating text from the floats at bottom of page.
- `\intextsep` : Space left on top and bottom of an in-text float.
- `\dblfloatsep` : Space between double-column floats.
- `\dbltextfloatsep` : Space between top double-column floats and text.
- `\dblfigrule` : Similar to `\topfigrule`, but for double-column floats.
- `\@fptop` : Glue to go at top of float column – must be 0pt + stretch
- `\@fpsep` : Glue to go between floats in a float column.
- `\@fpbot` : Glue to go at bottom of float column
– must be 0pt + stretch
- `\@dblfpptop`, `\@dblfpsep`, `\@dblfpbot`
: Analogous for double-column float page in two-column format.

FOOTNOTES: As in PLAIN, footnotes use `\insert\footins`.

PAGE LAYOUT SWITCHES AND MACROS

- `@twocolumn` : Boolean. T if two columns per page globally.

PAGE STYLE MACROS AND SWITCHES

```

\@oddhead      : IF @twoside = T
                  THEN macro to generate head of odd-numbered
                  pages.
                  ELSE macro to generate head of all pages.
\@evenhead     : IF @twoside = T
                  THEN macro to generate head of even-numbered
                  pages.
\@oddfoot      : IF @twoside = T
                  THEN macro to generate foot of odd-numbered
                  pages.
                  ELSE macro to generate foot of all pages.
\@evenfoot     : IF @twoside = T
                  THEN macro to generate foot of even-numbered
                  pages.
@specialpage   : boolean. T if current page is to have a special
                  format.
\@specialstyle : If its value is foo then
                  IF @specialpage = T
                  THEN the command \ps@foo is executed to
                  temporarily reset the page style parameters
                  before composing the current page.
                  This command should execute only \def's and
                  \edef's, making only local definitions.

```

FLOAT PLACEMENT PARAMETERS

The following parameters are set by the macro `\@floatplacement`.
When `\@floatplacement` is called,

`\@colht` is the height of the page or column being built. I.e.:

- * For single-column page it equals `\textheight`.
- * For double-column page it equals `\textheight - height`
of double-column floats on page.

Note that some are set globally and some locally:

```

\@topnum      :=G Maximum number of floats allowed on the top of a
                  column.
\@toproom     :=G Maximum amount of top of column devoted to floats—
                  excluding \textfloatsep separation below the floats
                  and \floatsep separation between them. For
                  two-column output, should be computed as a function
                  of \@colht.
\@botnum, \@botroom
              : Analogous to above.
\@colnum      :=G Maximum number of floats allowed in a column,
                  including in-text floats.
\@textmin     :=L Minimum amount of text (excluding footnotes) that
                  must appear on a text page.
                %% 27 Sep 85 : made local to
                %% \@addtocurcol and \@addtonextcol
                It is now also used locally in processing double
                floats.

```

`\@fpmin` :=L Minimum height of floats in a float column.

The macro `\@dblfloatplacement` sets the following parameters.

`\@dbltopnum` :=G Maximum number of double-column floats allowed at the top of a two-column page.

`\@dbltoproom` :=G Maximum height of double-column floats allowed at top of two-column page.

`\@fpmin` :=L Minimum height of floats in a float column.

It should also perform the following local assignments where necessary – i.e., where the new value differs from the old one:

`\@fptop` :=L `\@dblfpptop`

`\@fpsep` :=L `\@dblfpsep`

`\@fpbot` :=L `\@dblfpbot`

OUTPUT ROUTINE VARIABLES

`\@colht` : The total height of the current column. In single column style, it equals `\textheight`. In two-column style, it is `\textheight` minus the height of the double-column floats on the current page. MUST BE INITIALIZED TO `\textheight`.

`\@colroom` : The height available in the current column for text and footnotes. It equals `\@colht` minus the height of all floats committed to the top and bottom of the current column.

`\@textfloatsheight` : The total height of in-text floats on the current page.

`\footins` : Footnote insertion number.

`\@maxdepth` : Saved value of TeX's `\maxdepth`. Must be set when any routine sets `\maxdepth`.

CALLING THE OUTPUT ROUTINE

The output routine is called either by TeX's normal page-breaking mechanism, or by a macro putting a penalty $<$ or $= -10000$ in the output list. In the latter case, the penalty indicates why the output routine was called, using the following code.

penalty	reason
-10000	<code>\pagebreak</code> <code>\newpage</code>
-10001	<code>\clearpage</code> (<code>\penalty -10000 \vbox{}</code> <code>\penalty -10001</code>)
-10002	float insertion, called from horizontal mode
-10003	float insertion, called from vertical mode.
-10004	float insertion.

Note: A float or marginpar puts the following sequence in the output list: (i) a penalty of -10004,
(ii) a null `\vbox`
(iii) a penalty of -10002 or -10003.

This solves two special problems:

1. If the float comes right after a `\newpage` or `\clearpage`, then the first penalty is ignored, but the second one invokes the output routine.
2. If there is a split footnote on the page, the second 'page' puts out the rest of the footnote.

THE OUTPUT ROUTINE

FUNCTIONS USED IN THE OUTPUT ROUTINE:

`\@outputpage` : Produces an output page with the contents of box `\@outputbox` as the text part.

Also sets `\@colht :=G \textheight`.

The page style is determined as follows.

IF `@thispagestyle = true`

THEN use `\thispagestyle` style

ELSE use ordinary page style.

`\@tryfcolumn\FLIST` : Tries to form a float column composed of floats from `\FLIST` (if nonempty) with the following parameters:

`\@colht` : height of box

`\@fpmin` : minimum height of floats in the box

`\@fpsep` : interfloat space

`\@fptop` : glue at top of box

`\@fpbot` : glue at bottom of box.

If it succeeds, then it does the following:

* `\@outputbox :=L` the composed float box.

* `@fcolmade :=G true`

* `\FLIST :=G \FLIST` - floats put in box

* `\@freelist :=G \@freelist +` floats put in box

If it fails, then:

* `@fcolmade :=G false`

NOTE: BIT MUST BE A SINGLE TOKEN!

`\@makefcolumn \FLIST` : Same as `\@tryfcolumn` except that it fails to make a float column only if `\FLIST` is empty. Otherwise, it makes a float column containing at least the first box in `\FLIST`, disregarding `\@fpmin`.

`\@startcolumn` :

Calls `\@tryfcolumn\@deferlist`. If `\@tryfcolumn` returns with (globally set) `@fcolmade = false`, then:

* Globally sets `\@toplist` and `\@botlist` to floats

from `\@deferlist` to go at top and bottom of column, deleting them from `\@deferlist`. It does this using `\@colht` as the total height, the page style parameters `\@floatsep` and `\@textfloatsep`, and the float placement parameters `\@topnum`, `\@toproom`, `\@botnum`, `\@botroom`, `\@colnum` and `\textfraction`.

- * Globally sets `\@colroom` to `\@colht` minus the height of the added floats.

`\@startdblcolumn` :

Calls `\@tryfcolumn\@dbldeferlist{8}`. If `\@tryfcolumn` returns with (globally set) `@fcolmade = false`, then:

- * Globally sets `\@dbltoplist` to floats from `\@dbldeferlist` to go at top and bottom of column, deleting them from `\@dbldeferlist`. It does this using `\textheight` as the total height, and the parameters `\@dblfloatsep`, etc.
- * Globally sets `\@colht` to `\textheight` minus the height of the added floats.

`\@combinefloats` : Combines the text from box

`\@outputbox` with the floats from `\@toplist` and `\@botlist`, putting the new box in `\@outputbox`. It uses `\floatsep` and `\textfloatsep` for the appropriate separations. It puts the elements of `\@TOPLIST` and `\@BOTLIST` onto `\@freelist`, and makes those lists null.

`\@makecol` : Makes the contents of `\box255` plus the accumulated footnotes, plus the floats in `\@toplist` and `\@botlist`, into a single column of height `\@colht` (unless the page height has been locally changed), which it puts into box `\@outputbox`. It puts boxes in `\@midlist` back onto `\@freelist` and restores `\maxdepth`.

`\@opcol` : Outputs a column whose text is in box `\@outputbox`

If `@twocolumn = false`, then it calls `\@outputpage`, sets `\@colht :=G \textheight`, and calls `\@floatplacement`.

If `@twocolumn = true`, then:

If `@firstcolumn = true`, then it puts box `\@outputbox` into `\@leftcolumn` and sets `@firstcolumn :=G false`.

If `@firstcolumn = false`, then it puts out the current two-column page, any possible two-column float pages, and determines `\@dbltoplist` for the next page.

USER COMMANDS THAT CALL OR AFFECT THE OUTPUT ROUTINE

```

\newpage == BEGIN \par\vfil\penalty -10000 END

\clearpage == BEGIN \newpage
                  \write -1{}      % Part of hack to make sure no
                  \vbox{}          % \write's get lost.
                  \penalty -10001
                  END

\cleardoublepage == BEGIN \clearpage
                      if @twoside = true and c@page is even
                      then \hbox{} \newpage fi
                      END

\twocolumn[BOX] : starts a new page, changing to twocolumn setting
                  and puts BOX in a parbox of width \textwidth across the top.
                  Useful for full-width titles for double-column pages.
                  SURPRISE: The stretch from \@dbltextfloatsep will be inserted
                           between the BOX and the top of the two columns.

```

FLOAT-HANDLING MECHANISMS

The float environment obtains an insertion number B from the `\@freelist` (see below for a description of list manipulation), puts the float into box B and sets `\count B` to a FLOAT SPECIFIER. For a normal (not double-column) float, it then causes a page break in one of the following two ways:

- In outer hmode: `\vadjust{\penalty -10002}`
- In vmode : `\penalty -10003`.

For a double-column float, it puts B onto the `\@dbldeferlist`.

The float specifier has two components:

- * A PLACEMENT SPECIFICATION, describing where the float may be placed.
- * A TYPE, which is a power of two—e.g., figures might be type 1 floats, tables type 2 floats, programs type 4 floats, etc.

The float specifier is encoded as follows, where bit 0 is the least significant bit.

Bit	Meaning
0	1 iff the float may go where it appears in the text.
1	1 iff the float may go on the top of a page.
2	1 iff the float may go on the bottom of a page.
3	1 iff the float may go on a float page.
4	1 unless the PLACEMENT includes a !
5	1 iff a type 1 float

6 1 iff a type 2 float
etc.

A negative float specifier is used to indicate a marginal note.

MACROS AND DATA STRUCTURES FOR PROCESSING FLOATS

A FLOAT LIST consisting of the floats in boxes `\boxa ... \boxN` has the form:

```
\@elt \boxa ... \@elt \boxN
```

where `\boxI` is defined by

```
\newinsert\boxI
```

Normally, `\@elt` is `\let` to `\relax`. A test can be performed on the entire float list by locally `\def`'ing `\@elt` appropriately and executing the list.

This is a lot more efficient than looping through the list.

The following macros are used for manipulating float lists.

```
\@next \CS \LIST {NONEMPTY}{EMPTY} == %% NOTE: ASSUME \@elt
= \relax
  BEGIN  assume that \LIST == \@elt \B1 ... \@elt \Bn
        if n = 0
          then  EMPTY
        else  \CS      :=L \B1
              \LIST :=G \@elt \B2 ... \@elt \Bn
              NONEMPTY
        fi
  END
```

`\@bitor\NUM\LIST` : Globally sets switch `@test` to the disjunction for all `I` of bit `log2 \NUM` of the float specifiers of all the floats in `\LIST`.
I.e., `@test` is set to true iff there is at least one float in `\LIST` having bit `log2 \NUM` of its float specifier equal to 1.

Note: `log2 [(\count I)/32]` is the bit number corresponding to the type of float `I`. To see if there is any float in `\LIST` having the same type as float `I`, you run `\@bitor` with
`\NUM = [(\count I)/32] * 32`.

```
\@bitor\NUM\LIST ==
BEGIN
  @test :=G false
  { \@elt \CTR == if \NUM <> 0 then
                    if \count\CTR / \NUM is odd
                    then  @test := true          fi fi
```

```

\LIST
}
END

```

`\@cons\LIST\NUM` : Globally sets `\LIST := \LIST * \@elt \NUM`

```

\@cons\LIST\NUM ==
  BEGIN { \@elt == \relax
    \LIST :=G \LIST \@elt \NUM
  }

```

BOX LISTS FOR FLOAT-PLACEMENT ALGORITHMS

```

\@freelist      : List of empty boxes for placing new floats.
\@toplist       : List of floats to go at top of current column.
\@midlist       : List of floats in middle of current column.
\@botlist       : List of floats to go at bottom of current column.
\@deferlist     : List of floats to go after current column.
\@dbltoplist    : List of double-col. floats to go at top of current
                  page.
\@dbldeferlist  : List of double-column floats to go on subsequent
                  pages.

```

FLOAT-PLACEMENT ALGORITHMS

`\@addtobot` : Tries to put insert `\@currbox` on `\@botlist`.

Called only when:

- * `\ht BOX < \@colroom`
- * type of `\@currbox` not on `\@deferlist`
- * `\@colnum > 0`
- * `@insert = false`

If it succeeds, then:

- * sets `@insert true`
- * decrements `\@botroom` by `\ht BOX`
- * decrements `\@botnum` and `\@colnum` by 1
- * decrements `\@colroom` by `\ht BOX + either \floatsep`
or `\textfloatsep`, as appropriate.
- * sets `\maxdepth` to 0pt

`\@addtotoporbot` : Tries to put insert `\@currbox` on `\@toplist` or `\@botlist`.

Called only under same conditions as `\@addtobot`.

If it succeeds, then:

- * sets `@insert true`
- * decrements `\@toproom` or `\@botroom` by `\ht BOX`
- * decrements `\@colnum` and either `\@topnum` or `\@botnum` by 1
- * decrements `\@colroom` by `\ht BOX + \floatsep`

or `\textfloatsep`, as appropriate.

`\@addtocurcol` : Tries to add `\@currbox` to current column, setting
 `@insert` true if it succeeds, false otherwise.
 It will add `\@currbox` to top only if bit 0 of
 `\count \@currbox` is 0, and to the bottom only if
 bit 0 = 0 or an earlier float of the same type is
 put on the bottom.
 If the float is put in the text, then
 `\penalty\interlinepenalty` is put
 right after the float, before the following `\vskip`,
 and `\outputpenalty :=L 0`.

`\@addtonextcol` : Tries to add `\@currbox` to the next column, setting
 `@insert` true if it succeeds, false otherwise.

`\@addtodblcol` : Tries to add `\@currbox` to the next double-column page,
 adding it to `\@dbltoplist` if it succeeds and
 `\@dbldeferlist` if it fails.

```
\@addmarginpar ==
BEGIN
  if \@currlist nonempty
    then remove \@marbox from \@currlist
    add \@marbox and \@currbox to \@freelist
    %% NOTE: \@currbox = left box
  else LaTeX error: ? %% shouldn't happen
  fi
  \@tempcnta := 1      %% 1 = right, -1 = left
  if @twocolumn = true
    then if @firstcolumn = true
      then \@tempcnta := -1
    fi
    else if @mparswitch = true
      then if count0 odd
        else \@tempcnta := -1
      fi
    fi
    if @reversemargin = true
      then \@tempcnta := -\@tempcnta
    fi
  fi
  if \@tempcnta < 0 then \box\@marbox :=G \box\@currbox
  fi
  \@tempdima :=L maximum(\@mparbottom - \@pageht
                        + ht of \@marbox, 0)
  if \@tempdima > 0 then LaTeX warning: 'marginpar moved' fi
  \@mparbottom :=G \@pageht + \@tempdima + depth of \@marbox
                + \marginparpush
```

```

\@tempdima :=L \@tempdima - ht of \@marbox
\box\@marbox :=G \box\@currbox
\hbox{ \vbox { \vskip \@tempdima
\box\@marbox
}
height of \@marbox :=G depth of \@marbox :=G 0
\kern -\@pagedp
\nointerlineskip
\hbox{ if @tempcnta > 0 then \hskip \columnwidth
\hskip \marginparsep
else \hskip -\marginparsep
\hskip -\marginparwidth
fi
\box\@marbox \hss
}
\nobreak
\nointerlineskip
\hbox{\vrule height 0 width 0 depth \@pagedp}
END

```

Floats and marginpars add a lot of dead cycles.
End of historical L^AT_EX 2.09 comments.

```

7 \maxdeadcycles = 100
8 \let\@elt\relax
9 \def\@next#1#2#3#4{\ifx#2\@empty #4\else
10 \expandafter\@xnext #2\@#1#2#3\fi}
11 \def\@xnext \@elt #1#2\@#3#4{\def#3{#1}\gdef#4{#2}}
12 \def\@testfalse{\global\let@if@test\iffalse}
13 \def\@testtrue {\global\let@if@test\iftrue}
14 \@testfalse
15 \def\@bitor#1#2{\@testfalse {\let\@elt\@xbitor
16 \@tempcnta #1\relax #2}}

```

RmS 91/11/22: Added test for \count#1 = 0. Suggested by Chris Rowley.

```

17 \def\@xbitor #1{\@tempcntb \count#1
18 \ifnum \@tempcnta =\z@
19 \else
20 \divide\@tempcntb\@tempcnta
21 \ifodd\@tempcntb \@testtrue\fi
22 \fi}

```

DEFINITION OF FLOAT BOXES:

```

23 </2ekernel>
24 <latexrelease>\IncludeInRelease{2015/10/01}%
25 <latexrelease> \bx@ZZ{Extended float list}%
26 <*2ekernel|latexrelease>
27 \let\@elt\newinsert
28 <*2ekernel>
29 \def\@freelist{%

```

```

30 \@elt\bx@A\@elt\bx@B\@elt\bx@C\@elt\bx@D\@elt\bx@E
31 \@elt\bx@F\@elt\bx@G\@elt\bx@H\@elt\bx@I\@elt\bx@J
32 \@elt\bx@K\@elt\bx@L\@elt\bx@M\@elt\bx@N
33 \@elt\bx@O\@elt\bx@P\@elt\bx@Q\@elt\bx@R}
34 \@freelist
35 \</2ekernel>
36 \ifx\numexpr\@undefined\else
37 \def\reserved@a{%
38 \@elt\bx@S\@elt\bx@T\@elt\bx@U\@elt\bx@V
39 \@elt\bx@W\@elt\bx@X\@elt\bx@Y\@elt\bx@Z
40 \@elt\bx@AA\@elt\bx@BB\@elt\bx@CC\@elt\bx@DD\@elt\bx@EE
41 \@elt\bx@FF\@elt\bx@GG\@elt\bx@HH\@elt\bx@II\@elt\bx@JJ
42 \@elt\bx@KK\@elt\bx@LL\@elt\bx@MM\@elt\bx@NN
43 \@elt\bx@OO\@elt\bx@PP\@elt\bx@QQ\@elt\bx@RR
44 \@elt\bx@SS\@elt\bx@TT\@elt\bx@UU\@elt\bx@VV
45 \@elt\bx@WW\@elt\bx@XX\@elt\bx@YY\@elt\bx@ZZ}
46 \reserved@a
47 \def\@elt{\noexpand\@elt\noexpand}
48 \edef\@freelist{\@freelist\reserved@a}
49 \fi
50 \let\reserved@a\relax
51 \let\@elt\relax
52 \</2ekernel | latexrelease>
53 \<latexrelease>\EndIncludeInRelease
54 \<latexrelease>\IncludeInRelease{0000/00/00}%
55 \<latexrelease> \@elt\bx@ZZ}{Extended float list}%
56 \<latexrelease>\def\@freelist{%
57 \<latexrelease> \@elt\bx@A\@elt\bx@B\@elt\bx@C\@elt\bx@D\@elt\bx@E
58 \<latexrelease> \@elt\bx@F\@elt\bx@G\@elt\bx@H\@elt\bx@I\@elt\bx@J
59 \<latexrelease> \@elt\bx@K\@elt\bx@L\@elt\bx@M\@elt\bx@N
60 \<latexrelease> \@elt\bx@O\@elt\bx@P\@elt\bx@Q\@elt\bx@R}
61 \<latexrelease> \insscount=234
62 \<latexrelease>\EndIncludeInRelease
63 \<*2ekernel>

64 \gdef\@toplist{}
65 \gdef\@botlist{}
66 \gdef\@midlist{}
67 \gdef\@currlist{}
68 \gdef\@deferlist{}
69 \gdef\@dbltoplist{}

```

The new algorithm stores page wide floats together with column floats in a single \@deferlist list. We keep \@dbldeferlist initialised as empty so that packages that are testing for deferred floats can use the same code for old or new float handling.

```

70 \gdef\@dbldeferlist{}

PAGE LAYOUT PARAMETERS

71 \newdimen\topmargin
72 \newdimen\oddsidemargin
73 \newdimen\evensidemargin
74 \let\@themargin=\oddsidemargin
75 \newdimen\headheight
76 \newdimen\headsep
77 \newdimen\footskip

```

```

78 \newdimen\textheight
79 \newdimen\textwidth
80 \newdimen\columnwidth
81 \newdimen\columnsep
82 \newdimen\columnseprule
83 \newdimen\marginparwidth
84 \newdimen\marginparsep
85 \newdimen\marginparpush

```

\AtBeginDvi We use a box register in which to put stuff that must appear before anything else in the .dvi file.

The stuff in the box should not add any typeset material to the page when it is unboxed.

This interface is no longer used. Instead a new one is inside `ltshipout.dtx`. We only keep the box in case some old code refers to it directly (or we do some rollback).

```

86 \newbox\@begindvibox
87 %\DeclareRobustCommand \AtBeginDvi [1]{%
88 % \global \setbox \@begindvibox
89 % \vbox{\unvbox \@begindvibox #1}%
90 %}

```

(End definition for \AtBeginDvi and \@begindvibox. These functions are documented on page 828.)

\@maxdepth This is not the right place to set this; it needs to be set in a class/style file when `\maxdepth` is set.

Also, many settings to `\maxdepth` should be to `\@maxdepth`, probably?

```

91 \newdimen\@maxdepth
92 \@maxdepth = \maxdepth

```

(End definition for \@maxdepth.)

\paperheight New `\paper...` registers.

```

93 \newdimen\paperheight
94 \newdimen\paperwidth

```

(End definition for \paperheight and \paperwidth.)

\if@insert Local switches first:

```

95 \newif \if@insert

```

\if@specialpage These should definitely be global:

```

96 \newif \if@fcolmade
97 \newif \if@specialpage \@specialpagefalse

```

\if@twocolumn These should be global but are not always set globally in other files.

```

98 \newif \if@firstcolumn \@firstcolumntrue
99 \newif \if@twocolumn \@twocolumnfalse

```

Not sure about these: two questions. Should things which must apply to a whole document be local or global (they probably should be ‘preamble only’ commands)? Are these three such things?

```

100 \newif \if@twoside \@twosidefalse
101 \newif \if@reversemargin \@reversemarginfalse
102 \newif \if@mparswitch \@mparswitchfalse

```

This counter has been imported from ‘multicol’.

```
103 \newcount \col@number
104 \col@number \@ne
```

(End definition for \if@insert and others.)

Historical L^AT_EX 2.09 comments (not necessarily accurate any more):

INTERNAL REGISTERS

```
105 \newcount\@topnum
106 \newdimen\@toproom
107 \newcount\@dbltopnum
108 \newdimen\@dbltoproom
109 \newcount\@botnum
110 \newdimen\@botroom
111 \newcount\@colnum
112 \newdimen\@textmin
113 \newdimen\@fpmin
114 \newdimen\@colht
115 \newdimen\@colroom
116 \newdimen\@pageht
117 \newdimen\@pagedp
118 \newdimen\@mparbottom \@mparbottom\z@
119 \newcount\@currtype
120 \newbox\@outputbox
121 \newbox\@leftcolumn
122 \newbox\@holdpg

123 \def\@thehead{\@oddhead} % initialization
124 \def\@thefoot{\@oddfoot}
```

End of historical L^AT_EX 2.09 comments.

\clearpage The tests at the beginning are an experimental attempt to avoid a completely empty page after a \twocolumn[...]. This prevents the text from the argument vanishing into a float box, never to be seen again. We hope that it does not produce wrong formatting in other cases.

```
125 \def\clearpage{%
126   \ifvmode
127     \ifnum \@dbltopnum =\m@ne
128       \ifdim \pagetotal <\topskip
129         \hbox{}%
130       \fi
131     \fi
132   \fi
133   \newpage
134   \write\m@ne{}%
135   \vbox{}%
136   \penalty -\@Mi
137 }
```

(End definition for \clearpage.)

`\cleardoublepage`

```
138 \def\cleardoublepage{\clearpage\if@twoside \ifodd\c@page\else
139   \hbox{}\newpage\if@twocolumn\hbox{}\newpage\fi\fi\fi}
140 \endkernel
```

(End definition for \cleardoublepage.)

`\onecolumn`

```
141 \endkernel | fltrace)
142 \def\onecolumn{%
143   \clearpage
144   \global\columnwidth\textwidth
145   \global\hsize\columnwidth
146   \global\linewidth\columnwidth
147   \global\@twocolumnfalse
148   \col@number \@one
149   \@floatplacement}
```

(End definition for \onecolumn.)

`\newpage` The two checks at the beginning ensure that an item label or run-in section title immediately before a `\newpage` get printed on the correct page, the one before the page break.

All three tests are largely to make error processing more robust; that is why they all reset the flags explicitly, even when it would appear that this would be done by a `\leavevmode`.

```
150 \endkernel | fltrace)
151 \latexrelease\IncludeInRelease{2017/04/15}%
152 \latexrelease{\newpage}{Check depth of page}%
153 \endkernel | latexrelease | fltrace)
154 \def \newpage {%
155   \if@noskipsec
156     \ifx \@nodocument\relax
157       \leavevmode
158       \global \@noskipsecfalse
159     \fi
160   \fi
161   \if@inlabel
162     \leavevmode
163     \global \@inlabelfalse
164   \fi
165   \if@nobreak \@nobreakfalse \everypar{}\fi
166   \par
```

The `\vfil` at the end of the macro before the break penalty will normally result in the page being run short, even with `\flushbottom` in effect (in contrast to the behavior of `\pagebreak`). However, if there is some explicit stretch on the page, say, a `\vfill`, it has the undesired side-effect, that the last line will not align at its baseline if it contains characters going below the baseline, as the value of `\prevdepth` is no longer taken into account by $\mathrm{T\!E\!X}$. So we back up by that amount (or by `\maxdepth` if it is really huge), to mimic the normal behavior without the `\newpage`.

```
167   \ifdim\prevdepth>\z@
168     \vskip -%
169     \ifdim\prevdepth>\maxdepth
```

```

170         \maxdepth
171     \else
172         \prevdepth
173     \fi
174 \fi
175 \vfil
176 \penalty -\@M}
177 </2ekernel | latexrelease | fltrace>
178 <latexrelease>\EndIncludeInRelease
179 <latexrelease>\IncludeInRelease{0000/00/00}%
180 <latexrelease>         {\newpage}{Check depth of page}%
181 <latexrelease>\def \newpage {%
182 <latexrelease> \if@noskipsec
183 <latexrelease> \ifx \@nodocument\relax
184 <latexrelease> \leavevmode
185 <latexrelease> \global \@noskipsecfalse
186 <latexrelease> \fi
187 <latexrelease> \fi
188 <latexrelease> \if@inlabel
189 <latexrelease> \leavevmode
190 <latexrelease> \global \@inlabelfalse
191 <latexrelease> \fi
192 <latexrelease> \if@nobreak \@nobreakfalse \everypar{}\fi
193 <latexrelease> \par
194 <latexrelease> \vfil
195 <latexrelease> \penalty -\@M}
196 <latexrelease>\EndIncludeInRelease
197 <*2ekernel | fltrace>

```

(End definition for \newpage.)

\@emptycol It may be better to use an invisible rule rather than an empty box here.

```

198 \def \@emptycol {\vbox{}\penalty -\@M}

```

(End definition for \@emptycol.)

\twocolumn There are several bug fixes to the two-column stuff here.
\@topnewpage

```

199 \def \twocolumn {%
200     \clearpage
201     \global\columnwidth\textwidth
202     \global\advance\columnwidth-\columnsep
203     \global\divide\columnwidth\tw@
204     \global\hsize\columnwidth
205     \global\linewidth\columnwidth
206     \global\@twocolumntrue
207     \global\@firstcolumntrue
208     \col@number \tw@

```

There is no reason to put a \@dblfloatplacement here since \@topnewpage ignores these settings. The \@floatplacement is needed in case this comes after some changes.

```

209     \ifnextchar [\@topnewpage\@floatplacement
210 }

```

Note that here, getting a box from the freelist can assume success since this comes just after a `\clearpage`.

```

211 \long\def \@topnewpage [#1]{%
212   \nodocument
213   \@next\@currbox\@freelist{}{}%
214   \global \setbox\@currbox
215     \color@vbox
216     \normalcolor
217     \vbox {%
218       \hsize\textwidth
219       \@parboxrestore
220       \col@number \@ne
221       #1%
222       \vskip -\dbltextfloatsep
223     }%
224   \color@endbox

```

Added size test and warning message; perhaps we should use an error message.

```

225   \ifdim \ht\@currbox>\textheight
226     \ht\@currbox \textheight
227   \fi

```

This next line is not essential but it is more robust to make this value non-zero, in case of weird errors.

This next bit is what is needed from `\@addtodblcol`, plus some extra checks for error trapping.

```

228   \global \count\@currbox \tw@
229   \@tempdima -\ht\@currbox
230   \advance \@tempdima -\dbltextfloatsep
231   \global \advance \@colht \@tempdima
232   \ifx \@dbltoplist \@empty
233   \else
234     \@latex@error{Float(s) lost}\@ehb
235     \let \@dbltoplist \@empty
236   \fi
237   \@cons \@dbltoplist \@currbox

```

This setting of `\@dbltopnum` is used only to change the typesetting in `\@combinedblfloats`.

```

238   \global \@dbltopnum \m@ne
239   \if@trace
240     \fl@trace{dbltopnum set to -1 (= \the \@dbltopnum) (topnewpage)}%
241   \fi

```

At points such as this we need to check that there is still a minimal amount of room left on the page; this uses an arbitrary small value at present; but note that this value is larger than that used when checking that page is too full of normal floats.

If there is little room left we just force a page-break, OK? This involves producing two empty columns. The second empty column may be produced by `\output`, in which case an extra, misleading, warning will be generated, OK? (This happens only when there is too little room left on the page for any float.) Otherwise (i.e. if the size is such that it is allowed as a normal float) the extra `\@emptycol` will be invoked in the second column by the conditional code guarded by the `\if@firstcolumn` test.

I now think that the cut-off point here should be `3\baselineskip`, but we make it a bit less so that 3 lines of text will be allowed, OK?

Since this happens only when there is nothing on the page but the ‘top-box’, the empty box should not cause any problem other than some overfull box messages, which is not entirely misleading.

Here we need two page-ends since both columns need to be empty.

```

242 \ifdim \@colht<2.5\baselineskip
243   \@latex@warning@no@line {Optional argument of \noexpand\twocolumn
244     too tall on page \thepage}%
245   \@emptycol
246   \if@firstcolumn
247   \else
248     \@emptycol
249   \fi
250 \else
251   \global \vsize \@colht
252   \global \@colroom \@colht
253   \@floatplacement
254 \fi
255 }
```

(End definition for \twocolumn and \@topnewpage.)

`\output` This needs some small adjustments. We cannot guarantee that the float mechanism will
`\@specialoutput` interact correctly with this stuff, but that mechanism does not always work properly with footnotes already.

RmS 91/09/29:

added reset of `\par` to the output routine. This avoids problems when the output routine is called within a list where `\par` may be a no-op.

```

256 \output {%
257   \let \par \@@par
258   \ifnum \outputpenalty<-\@M
259     \@specialoutput
260   \else
261     \@makecol
262     \@opcol
```

Moved to `\@opcol: \@floatplacement`.

```

263   \@startcolumn
```

This loop could be replaced by an `\expandafter` tail recursion in `\@startcolumn`.

```

264   \@whilesw \if@fcolmade \fi
265   {%
266   < *trace>
267     \fl@trace{PAGE: float \if@twocolumn column \else page \fi
268       completed}%
269   < /trace>
270   \@opcol\@startcolumn}%
271   \fi
272   \ifnum \outputpenalty>-\@Miv
```

At points such as this we need to check that there is still a minimal amount of room left on the page; this uses an arbitrary small value at present. If there is little room left we just force a page-break, OK?

This bit is essential only if a float has just been processed so maybe it should be moved; but this is the natural place at which to set the vsize and a test would need to be done anyway. A check has been added to ensure that there really has been a change in the value of `\@colroom`.

Since this happens only when there is nothing on the page but floats, the empty box should not cause any problem other than some overfull box messages, which is not entirely misleading.

The twocolumn case does not need any extra code here since this is the `\output` itself; in the second column there will still not be enough room left so `\@emptycol` will be executed again when the OR is called by the page builder when it gets to the penalty inserted by the first execution. (The page-builder is never invoked whilst the OR is being executed since it builds a inner vlist; thus any conditional code for the two-column case within `\output` may not get executed with the correct value of `\if@firstcolumn`.

```

273   \ifdim \@colroom<1.5\baselineskip
274     \ifdim \@colroom<\textheight
275       \@latex@warning@no@line {Text page \thepage\space
276                               contains only floats}%
277       \@emptycol
278   %       \if@twocolumn
279   %       \if@firstcolumn
280   %       \else
281   %       \@emptycol
282   %       \fi
283   %       \fi
284   \else
285     \global \vsize \@colroom
286   \fi
287   \else
288     \global \vsize \@colroom
289   \fi
290   \else
291     \global \vsize \maxdimen
292   \fi
293 }

```

Historical L^AT_EX 2.09 comments (not necessarily accurate any more):

CHANGES TO `\@specialoutput`:

* `\penalty\z@` changed to `\penalty\interlinepenalty` so `\samepage` works properly with figure and table environments.

(Changed 23 Oct 86)

* Definition of `\@specialoutput` changed 26 Feb 88 so `\@pageht` and `\@pagedp` aren't changed for a marginal note.

(Change suggested by Chris Rowley.)

End of historical L^AT_EX 2.09 comments.

```

294 \gdef\@specialoutput{%
295   \ifnum \outputpenalty>-\@Mii
296     \@docclearpage

```

```

297 \else
298 \ifnum \outputpenalty<-\@Miii
299 \ifnum \outputpenalty<-\@MM \deadcycles \z@ \fi
300 \global \setbox\@holdpg \vbox {\unvbox\@cclv}%
301 \else

```

Note that `\boxmaxdepth` should not be set here since we wish to record the natural depth of the holdpg box.

This is changed so as to not lose anything, such as writes and marks, which may get into box 255 and should be returned to the list. This should only happen when the first penalty in the mechanism is discarded and therefore `\@holdpg` should always be void in this case. This can happen because a penalty is discarded whenever there is no box on the list.

It was just: `\setbox\@tempboxa \box \@cclv`.

The last box which is removed is the box put there by the double-penalty mechanism. The `\unskip` then removes the `\topskip` which is put there since the box is the first on the page.

```

302 \global \setbox\@holdpg \vbox{%
303 \unvbox\@holdpg
304 \unvbox\@cclv

```

We must now remove the box added by the float mechanism and the `\topskip` glue therefore added above it by T_EX.

```

305 \setbox\@tempboxa \lastbox
306 \unskip
307 }%

```

These two are needed as separate dimensions only by `\@addmarginpar`; for other purposes we put the whole size into `\@pageht` (see below).

```

308 \@pagedp \dp\@holdpg
309 \@pageht \ht\@holdpg
310 \unvbox \@holdpg
311 \@next\@currbox\@currlist{%
312 \ifnum \count\@currbox>\z@

```

Putting the whole size into `\@pageht` (see above).

```

313 \advance \@pageht \@pagedp
314 \ifvoid\footins \else
315 \advance \@pageht \ht\footins
316 \advance \@pageht \skip\footins
317 \advance \@pageht \dp\footins
318 \fi
319 \ifvbox \@kludgeins

```

We want to make the adjustment due to this insert only if the non-star form is used. The *-form will probably not work with floats, but maybe it still could make some adjustment here even so?

```

320 \ifdim \wd\@kludgeins=\z@
321 \advance \@pageht \ht\@kludgeins
322 (*trace)
323 \fl@trace {Extra size added: \the \ht\@kludgeins}%
324 (/trace)
325 \fi
326 \fi

```

This version puts the inserts back just before the additional material; it could be moved earlier, before unboxing the page-so-far. Neither is guaranteed not to put things on the wrong page. This version is similar to the original version.

```

327         \@reinserts
328         \@addtocurcol
329     \else
330         \@reinserts
331         \@addmarginpar
332     \fi
333 } \@latexbug

```

A 2e change: use `\addpenalty` instead of `\penalty` here. Some penalty is needed to create a potential break-point immediately after the reinserts (or the marginal). Otherwise there can be no possibility to break here and this can cause the reinserts or the marginal to appear on the next page (which is often incorrect). However, if the nobreak flag is true, a `\nobreak` must be correct.

```

334     \ifnum \outputpenalty<\z@
335     \if@nobreak
336         \nobreak
337     \else
338         \addpenalty \interlinepenalty
339     \fi
340 \fi
341 \fi
342 \fi
343 }
344 </2ekernel | fltrace>

```

(End definition for `\output` and `\@specialoutput`.)

`\@testwrongwidth` Test if the float box has the wrong width when trying to place it into some area. (Actually the test is for a conventional depth setting rather than for the width of the float. For that reason the box depth was explicitly tailored when the float was created).

`\f@depth`

```

345 <[latexrelease]>\IncludeInRelease{2015/01/01}%
346 <[latexrelease]>          {\@testwrongwidth}{float order in 2-column}%
347 <*2ekernel | latexrelease | fltrace>

348 \def\@testwrongwidth #1{%
349     \ifdim\dp#1=\f@depth
350     <*trace>
351         \fl@trace{\string#1
352             \ifdim\f@depth=\z@ single \else double \fi
353             column float -- ok}%
354 </trace>
355     \else
356         \global\@testtrue
357     <*trace>
358         \fl@trace{\string#1
359             \ifdim\f@depth=\z@ double \else single \fi
360             column float -- wrong}%
361 </trace>
362     \fi}%

```

Normally looking for single column floats, which have zero depth.

```

363 \let\f@depth\z@

```

```

364 </2ekernel | latexrelease | fltrace>
365 <latexrelease>\EndIncludeInRelease
366 <latexrelease>\IncludeInRelease{0000/00/00}%
367 <latexrelease>{\@testwrongwidth}{float order in 2-column}%
368 <latexrelease>\let\@testwrongwidth\@undefined
369 <latexrelease>\let\f@depth\@undefined
370 <latexrelease>\EndIncludeInRelease

```

(End definition for \@testwrongwidth and \f@depth.)

\@docclearpage This is a very much an emergency action, just dumping everything: footnotes first then floats. A more sophisticated version is needed; but even more urgent is a bug-free version (see, for example, pr/3528).

Also, it puts any left-over non-boxes (writes, specials, etc.) back after any float pages created: this is a very bad bug since, for example, a kludge insert will be in quite the wrong place and, worse, be irremovable and uncancelable.

All the remaining changes are replacing the double column defer list or inserting the extra test \@testwrongwidth{<box>} at suitable places. That is at places where a box is taken off the deferlist.

```

371 <latexrelease>\IncludeInRelease{2015/01/01}{\@docclearpage}%
372 <latexrelease>{\float order in 2-column}%
373 <*2ekernel | latexrelease>
374 \def \@docclearpage {%
375     \ifvoid\footins
376         \ifvbox\@kludgeins
377             {\setbox \@tempboxa \box \@kludgeins}%
378             <*trace>
379             \fl@trace {kludgeins box made void}%
380         </trace>
381         \fi
382         \setbox\@tempboxa\vsplit\@cclv to\z@ \unvbox\@tempboxa
383         \setbox\@tempboxa\box\@cclv
384         \xdef\@deferlist{\@toplist\@botlist\@deferlist}%
385         \global \let \@toplist \@empty
386         \global \let \@botlist \@empty
387         \global \@colroom \@colht
388         \ifx \@currlist\@empty
389             \else
390                 \@latexerror{Float(s) lost}\@ehb
391                 \global \let \@currlist \@empty
392             \fi
393         \@makefcolumn\@deferlist
394         \@whilesw\if@fcolmade \fi{\@opcol\@makefcolumn\@deferlist}%
395         \if@twocolumn
396             \if@firstcolumn
397                 \xdef\@deferlist{\@dbltoplist\@deferlist}%
398                 \global \let \@dbltoplist \@empty
399                 \global \@colht \textheight
400                 \begingroup
401                 \@dblfloatplacement

```



```

402             \@makefcolumn\@deferlist
403             \@whiles\if@fcolmade \fi{\@outputpage
404                                     \@makefcolumn\@deferlist}%
405         \endgroup
406     \else
407         \vbox{}\clearpage
408     \fi
409 \fi

```

the next line is needed to avoid losing floats in certain circumstances a single call to the original `\docclearpage` will now no longer output all floats.

```

410     \ifx\@deferlist\@empty \else\clearpage \fi
411 \else
412     \setbox\@cclv\vbox{\box\@cclv\vfil}%
413     \@makecol\@opcol
414     \clearpage
415 \fi
416 }%
417 </2kernel | latexrelease>
418 <latexrelease>\EndIncludeInRelease
419 <latexrelease>\IncludeInRelease{0000/00/00}{\@docclearpage}%
420 <latexrelease>                                     {float order in 2-column}%
421 <latexrelease>\def \@docclearpage {%
422 <latexrelease>     \ifvoid\footins

```

We empty any left over kludge insert box here; this is a temporary fix. It should perhaps be applied to one page of cleared floats, but who cares? The whole of this stuff needs completely redoing for many such reasons.

```

423 <latexrelease>     \ifvbox\@kludgeins
424 <latexrelease>     {\setbox \@tempboxa \box \@kludgeins}%
425 <*trace>
426 <latexrelease>     \fl@trace {kludgeins box made void}%
427 </trace>
428 <latexrelease>     \fi
429 <latexrelease>     \setbox\@tempboxa\vsplit\@cclv to\z@ \unvbox\@tempboxa
430 <latexrelease>     \setbox\@tempboxa\box\@cclv
431 <latexrelease>     \xdef\@deferlist{\@toplist\@botlist\@deferlist}%
432 <latexrelease>     \global \let \@toplist \@empty
433 <latexrelease>     \global \let \@botlist \@empty
434 <latexrelease>     \global \@colroom \@colht
435 <latexrelease>     \ifx \@currlist\@empty
436 <latexrelease>     \else
437 <latexrelease>         \@latexerr{Float(s) lost}\@ehb
438 <latexrelease>     \global \let \@currlist \@empty
439 <latexrelease>     \fi
440 <latexrelease>     \@makefcolumn\@deferlist
441 <latexrelease>     \@whiles\if@fcolmade \fi
442 <latexrelease>         {\@opcol\@makefcolumn\@deferlist}%
443 <latexrelease>     \if@twocolumn
444 <latexrelease>         \if@firstcolumn
445 <latexrelease>             \xdef\@dbldeferlist{\@dbltoplist\@dbldeferlist}%

```

```

446 <latexrelease>          \global \let \@dbltoplist \@empty
447 <latexrelease>          \global \@colht \textheight
448 <latexrelease>          \begingroup
449 <latexrelease>            \dblfloatplacement
450 <latexrelease>            \makefcolumn\@dbldeferlist
451 <latexrelease>            \whiles\if@fcolmade \fi
452 <latexrelease>            {\@outputpage\@makefcolumn\@dbldeferlist}%
453 <latexrelease>          \endgroup
454 <latexrelease>          \else
455 <latexrelease>            \vbox{}\clearpage
456 <latexrelease>          \fi
457 <latexrelease>          \fi
458 <latexrelease>          \else
459 <latexrelease>            \setbox\@cclv\vbox{\box\@cclv\vfil}%
460 <latexrelease>            \makecol\@opcol
461 <latexrelease>            \clearpage
462 <latexrelease>          \fi
463 <latexrelease>        }%
464 <latexrelease>\EndIncludeInRelease

```

(End definition for \@doclearpage.)

\@opcol Several changes in detail here.

```

465 <*2ekernel | fltrace>
466 \def \@opcol {%
467   \if@twocolumn
468     \outputdblcol
469   \else
470     \outputpage
471 <*trace>
472   \fl@trace{PAGE: one column (float? see above) page completed}%
473 </trace>

```

Not needed since it comes after \@outputpage:

```

474 %   \global\@colht\textheight
475   \fi

```

These do not need to be done every time \@opcol is used: they should be grouped together since they all need to be done at the end of the non-special output routine, or at the end of a clearpage one.

```

476   \global \@mparbottom \z@ \global \@textfloatsheight \z@
477   \floatplacement
478 }
479 </2ekernel | fltrace>

```

(End definition for \@opcol.)

\@makecol We must rewrite this macro to allow for variations in page-makeup required by changes in page-length.

This uses a different macro if a special-length column is being produced.

```

480 <*2ekernel>
481 \gdef \@makecol {%
482   \ifvoid\footins
483     \setbox\@outputbox \box\@cclv
484   \else
485     \setbox\@outputbox \vbox {%

```

This `\boxmaxdepth` setting is to ensure that deep footnotes do not overwrite the footer (on account of the negative skip added later): it should use `\@maxdepth` otherwise the change is pointless when there are footnotes.

But see also its use when combining floats.

```

486      \boxmaxdepth \@maxdepth

487 %      \@tempdima\dp\@cclv
488      \unvbox \@cclv
489 %      \vskip-\@tempdima
490      \vskip \skip\footins

491      \color@begingroup
492      \normalcolor
493      \footnoterule
494      \unvbox \footins
495      \color@endgroup
496    }%
497  \fi

```

The h floats have now been finally committed to this page so we can reset their list. The top and bottom floats are then added to the page.

```

498  \let\@elt\relax
499  \xdef\@freelist{\@freelist\@midlist}%

500  \global \let \@midlist \@empty
501  \@combinefloats

```

The variations start here in case `\enlargethispage` has been used.

```

502  \ifvbox\@kludgeins
503    \@makespecialcolbox
504  \else

```

This extra reboxing is only needed to add the `\@texttop` and `\@textbottom` but this could be done earlier, when the floats are added.

The `\boxmaxdepth` resetting here will have no effect unless `\@textbottom` ends with a box or rule. So is this (or possibly `\@maxdepth`) the correct value?

The `\vskip -\dimen@` ensures that the visible depth of the box does not affect the placement of anything on the page. Thus very deep pages will overprint the footer; but these should have been prevented by suitable settings of the maxdepths at appropriate times.

If `\@textbottom` ends with a box or rule of non-zero depth then this skip adjustment should be done again after it.

I think that the final boxing of the main text page could have a common ending which may make it simpler to see what is going on.

This needs further investigation, especially in the ‘special case’.

Also, the `\boxmaxdepth` setting here affects what happens within `\@texttop` and `\@textbottom`, should it? Is it needed at all?

RmS 91/10/22: Replaced `\dimen128` by `\dimen@`.

```

505  \setbox\@outputbox \vbox to\@colht {%
506 %      \boxmaxdepth \maxdepth           %??
507      \@texttop
508      \dimen@ \dp\@outputbox
509      \unvbox \@outputbox

```

```

510      \vskip -\dimen@
511      \@textbottom
512    }%
513  \fi
514  \global \maxdepth \@maxdepth
515 }

```

(End definition for \makecol.)

\@reinserts This is the code which reinserts the inserts. It puts them all in one place; this can make some of them come out on the wrong page. It has been put into a separate macro to expedite experimentation.

```

516 \gdef \@reinserts{%
517   \ifvoid\footins\else\insert\footins{\unvbox\footins}\fi
518   \ifvbox\@kludgeins\insert\@kludgeins
519     {\unvbox\@kludgeins}\fi
520 }
521 </2ekernel>

```

(End definition for \@reinserts.)

\@makespecialcolbox This implements certain variations in page-makeup.

```

522 <*2ekernel | fltrace>
523 \gdef \@makespecialcolbox {%
524 <*trace>
525   \fl@trace{Kludgeins ht \the\ht\@kludgeins\space
526                                     dp \the\dp\@kludgeins\space
527                                     wd \the\wd\@kludgeins}%
528 </trace>

```

First we find the natural height of the column.

See above for discussion of what is happening here.

This needs further investigation, especially in this ‘special case’.

```

529 \setbox\@outputbox \vbox {%
530   \@texttop
531   \dimen@ \dp\@outputbox
532   \unvbox\@outputbox
533   \vskip-\dimen@
534 }%
535 \@tempdima \@colht
536 \ifdim \wd\@kludgeins>\z@

```

Note that in this case (the *-version), the height of the \@kludgeins box is not used since its value is somewhat arbitrary: it need only be big enough to ensure that the page-break is not taken prematurely.

Here we calculate how much vertical space needs to be added in order to enable the column to fit into a box of size \@colht using the best information we have about the amount of shrink available (another thing which is known internally about a box, but cannot be accessed at the T_EX level!).

This needs T_EX3 otherwise \pageshrink is zero anyway; it may not be exactly the figure we wish as it is the total available from the all the material collected before the page-break decision is made. It will, we think, always be an overestimate of the actual shrink in the box; therefore this should always force the shortest possible column with the possibility of an overfull box.

This should work for both flush- and ragged-bottom setting since it makes the contents no smaller than the size (`\@colht`) of the box into which they are put.

Their should perhaps be an upper limit, of 0pt?, on the extra space added to force shrinking.

See above for a discussion of the `\boxmaxdepth` setting here.

```

537     \advance \@tempdima -\ht\@outputbox
538     \advance \@tempdima \pageshrink
539 <*trace>
540     \fl@trace {Natural ht of col: \the \ht\@outputbox}%
541     \fl@trace {\string \@colht: \the \@colht}%
542     \fl@trace {Pageshrink added: \the \pageshrink}%
543     \fl@trace {Hence, space added: \the \@tempdima}%
544 </trace>
545     \setbox\@outputbox \vbox to \@colht {%
546 %       \boxmaxdepth \maxdepth
547       \unvbox\@outputbox
548       \vskip \@tempdima
549       \@textbottom
550     }%
```

For the unstarred version, the final size of the page is precisely specified. Therefore, at least for the flush-bottom case, we need to ensure that, visually, it has this size exactly.

Thus we calculate this size and set the material in a box of this size, which is then put into a box of size `\@colht` with `\vss` at the bottom.

```

551     \else
552     \advance \@tempdima -\ht\@kludgeins
553 <*trace>
554     \fl@trace {Natural ht of col: \the \ht\@outputbox}%
555     \fl@trace {\string \@colht: \the \@colht}%
556     \fl@trace {Extra size added: -\the \ht \@kludgeins}%
557     \fl@trace {Hence, height of inner box: \the \@tempdima}%
558     \fl@trace {Max? pageshrink available: \the \pageshrink}%
559 </trace>
```

This type of final packaging could be done always; this may simplify all of this page-makeup.

It is not necessary to set `\boxmaxdepth` here since the `\@outputbox` ends with glue.

```

560     \setbox \@outputbox \vbox to \@colht {%
561       \vbox to \@tempdima {%
562         \unvbox\@outputbox
563         \@textbottom}%
564     \vss}%
565 \fi
```

Finally we need to explicitly make the insert box void.

```

566     {\setbox \@tempboxa \box \@kludgeins}%
567 <*trace>
568     \fl@trace {kludgeins box made void}%
569 </trace>
570 }
571 </2ekernel | fltrace>
```

(End definition for \@makespecialcolbox.)

```

\@texttop These do nothing as a default.
\@textbottom
572 <*2ekernel>
573 \let \@texttop \relax
574 \let \@textbottom \relax

```

(End definition for \@texttop and \@textbottom.)

\@resetactivechars RmS 93/09/06: added hook to protect against certain active characters in the output routine. Default checks are for active space and end-of-line.

```

575 \def\@activechar@info #1{%
576     \@latex@info@no@line {Active #1 character found while
577                             output routine is active
578                             \MessageBreak
579                             This may be a bug in a package file
580                             you are using}%
581 }

```

Do not put any spaces in this next bit!

```

582 \begingroup
583 \obeylines\obeyspaces%
584 \catcode'\'\active%
585 \gdef\@resetactivechars{%
586 \def~M{\@activechar@info{EOL}\space}%
587 \def {\@activechar@info{space}\space}%
588 \let'\active@math@prime}%
589 \endgroup

```

(End definition for \@resetactivechars and \@activechar@info.)

\@outputpage The \color@hbox hooks here are used to avoid putting just a colour special into an otherwise empty box (in a header or footer). These boxes are often set to be completely empty and so adding a special produces a very underfull box message.

There has been extensive tidying up of the old code here; including the removal of a level of grouping.

The setting of \protect immediately before the \shipout is needed so that protected commands within \writes are handled correctly.

Within shipout's vbox it is reset to its default value, \relax.

Resetting it to its default value after the shipout has been completed (and the contents of the writes have been expanded) must be done by use of \aftergroup. This is because it must have the value \relax before macros coming from other uses of \aftergroup within this box are expanded.

Putting this into the \aftergroup token list does not affect the definition used in expanding the \writes because the aftergroup token list is only constructed when popping the save-stack, it is not expanded until after the shipout is completed.

Question: should things from an \aftergroup within the shipped out box be executed in the environment set up for the writes, or after it finishes?

A lot of this code has been in-lined to prevent mis-use of internal commands as hooks.

```

590 </2ekernel>
591 <latexrelease>\IncludeInRelease{2017/04/15}%
592 <latexrelease> {\@outputpage}{Reset language for hyphenation}%
593 <*2ekernel | latexrelease>
594 \def\@outputpage{%

```

The `\endgroup` is put in by `\aftergroup`.

```
595 \begingroup
```

Now all the set-up stuff has been in-lined for Frank.

First the stuff for the writes.

From here ... was in the command `\@writesetup`.

```
596 \let \protect \noexpand
```

RmS 93/08/19: Redefined accents to allow changes in font encoding; but exactly why was this needed?

Reset `\language` to the value current at `\begin{document}`. In particular this ensures that a pagebreak in `verbatim` does not prevent hyphenation in the page head.

```
597 \language\document@default@language
```

The `\catcode'\ = 10` was removed as it was considered useless (presumably because nothing gets tokenized during shipout).

This was put in as some error produced active spaces in a mark, I think.

Why was the hyphen reset?

```
598 \@resetactivechars
```

If a page break happens between the start of a list and its first item the `@newlist` will be true and this will mess up any list that is used in the header or footer of the page. So we have to reset that flag.

```
599 \global\let\@@if@newlist\if@newlist
```

```
600 \global\@newlistfalse
```

This next hook replaces the following:

```
\let\-\@dischyph
```

```
\let'\@acci\let'\@accii\let\=\@acciii
```

```
\let\\@normalcr
```

```
\let\par\@par %% 15 Sep 87 (this was once inside the box)
```

and it does more than they did; in particular it sets:

```
\parindent\z@
```

```
\parskip\z@skip
```

```
\everypar{}%
```

```
\leftskip\z@skip
```

```
\rightskip\z@skip
```

```
\parfillskip\@flushglue
```

```
\lineskip\normallineskip
```

```
\baselineskip\normalbaselineskip
```

```
\sloppy
```

```
601 \@parboxrestore
```

... to here was in the command `\@writesetup`.

```
602 \shipout \vbox{%
```

```
603 \set@typeset@protect
```

```
604 \aftergroup \endgroup
```

Correct? or just restore by ending the group?

```
605 \aftergroup \set@typeset@protect
```

This first bit has been moved inside the shipped out box.

Now the setup inside the shipped out box; this should contain all the stuff that could only affect typesetting; other stuff may need to be reset for the writes also.

From here ... was in the command \@shipoutsetup.

```
606 \if@specialpage
607 \global\@specialpagefalse\@nameuse{ps@\@specialstyle}%
608 \fi
609 \if@twoside
610 \ifodd\count\z@ \let\@thehead\@oddhead \let\@thefoot\@oddfoot
611 \let\@themargin\oddsidemargin
612 \else \let\@thehead\@evenhead
613 \let\@thefoot\@evenfoot \let\@themargin\evensidemargin
614 \fi
615 \fi
```

The rest was always inside the box.

RmS 91/08/15: added this line:

```
616 \reset@font
```

RmS 93/08/06 Added \lineskiplimit=0pt to guard against it being nonzero: e.g. by \offinterlineskip being in effect.

There are probably lots of other things that may need resetting.

```
617 \normalsize
```

Reset the space factors.

```
618 \normalsfcodes
```

Reset these here (previously reset separately for head and foot)

```
619 \let\label\@gobble
620 \let\index\@gobble
621 \let\glossary\@gobble
622 \baselineskip\z@skip \lineskip\z@skip \lineskiplimit\z@
```

... to here was in the command \@shipoutsetup.

```
623 \@beginndvi
624 \vskip \topmargin
625 \moveright\@themargin \vbox {%
626 \setbox\@tempboxa \vbox to\headheight{%
627 \vfil
628 \color@hbox
629 \normalcolor
630 \hb@xt@\textwidth{\@thehead}%
631 \color@endbox
```

22 Feb 87

```
632 }%
633 \dp\@tempboxa \z@
634 \box\@tempboxa
635 \vskip \headsep
636 \box\@outputbox
637 \baselineskip \footskip
638 \color@hbox
639 \normalcolor
640 \hb@xt@\textwidth{\@thefoot}%
641 \color@endbox
```



```

642     }%
643 }%
\endgroup now inserted by \aftergroup
  Restore \if@newlist
644   \global\let\if@newlist\@if@newlist
645   \global \colht \textheight
646   \stepcounter{page}%

```

It is now clear that this does something useful, thanks to Piet van Oostrum. It is needed because a float page is made without using TeX's page-builder; thus the output routine is never called so the marks are not updated.

```

647   \let\firstmark\botmark
648 }
649 </2ekernel | latexrelease>
650 <latexrelease>\EndIncludeInRelease
651 <latexrelease>\IncludeInRelease{0000/00/00}%
652 <latexrelease> {\@outputpage}{Reset language for hyphenation}%
653 <latexrelease>\def\@outputpage{%
654 <latexrelease>\begingroup
655 <latexrelease> \let \protect \noexpand
656 <latexrelease> \@resetactivechars
657 <latexrelease> \global\let\@if@newlist\if@newlist
658 <latexrelease> \global\@newlistfalse
659 <latexrelease> \@parboxrestore
660 <latexrelease> \shipout \vbox{%
661 <latexrelease> \set@typeset@protect
662 <latexrelease> \aftergroup \endgroup
663 <latexrelease> \aftergroup \set@typeset@protect
664 <latexrelease> \if@specialpage
665 <latexrelease> \global\@specialpagefalse\@nameuse{ps@\@specialstyle}%
666 <latexrelease> \fi
667 <latexrelease> \if@twoside
668 <latexrelease> \ifodd\count\z@
669 <latexrelease> \let\@thehead\@oddhead \let\@thefoot\@oddfoot
670 <latexrelease> \let\@themargin\oddsidemargin
671 <latexrelease> \else \let\@thehead\@evenhead
672 <latexrelease> \let\@thefoot\@evenfoot \let\@themargin\evensidemargin
673 <latexrelease> \fi
674 <latexrelease> \fi
675 <latexrelease> \reset@font
676 <latexrelease> \normalsize
677 <latexrelease> \normalsfcodes
678 <latexrelease> \let\label\@gobble
679 <latexrelease> \let\index\@gobble
680 <latexrelease> \let\glossary\@gobble
681 <latexrelease> \baselineskip\z@skip \lineskip\z@skip \lineskiplimit\z@
682 <latexrelease> \@begindvi
683 <latexrelease> \vskip \topmargin
684 <latexrelease> \moveright\@themargin \vbox {%
685 <latexrelease> \setbox\@tempboxa \vbox to\headheight{%
686 <latexrelease> \vfil
687 <latexrelease> \color\hbox
688 <latexrelease> \normalcolor

```

```

689 <latexrelease>          \hb@xt@\textwidth{\@thehead}%
690 <latexrelease>          \color@endbox
691 <latexrelease>          }%
692 <latexrelease>          \dp\@tempboxa \z@
693 <latexrelease>          \box\@tempboxa
694 <latexrelease>          \vskip \headsep
695 <latexrelease>          \box\@outputbox
696 <latexrelease>          \baselineskip \footskip
697 <latexrelease>          \color@hbox
698 <latexrelease>          \normalcolor
699 <latexrelease>          \hb@xt@\textwidth{\@theft}%
700 <latexrelease>          \color@endbox
701 <latexrelease>          }%
702 <latexrelease>          }%
703 <latexrelease> \global\let\if@newlist\@if@newlist
704 <latexrelease> \global \colht \textheight
705 <latexrelease> \stepcounter{page}%
706 <latexrelease> \let\firstmark\botmark
707 <latexrelease> }
708 <latexrelease> \EndIncludeInRelease
709 <*2ekernel>

```

(End definition for \@outputpage, \@shipoutsetup, and \@writesetup.)

\@beginndvi This unboxes stuff that must appear before anything else in the .dvi file, then returns that box register to the free list and cancels itself.

The stuff in the box should not add any typeset material to the page.

```

710 \def \@beginndvi{%
711   \unvbox \@beginndvibox
712   \global\let \@beginndvi \@empty
713 }

```

(End definition for \@beginndvi.)

\@combinefloats The \boxmaxdepth setting here was not made local to a box so was dangerous. It is needed only within the box made by \@cflt (and not normally even there), so it has been moved there; this also agrees with the original pseudocode.

```

714 \def \@combinefloats {%
715   % \boxmaxdepth \maxdepth
716   \ifx \@toplist\@empty \else \@cflt \fi
717   \ifx \@botlist\@empty \else \@cflb \fi
718 }

719 \def \@cflt{%
720   \let \@elt \@comflelt
721   \setbox\@tempboxa \vbox{%
722     \@toplist
723     \setbox\@outputbox \vbox{%
724       \boxmaxdepth \maxdepth
725       \unvbox\@tempboxa
726       \vskip -\floatsep
727       \topfigrule
728       \vskip \textfloatsep
729       \unvbox\@outputbox

```

```

730 }%
731 \let\@elt\relax
732 \xdef\@freelist{\@freelist\@toplist}%
733 \global\let\@toplist\@empty
734 }
735 \def \@cflb {%
736 \let\@elt\@comflelt
737 \setbox\@tempboxa \vbox{}%
738 \@botlist
739 \setbox\@outputbox \vbox{%
740 \unvbox\@outputbox
741 \vskip \textfloatsep
742 \botfigrule
743 \unvbox\@tempboxa
744 \vskip -\floatsep
745 }%
746 \let\@elt\relax
747 \xdef\@freelist{\@freelist\@botlist}%
748 \global \let \@botlist\@empty
749 }

```

(End definition for \@combinefloats, \@cflt, and \@cflb.)

```

\@comflelt
\@comdblfelet
\@combinedblfloats
750 \def\@comflelt#1{\setbox\@tempboxa
751 \vbox{\unvbox\@tempboxa\box #1\vskip\floatsep}}
752 \def\@comdblfelet#1{\setbox\@tempboxa
753 \vbox{\unvbox\@tempboxa\box #1\vskip\dblfloatsep}}
754 \def \@combinedblfloats{%
755 \ifx \@dbltoplist \@empty
756 \else
757 \setbox\@tempboxa \vbox{}%
758 \let \@elt \@comdblfelet
759 \@dbltoplist
760 \let \@elt \relax
761 \xdef \@freelist {\@freelist\@dbltoplist}%
762 \global\let \@dbltoplist \@empty
763 \setbox\@outputbox \vbox to\textheight

```

The setting of \boxmaxdepth here has no effect since the \@outputbox should already have depth zero. Even so, it would have no effect on the layout of the page.

```

764 {\boxmaxdepth\maxdepth %% probably not needed, CAR
765 \unvbox\@tempboxa\vskip-\dblfloatsep

```

Here we need different typesetting if the top float comes from \@topnewpage.

```

766 \ifnum \@dbltopnum>\m@ne
767 \dblfigrule
768 \fi
769 \vskip \dbltextfloatsep

```

If pdf links are present in the galley and those links get broken across pages they have to end up being on the same level of boxing (even if not actually in the same structure) due to some engine restrictions in pdf \TeX and Lua \TeX . We therefore unbox \@outputbox

here (which only contains a single `\hbox`) so that this case has the same boxing level as a normal twocolumn page without top floats.

```

770      \unvbox\@outputbox
771    }%
772  \fi
773 }
774 \endkernel

```

(End definition for `\@comfleft`, `\@comdblleft`, and `\@combinedblfloats`.)

`\@startcolumn` We could combine (most of) these two into `\@startcol <list>`. Note that `\@xstartcol` was only used once (i.e. in `\@startcolumn`); it has therefore been removed. This is not quite as efficient but it now has the same structure as `\@startdblcolumn`.

The empty-list test has been moved to `\@tryfcolumn`.

```

775 \endkernel | fltrace)
776 \def \@startcolumn {%
777   \global \@colroom \@colht
778   \@tryfcolumn \@deferlist
779   \if@fcolmade
780 \endkernel | fltrace)
781   \fl@trace{PAGE: float \if@twocolumn column \else page \fi
782             completed}%
783 \endkernel | fltrace)
784 \else
785   \begingroup
786     \let \reserved@b \@deferlist
787     \global \let \@deferlist \@empty
788     \let \@elt \@scolelt
789     \reserved@b
790   \endgroup
791 \fi
792 }

```

This one does not need to set `\@colht`.

```

793 \endkernel | fltrace)
794 \latexrelease | fltrace)\IncludeInRelease{2015/01/01}%
795 \latexrelease | fltrace) {\@startdblcolumn}{float order in 2-column}%
796 \endkernel | latexrelease | fltrace)
797 \def \@startdblcolumn {%
798   \@tryfcolumn \@deferlist
799   \if@fcolmade
800 \fltrace) \fl@trace{PAGE: double float page completed}%
801 \else
802   \begingroup
803     \let \reserved@b \@deferlist
804     \global \let \@deferlist \@empty
805     \let \@elt \@sdblcolelt
806     \reserved@b
807   \endgroup
808 \fi
809 }%
810 \endkernel | latexrelease | fltrace)
811 \latexrelease | fltrace)\EndIncludeInRelease

```

```

812 <latexrelease | fltrace>\IncludeInRelease{0000/00/00}%
813 <latexrelease | fltrace> {\@startdblcolumn}{float order in 2-column}%
814 <latexrelease | fltrace>\def \@startdblcolumn {%

Not needed since this always comes after \@outputpage:

815 <latexrelease | fltrace>% \global \@colht \textheight
816 <latexrelease | fltrace> \@tryfcolumn \@dbldeferlist
817 <latexrelease | fltrace> \if@fcolmade
818 <*trace>
819 <latexrelease | fltrace> \fl@trace{PAGE: double float page completed}%
820 </trace>
821 <latexrelease | fltrace> \else

822 <latexrelease | fltrace> \begingroup
823 <latexrelease | fltrace> \let \reserved@b \@dbldeferlist
824 <latexrelease | fltrace> \global \let \@dbldeferlist \@empty
825 <latexrelease | fltrace> \let \@elt \@sdblcolelt
826 <latexrelease | fltrace> \reserved@b
827 <latexrelease | fltrace> \endgroup
828 <latexrelease | fltrace> \fi
829 <latexrelease | fltrace>}%
830 <latexrelease | fltrace>\EndIncludeInRelease
831 <*2ekernel | fltrace>

```

(End definition for \@startcolumn and \@startdblcolumn.)

\@tryfcolumn Now tests if its list is empty before any further exertion.

```

832 \def \@tryfcolumn #1{%
833   \global \@fcolmadefalse
834   \ifx #1\@empty
835   \else
836   <*trace>
837     \fl@trace{PAGE: try float \if@twocolumn column/page\else page\fi
838               ---\string #1}%
839     \fl@trace{----- \string #1: #1}%
840   </trace>

841   \xdef\@trylist{#1}%
842   \global \let \@failedlist \@empty
843   \begingroup
844     \let \@elt \@xtryfc \@trylist
845   \endgroup
846   \if@fcolmade
847     \@vtryfc #1%
848   \fi
849   \fi
850 }
851 </2ekernel | fltrace>

```

(End definition for \@tryfcolumn.)

```

852 <*2ekernel>

```

\@scolelt

```

853 \def\@scolelt#1{\def\@currbox{#1}\@addtonextcol}

```

```

(End definition for \@scolelt.)

\@sdblcrolelt
854 \def\@sdblcrolelt#1{\def\@currbox{#1}\@addtodblcol}
(End definition for \@sdblcrolelt.)

\@vtryfc
855 \def\@vtryfc #1{%
856   \global\setbox\@outputbox\vbox{}%
857   \let\@elt\@wtryfc
858   \@flsucceed
859   \global\setbox\@outputbox \vbox to\@colht{%
860     \vskip \@fptop
861     \vskip -\@fpsep
862     \unvbox \@outputbox
863     \vskip \@fpbot}%
864   \let\@elt\relax
865   \xdef #1{\@failedlist\@flfail}%
866   \xdef\@freelist{\@freelist\@flsucceed}}
(End definition for \@vtryfc.)

\@wtryfc
867 \def\@wtryfc #1{%
868   \global\setbox\@outputbox\vbox{%
869     \unvbox\@outputbox
870     \vskip\@fpsep
871     \box #1}}
(End definition for \@wtryfc.)

\@xtryfc
872 </2ekernel>
873 <latexrelease>\IncludeInRelease{2015/01/01}{\@xtryfc}%
874 <latexrelease> {float order in 2-column}%
875 <*2ekernel | latexrelease>
876 \def\@xtryfc #1{%
877   \@next\reserved@a\@trylist{}{}%
878   \@currtype \count #1%
879   \divide\@currtype\@xxxii
880   \multiply\@currtype\@xxxii
881   \@bitor \@currtype \@failedlist
882   \@testfp #1%
883   \@testwrongwidth #1%
884   \ifdim \ht #1>\@colht
885     \@testtrue
886   \fi
887   \if@test
888     \@cons\@failedlist #1%
889   \else
890     \@ytryfc #1%
891   \fi}%
892 </2ekernel | latexrelease>

```

```

893 <latexrelease>\EndIncludeInRelease
894 <latexrelease>\IncludeInRelease{0000/00/00}{\@xtryfc}%
895 <latexrelease>                                     {float order in 2-column}%
896 <latexrelease>\def\@xtryfc #1{%
897 <latexrelease>  \@next\reserved@a\@trylist{}}}%
898 <latexrelease>  \@currtype \count #1%
899 <latexrelease>  \divide\@currtype\@xxxii
900 <latexrelease>  \multiply\@currtype\@xxxii
901 <latexrelease>  \@bitor \@currtype \@failedlist
902 <latexrelease>  \@testfp #1%
903 <latexrelease>  \ifdim \ht #1>\@colht
904 <latexrelease>    \testtrue
905 <latexrelease>  \fi
906 <latexrelease>  \if@test
907 <latexrelease>    \@cons\@failedlist #1%
908 <latexrelease>  \else
909 <latexrelease>    \@ytryfc #1%
910 <latexrelease>  \fi}%
911 <latexrelease>\EndIncludeInRelease
912 <*2kernel>

```

(End definition for \@xtryfc.)

\@ytryfc

```

913 \def\@ytryfc #1{%
914   \begingroup
915   \gdef\@flsucceed{\@elt #1}%
916   \global\let\@flfail\@empty
917   \@tempdima\ht #1%
918   \let\@elt\@ztryfc
919   \@trylist
920   \ifdim \@tempdima >\@fpmin
921     \global\@fcolmadetrue
922   \else
923     \@cons\@failedlist #1%
924   \fi
925   \endgroup
926   \if@fcolmade
927     \let\@elt\@gobble
928   \fi}

```

(End definition for \@ytryfc.)

\@ztryfc

```

929 </2kernel>
930 <latexrelease>\IncludeInRelease{2015/01/01}{\@ztryfc}%
931 <latexrelease>                                     {float order in 2-column}%
932 <*2kernel | latexrelease>
933 \def\@ztryfc #1{%
934   \@tempcnta\count #1%
935   \divide\@tempcnta\@xxxii
936   \multiply\@tempcnta\@xxxii
937   \@bitor \@tempcnta {\@failedlist \@flfail}%
938   \@testfp #1%

```

```

not in fixfloats?
939 \testwrongwidth #1%
940 \@tempdimb\@tempdima
941 \advance\@tempdimb\ht #1%
942 \advance\@tempdimb\@fpsep
943 \ifdim \@tempdimb >\@colht
944 \testtrue
945 \fi
946 \if@test
947 \cons\@flfail #1%
948 \else
949 \cons\@flsucceed #1%
950 \tempdima\@tempdimb
951 \fi}%
952 </2ekernel | latexrelease>
953 <latexrelease>\EndIncludeInRelease
954 <latexrelease>\IncludeInRelease{0000/00/00}{\ztryfc}%
955 <latexrelease> {float order in 2-column}%
956 <latexrelease>\def\@ztryfc #1{%
957 <latexrelease> \@tempcnta \count#1%
958 <latexrelease> \divide\@tempcnta\@xxxii
959 <latexrelease> \multiply\@tempcnta\@xxxii
960 <latexrelease> \bitor \@tempcnta {\@failedlist \@flfail}%
961 <latexrelease> \testfp #1%
962 <latexrelease> \@tempdimb\@tempdima
963 <latexrelease> \advance\@tempdimb \ht#1%
964 <latexrelease> \advance\@tempdimb\@fpsep
965 <latexrelease> \ifdim \@tempdimb >\@colht
966 <latexrelease> \testtrue
967 <latexrelease> \fi
968 <latexrelease> \if@test
969 <latexrelease> \cons\@flfail #1%
970 <latexrelease> \else
971 <latexrelease> \cons\@flsucceed #1%
972 <latexrelease> \tempdima\@tempdimb
973 <latexrelease> \fi}%
974 <latexrelease>\EndIncludeInRelease

```

(End definition for \@ztryfc.)

The major changes for float suppression and the changes to the float mechanism to make it conform to the documentation are in these next macros.

\@addtobot Lots of changes.

```

975 <*2ekernel | fltrace>
976 \def \@addtobot {%
977 <*trace>
978 \fl@trace{***Start addtobot}%
979 </trace>
980 \@getfpsbit 4\relax
981 <*trace>
982 \fl@trace{fpstype \ifodd \@tempcnta OK \else not \fi bot:
983 \the \@fpstype}%
984 </trace>

```



```

985 \ifodd \@tempcnta
986 \flsetnum \@botnum
987 \ifnum \@botnum>\z@
988 \@tempswafalse
989 \flcheckspace \@botroom \@botlist
990 \if@tempswa

```

This next line means that this page is produced with box 255 having depth zero, rather than the normal maxdepth: is this needed, useful?

```

991 \global \maxdepth \z@
992 \flupdates \@botnum \@botroom \@botlist
993 \*trace
994 \fl@trace{colroom (after-bot) = \the \@colroom}%
995 \fl@trace{colnum (after-bot) = \the \@colnum}%
996 \fl@trace{botnum (after-bot) = \the \@botnum}%
997 \fl@trace{***Success: bot}%
998 \*trace
999 \inserttrue
1000 \fi
1001 \*trace
1002 \else
1003 \fl@trace{Fail: botnum = \the \@botnum:
1004 fpstype \the \@fpstype=ORD?}%
1005 \ifnum \@fpstype<\sixt@n
1006 \fl@trace{ERROR: !b float not successful (addtobot)}%
1007 \fi
1008 \*trace
1009 \fi
1010 \fi
1011 }

```

(End definition for \@addtobot.)

\@addtotoporbot Lots of changes.

```

1012 \def \@addtotoporbot {%
1013 \*trace
1014 \fl@trace{***Start addtotoporbot}%
1015 \*trace
1016 \@getfpsbit \tw@
1017 \*trace
1018 \fl@trace{fpstype \ifodd \@tempcnta OK \else not \fi top:
1019 \the \@fpstype}%
1020 \*trace
1021 \ifodd \@tempcnta
1022 \flsetnum \@topnum
1023 \ifnum \@topnum>\z@
1024 \@tempswafalse
1025 \flcheckspace \@toproom \@toplist
1026 \if@tempswa
1027 \@bitor\@currtype{\@midlist\@botlist}%
1028 \*trace
1029 \fl@trace{(mid+bot)list: \@midlist, \@botlist:
1030 (addtotoporbot-before)}%
1031 \*trace

```

```

1032         \if@test
1033     <*trace>
1034         \fl@trace{type already on list: mid or bot---sent to addtobot}%
1035     </trace>
1036     \else
1037         \@flupdates \@topnum \@toproom \@toplist
1038     <*trace>
1039         \fl@trace{colroom (after-top) = \the \@colroom}%
1040         \fl@trace{colnum (after-top) = \the \@colnum}%
1041         \fl@trace{topnum (after-top) = \the \@topnum}%
1042         \fl@trace{***Success: top}%
1043     </trace>
1044         \@inserttrue
1045     \fi
1046 \fi
1047 <*trace>
1048     \else
1049         \fl@trace{Fail: topnum = \the \@topnum: fpstype
1050                                     \the \@fpstype=ORD?}%
1051         \ifnum \@fpstype<\sist@n
1052             \fl@trace{ERROR: !t float not successful (addtotoporbot)}%
1053         \fi
1054     </trace>
1055     \fi
1056 \fi
1057 \if@insert
1058     \else
1059 <*trace>
1060     \fl@trace{sent to addtobot (addtotoporbot)}%
1061 </trace>
1062     \@addtobot
1063 \fi
1064 }
1065 </2ekernel | fltrace>

```

(End definition for \@addtotoporbot.)

\@addtocurcol Lots of changes.

```

1066 <latexrelease | fltrace | flafter>\IncludeInRelease{2015/01/01}%
1067 <latexrelease | fltrace | flafter> {\@addtocurcol}{float order in 2-column}%
1068 <*2ekernel | latexrelease | fltrace | flafter>
1069 \def \@addtocurcol {%
1070 <*trace>
1071     \fl@trace{***Start addtocurcol}%
1072 </trace>
1073     \@insertfalse
1074     \@setfloatypecounts
1075     \ifnum \@fpstype=8
1076 <*trace>
1077         \fl@trace{fpstype !p only (addtocurcol): \the \@fpstype = 8?}%
1078 </trace>
1079     \else
1080         \ifnum \@fpstype=24
1081 <*trace>

```

```

1082         \fl@trace{fpstype p only (addtocurcol): \the \@fpstype = 24?}%
1083     </trace>
1084         \else
1085             \@flsettextmin

```

This is a new adjustment which is quite a major change in functionality; but it implements the documentation. Note that `\@reqcolroom` will include the whole of the page-so-far, and hence includes `\@textfloatsheight` of floats, so before comparing it with `\@textmin`, we add this to `\@textmin` also.

```

1086 <*trace>
1087     \fl@trace{textfloatsheight (before) = \the \@textfloatsheight}%
1088 </trace>
1089     \advance \@textmin \@textfloatsheight
1090     \@reqcolroom \@pageht

```

This line must be removed since `\@specialoutput` changed.

```

1091 %         \advance \@reqcolroom \@pagedp
1092 <*trace>
1093     \fl@trace{textmin + textfloatsheight: \the \@textmin}%
1094     \fl@trace{page-so-far: \the \@reqcolroom}%
1095 </trace>
1096     \ifdim \@textmin>\@reqcolroom
1097         \@reqcolroom \@textmin
1098 <*trace>
1099     \fl@trace{ORD? textmin being used}%
1100 </trace>
1101     \fi
1102     \advance \@reqcolroom \ht\@currbox
1103 <*trace>
1104     \fl@trace{float size = \the \ht \@currbox (addtocurcol)}%
1105     \fl@trace{colroom = \the \@colroom (addtocurcol)}%
1106     \fl@trace{reqcolroom = \the \@reqcolroom (addtocurcol)}%
1107 </trace>
1108     \ifdim \@colroom>\@reqcolroom
1109         \@flsetnum \@colnum
1110         \ifnum \@colnum>\z@
1111             \@bitor\@currtype\@deferlist

```

We need to defer the float also if its width doesn't fit.

```

1112         \@testwrongwidth\@currbox
1113 <*trace>
1114         \fl@trace{deferlist: \@deferlist: (addtocurcol-before)}%
1115 </trace>
1116         \if@test
1117 <*trace>
1118             \fl@trace{type already on list: defer (addtocurcol)}%
1119 </trace>
1120         \else
1121             \@bitor\@currtype\@botlist
1122 <*trace>
1123             \fl@trace{botlist: \@botlist: (addtocurcol-before)}%
1124 </trace>
1125         \if@test
1126 <*trace>

```

```

1127         \fl@trace{type already on list: bot---sent to addtobot}%
1128     </trace>
1129     \@addtobot
1130 \else
1131 <*trace>
1132     \fl@trace{fpstype \ifodd \@tempcnta OK \else not \fi
1133         here: \the \@fpstype}%
1134 </trace>
1135     \ifodd \count\@currbox
1136     \advance \@reqcolroom \intextsep
1137     \ifdim \@colroom>\@reqcolroom
1138     \global \advance \@colnum \m@ne
1139     \global \advance \@textfloatsheight \ht\@currbox

```

This may sometimes give an overestimate.

```

1140     \global \advance \@textfloatsheight 2\intextsep
1141     \@cons \@midlist \@currbox
1142 <*trace>
1143     \fl@trace{***Success: here}%
1144     \fl@trace{textfloatsheight (after-here) =
1145         \the \@textfloatsheight}%
1146     \fl@trace{colnum (after-here) = \the \@colnum}%
1147 </trace>

```

CHANGE TO \@addtocurcol:

\penalty\z@ changed to \penalty\interlinepenalty so \samepage works properly with figure and table environments. (Changed 23 Oct 86)

There is also an \addpenalty\interlinepenalty above.

Since in 2e \samepage is no longer supported, these could be removed.

Although it is best to use \addvspace in case two h floats come together, this makes other spacing more difficult to adjust; whereas if a user specifies two h floats together then they can more easily get the spacing correct by ad hoc commands.

It is necessary to adjust for the addition of \parskip here in case the float is added between paragraphs (i.e. when in vertical mode).

If the nobreak switch is true we need to reset it and clear \everypar since the float may not reset the flag and cannot reset the \everypar globally.

Typesetting starts here (we are in vertical mode).

```

1148     \if@nobreak
1149     \nobreak
1150     \@nobreakfalse
1151     \everypar{}%
1152 \else
1153     \addpenalty \interlinepenalty
1154 \fi
1155     \vskip \intextsep
1156     \box\@currbox
1157     \penalty\interlinepenalty
1158     \vskip\intextsep
1159     \ifnum\outputpenalty <-\@Mii \vskip -\parskip\fi

```

Typesetting ends here.

```

1160     \outputpenalty \z@
1161     \@inserttrue
1162 <*trace>

```

```

1163             \else
1164             \fl@trace{Fail---no room at 2nd test of colroom
1165                     (addtocurcol \string\intextsep)}%
1166         </trace>
1167         \fi
1168     \fi
1169     \if@insert
1170     \else
1171     <*2ekernel | fltrace | latexrelease>
1172     <*trace>
1173         \fl@trace{not here: sent to addtotoporbot}%
1174     </trace>
1175     \@addtotoporbot
1176     </2ekernel | fltrace | latexrelease>
1177     <*!2ekernel&!fltrace&!latexrelease>
1178     <*trace>
1179         \fl@trace{not here: sent to addtobot}%
1180     </trace>
1181     \@addtobot
1182     </!2ekernel&!fltrace&!latexrelease>
1183     \fi
1184     \fi
1185     \fi
1186     <*trace>
1187     \else
1188         \fl@trace{Fail: colnum = \the \@colnum:
1189                 fpstype \the \@fpstype=ORD?}%
1190         \ifnum \@fpstype<\sist@n
1191             \fl@trace{ERROR: BANG float not successful (addtocurcol)}%
1192             \fi
1193     </trace>
1194     \fi
1195     <*trace>
1196     \else
1197         \fl@trace{Fail---no room: fl box ht: \the \ht \@currbox
1198                 (addtocurcol)}%
1199     </trace>
1200     \fi
1201     \fi
1202     \fi
1203     \if@insert
1204     \else
1205         \@resetfyps
1206     <*trace>
1207         \fl@trace{put on deferlist (addtocurcol)}%
1208     </trace>
1209     \@cons\@deferlist\@currbox
1210     <*trace>
1211         \fl@trace{deferlist: \@deferlist: (addtocurcol-after)}%
1212     </trace>

```

```

1213 \fi
1214 }%
1215 </2ekernel | latexrelease | fltrace | flafter>
1216 <latexrelease | fltrace | flafter>\EndIncludeInRelease
1217 <latexrelease | fltrace | flafter>\IncludeInRelease{0000/00/00}%
1218 <latexrelease | fltrace | flafter> {\@addtocurcol}{float order in 2-column}%
1219 <latexrelease | fltrace | flafter>\def \@addtocurcol {%
1220 *trace>
1221 <latexrelease | fltrace | flafter> \fl@trace{***Start addtocurcol}%
1222 </trace>
1223 <latexrelease | fltrace | flafter> \insertfalse
1224 <latexrelease | fltrace | flafter> \setfloattypecounts
1225 <latexrelease | fltrace | flafter> \ifnum \@fpstype=8
1226 *trace>
1227 <latexrelease | fltrace | flafter> \fl@trace{fpstype !p only (addtocurcol):
1228 <latexrelease | fltrace | flafter> \the \@fpstype = 8?}%
1229 </trace>
1230 <latexrelease | fltrace | flafter> \else
1231 <latexrelease | fltrace | flafter> \ifnum \@fpstype=24
1232 *trace>
1233 <latexrelease | fltrace | flafter> \fl@trace{fpstype p only (addtocurcol):
1234 <latexrelease | fltrace | flafter> \the \@fpstype = 24?}%
1235 </trace>
1236 <latexrelease | fltrace | flafter> \else
1237 <latexrelease | fltrace | flafter> \flsettextmin

```

This is a new adjustment which is quite a major change in functionality; but it implements the documentation. Note that \@reqcolroom will include the whole of the page-so-far, and hence includes \@textfloatsheight of floats, so before comparing it with \@textmin, we add this to \@textmin also.

```

1238 *trace>
1239 <latexrelease | fltrace | flafter> \fl@trace{textfloatsheight (before) =
1240 <latexrelease | fltrace | flafter> \the \@textfloatsheight}%
1241 </trace>
1242 <latexrelease | fltrace | flafter> \advance \@textmin \@textfloatsheight
1243 <latexrelease | fltrace | flafter> \@reqcolroom \@pageht

```

This line must be removed since \@specialoutput changed.

```

1244 % \advance \@reqcolroom \@pagedp
1245 *trace>
1246 <latexrelease | fltrace | flafter> \fl@trace{textmin + textfloatsheight:
1247 <latexrelease | fltrace | flafter> \the \@textmin}%
1248 <latexrelease | fltrace | flafter> \fl@trace{page-so-far: \the \@reqcolroom}%
1249 <latexrelease | fltrace | flafter>
1250 </trace>
1251 <latexrelease | fltrace | flafter> \ifdim \@textmin>\@reqcolroom
1252 <latexrelease | fltrace | flafter> \@reqcolroom \@textmin
1253 *trace>
1254 <latexrelease | fltrace | flafter> \fl@trace{ORD? textmin being used}%
1255 </trace>
1256 <latexrelease | fltrace | flafter> \fi
1257 <latexrelease | fltrace | flafter> \advance \@reqcolroom \ht\@currbox
1258 *trace>
1259 <latexrelease | fltrace | flafter> \fl@trace{float size =
1260 <latexrelease | fltrace | flafter> \the \ht \@currbox (addtocurcol)}%

```

```

1261 <latexrelease | fltrace | flafter> \fl@trace{colroom =
1262 <latexrelease | fltrace | flafter> \the \@colroom (addtocurcol)}%
1263 <latexrelease | fltrace | flafter> \fl@trace{reqcolroom =
1264 <latexrelease | fltrace | flafter> \the \@reqcolroom (addtocurcol)}%
1265 </trace>
1266 <latexrelease | fltrace | flafter> \ifdim \@colroom>\@reqcolroom
1267 <latexrelease | fltrace | flafter> \@flsetnum \@colnum
1268 <latexrelease | fltrace | flafter> \ifnum \@colnum>\z@
1269 <latexrelease | fltrace | flafter> \@bitor\@currtype\@deferlist
1270 <*trace>
1271 <latexrelease | fltrace | flafter> \fl@trace{deferlist:
1272 <latexrelease | fltrace | flafter> \@deferlist: (addtocurcol-before)}%
1273 </trace>
1274 <latexrelease | fltrace | flafter> \if@test
1275 <*trace>
1276 <latexrelease | fltrace | flafter> \fl@trace{type already on list:
1277 <latexrelease | fltrace | flafter> defer (addtocurcol)}%
1278 </trace>
1279 <latexrelease | fltrace | flafter> \else
1280 <latexrelease | fltrace | flafter> \@bitor\@currtype\@botlist
1281 <*trace>
1282 <latexrelease | fltrace | flafter> \fl@trace{botlist: \@botlist:
1283 <latexrelease | fltrace | flafter> (addtocurcol-before)}%
1284 </trace>
1285 <latexrelease | fltrace | flafter> \if@test
1286 <*trace>
1287 <latexrelease | fltrace | flafter> \fl@trace{type already on list:
1288 <latexrelease | fltrace | flafter> bot---sent to addtobot}%
1289 </trace>
1290 <latexrelease | fltrace | flafter> \@addtobot
1291 <latexrelease | fltrace | flafter> \else
1292 <*trace>
1293 <latexrelease | fltrace | flafter> \fl@trace{fpstype
1294 <latexrelease | fltrace | flafter> \ifodd \@tempcnta OK \else not \fi
1295 <latexrelease | fltrace | flafter> here: \the \@fpstype}%
1296 </trace>
1297 <latexrelease | fltrace | flafter> \ifodd \count\@currbox
1298 <latexrelease | fltrace | flafter> \advance \@reqcolroom \intextsep
1299 <latexrelease | fltrace | flafter> \ifdim \@colroom>\@reqcolroom
1300 <latexrelease | fltrace | flafter> \global \advance \@colnum \m@ne
1301 <latexrelease | fltrace | flafter> \global \advance
1302 <latexrelease | fltrace | flafter> \@textfloatsheight\ht\@currbox
1303 <latexrelease | fltrace | flafter> \global \advance
1304 <latexrelease | fltrace | flafter> \@textfloatsheight 2\intextsep
1305 <latexrelease | fltrace | flafter> \@cons \@midlist \@currbox
1306 <*trace>
1307 <latexrelease | fltrace | flafter> \fl@trace{***Success: here}%
1308 <latexrelease | fltrace | flafter> \fl@trace{textfloatsheight
1309 <latexrelease | fltrace | flafter> (after-here) =
1310 <latexrelease | fltrace | flafter> \the \@textfloatsheight}%
1311 <latexrelease | fltrace | flafter> \fl@trace{colnum (after-here) =
1312 <latexrelease | fltrace | flafter> \the \@colnum}%
1313 </trace>

```

This may sometimes give an overestimate.

CHANGE TO \@addtocurcol:

\penalty\z@ changed to \penalty\interlinepenalty so \samepage works properly with figure and table environments. (Changed 23 Oct 86)

There is also an \addpenalty\interlinepenalty above.

Since in 2e \samepage is no longer supported, these could be removed.

Although it is best to use \addvspace in case two h floats come together, this makes other spacing more difficult to adjust; whereas if a user specifies two h floats together then they can more easily get the spacing correct by ad hoc commands.

It is necessary to adjust for the addition of \parskip here in case the float is added between paragraphs (i.e. when in vertical mode).

If the nobreak switch is true we need to reset it and clear \everypar since the float may not reset the flag and cannot reset the \everypar globally.

Typesetting starts here (we are in vertical mode).

```

1314 <latexrelease | fltrace | flafter>          \if@nobreak
1315 <latexrelease | fltrace | flafter>          \nobreak
1316 <latexrelease | fltrace | flafter>          \@nobreakfalse
1317 <latexrelease | fltrace | flafter>          \everypar{}%
1318 <latexrelease | fltrace | flafter>          \else
1319 <latexrelease | fltrace | flafter>          \addpenalty\interlinepenalty
1320 <latexrelease | fltrace | flafter>          \fi
1321 <latexrelease | fltrace | flafter>          \vskip \intextsep
1322 <latexrelease | fltrace | flafter>          \box\@currbox
1323 <latexrelease | fltrace | flafter>          \penalty\interlinepenalty
1324 <latexrelease | fltrace | flafter>          \vskip\intextsep
1325 <latexrelease | fltrace | flafter>          \ifnum\outputpenalty
1326 <latexrelease | fltrace | flafter>          <-\@Mii \vskip
1327 <latexrelease | fltrace | flafter>          -\parskip\fi

```

Typesetting ends here.

```

1328 <latexrelease | fltrace | flafter>          \outputpenalty \z@
1329 <latexrelease | fltrace | flafter>          \@inserttrue
1330 <*trace>
1331 <latexrelease | fltrace | flafter>          \else
1332 <latexrelease | fltrace | flafter>          \fl@trace{Fail---no room at 2nd test of colroom
1333 <latexrelease | fltrace | flafter>          (addtocorcol \string\intextsep)}%
1334 </trace>
1335 <latexrelease | fltrace | flafter>          \fi
1336 <latexrelease | fltrace | flafter>          \fi
1337 <latexrelease | fltrace | flafter>          \if@insert
1338 <latexrelease | fltrace | flafter>          \else

```

Next set of docstrip guards are a bit weird, essentially \@addtotoporbot ends up inside the kernel and the fltrace package and \@addtotoporbot shows up in the flafter package. Guess that could have been done a bit more obvious :-)

```

1339 <*2ekernel | fltrace>
1340 <*trace>
1341 <latexrelease | fltrace | flafter>          \fl@trace{not here: sent to addtotoporbot}%
1342 </trace>
1343 <latexrelease | fltrace | flafter>          \@addtotoporbot
1344 </2ekernel | fltrace>
1345 <!*2ekernel&!autoload&!fltrace>
1346 <*trace>
1347 <latexrelease | fltrace | flafter>          \fl@trace{not here: sent to addtobot}%

```



```

1348 </trace>
1349 <latexrelease | fltrace | flafter> \addtobot
1350 </!2kernel&!autoload&!fltrace>
1351 <latexrelease | fltrace | flafter> \fi
1352 <latexrelease | fltrace | flafter> \fi
1353 <latexrelease | fltrace | flafter> \fi
1354 <*trace>
1355 <latexrelease | fltrace | flafter> \else
1356 <latexrelease | fltrace | flafter> \fl@trace{Fail: colnum = \the \@colnum:
1357 <latexrelease | fltrace | flafter> fpstype \the \@fpstype=ORD?}%
1358 <latexrelease | fltrace | flafter> \ifnum \@fpstype<\sist@n
1359 <latexrelease | fltrace | flafter> \fl@trace{ERROR: BANG float not successful
1360 <latexrelease | fltrace | flafter> (addtocurcol)}}%
1361 <latexrelease | fltrace | flafter> \fi
1362 </trace>
1363 <latexrelease | fltrace | flafter> \fi
1364 <*trace>
1365 <latexrelease | fltrace | flafter> \else
1366 <latexrelease | fltrace | flafter> \fl@trace{Fail---no room: fl box ht:
1367 <latexrelease | fltrace | flafter> \the \ht \@currbox (addtocurcol)}}%
1368 </trace>
1369 <latexrelease | fltrace | flafter> \fi
1370 <latexrelease | fltrace | flafter> \fi
1371 <latexrelease | fltrace | flafter> \fi
1372 <latexrelease | fltrace | flafter> \if@insert
1373 <latexrelease | fltrace | flafter> \else
1374 <latexrelease | fltrace | flafter> \@resetfps
1375 <*trace>
1376 <latexrelease | fltrace | flafter> \fl@trace{put on deferlist (addtocurcol)}}%
1377 </trace>
1378 <latexrelease | fltrace | flafter> \@cons\@deferlist\@currbox
1379 <*trace>
1380 <latexrelease | fltrace | flafter> \fl@trace{deferlist: \@deferlist:
1381 <latexrelease | fltrace | flafter> (addtocurcol-after)}}%
1382 </trace>
1383 <latexrelease | fltrace | flafter> \fi
1384 <latexrelease | fltrace | flafter> }%
1385 <latexrelease | fltrace | flafter> \EndIncludeInRelease

```

(End definition for \addtocurcol.)

\@addtonextcol Lots of changes.

```

1386 <latexrelease | fltrace> \IncludeInRelease{2015/01/01}
1387 <latexrelease | fltrace> {\@addtonextcol}{float order in 2-column}%
1388 <*2kernel | latexrelease | fltrace>
1389 \def\@addtonextcol{%
1390 \begingroup
1391 <*trace>
1392 \fl@trace{***Start addtonextcol}%
1393 </trace>
1394 \@insertfalse
1395 \@setfloattypecounts
1396 \ifnum \@fpstype=8
1397 <*trace>

```

```

1398     \fl@trace{fpstype not curcol: \the \@fpstype = 8?}%
1399 </trace>
1400     \else
1401         \ifnum \@fpstype=24
1402 <*trace>
1403         \fl@trace{fpstype not curcol: \the \@fpstype = 24?}%
1404 </trace>
1405     \else
1406         \@flsettextmin
1407 <*trace>
1408         \fl@trace{text-so-far: Opt (top of col)}%
1409 </trace>
1410         \@reqcolroom \ht\@currbox
1411 <*trace>
1412         \fl@trace{float size: \the \@reqcolroom (addtonextcol)}%
1413 </trace>
1414         \advance \@reqcolroom \@textmin
1415 <*trace>
1416         \fl@trace{colroom = \the \@colroom (addtonextcol)}%
1417         \fl@trace{reqcolroom = \the \@reqcolroom (addtonextcol)}%
1418 </trace>
1419         \ifdim \@colroom>\@reqcolroom
1420             \@flsetnum \@colnum
1421             \ifnum\@colnum>\z@
1422                 \@bitor\@currtype\@deferlist
1423 <*trace>
1424                 \fl@trace{deferlist: \@deferlist: (addtonextcol-before)}%
1425 </trace>
1426             \@testwrongwidth\@currbox
1427             \if@test
1428 <*trace>
1429                 \fl@trace{type already on list: defer (addtonextcol)}%
1430 </trace>
1431             \else
1432 <*trace>
1433                 \fl@trace{sent to addtotoporbot (addtonextcol)}%
1434 </trace>
1435                 \@addtotoporbot
1436             \fi
1437         \fi
1438 <*trace>
1439         \else
1440             \fl@trace{Fail---no room: fl box ht: \the \ht \@currbox
1441                                     (addtonextcol)}%
1442 </trace>
1443         \fi
1444     \fi
1445     \fi
1446     \if@insert
1447     \else
1448 <*trace>
1449         \fl@trace{put back on deferlist (addtonextcol)}%
1450 </trace>

```

```

1451 \cons\@deferlist\@currbox
1452 <*trace>
1453 \fl@trace{deferlist: \@deferlist: (addtonextcol-after)}%
1454 </trace>
1455 \fi
1456 <*trace>
1457 \fl@trace{End of addtonextcol -- locally counts:}%
1458 \fl@trace{col: \the\@colnum. top: \the \@topnum. bot: \the \@botnum.}%
1459 </trace>
1460 \endgroup
1461 <*trace>
1462 \fl@trace{End of addtonextcol -- globally counts:}%
1463 \fl@trace{col: \the\@colnum. top: \the \@topnum. bot: \the \@botnum.}%
1464 </trace>
1465 }%
1466 </2ekernel | latexrelease | fltrace>
1467 <latexrelease | fltrace>\EndIncludeInRelease
1468 <latexrelease | fltrace>\IncludeInRelease{0000/00/00}%
1469 <latexrelease | fltrace> {\@addtonextcol}{float order in 2-column}%
1470 <latexrelease | fltrace>\def\@addtonextcol{%
1471 <latexrelease | fltrace> \begingroup
1472 <*trace>
1473 <latexrelease | fltrace> \fl@trace{***Start addtonextcol}%
1474 </trace>
1475 <latexrelease | fltrace> \@insertfalse
1476 <latexrelease | fltrace> \@setfloattyperecounts
1477 <latexrelease | fltrace> \ifnum \@fpstype=8
1478 <*trace>
1479 <latexrelease | fltrace> \fl@trace{fpstype not curcol:
1480 <latexrelease | fltrace> \the \@fpstype = 8?}%
1481 </trace>
1482 <latexrelease | fltrace> \else
1483 <latexrelease | fltrace> \ifnum \@fpstype=24
1484 <*trace>
1485 <latexrelease | fltrace> \fl@trace{fpstype not curcol:
1486 <latexrelease | fltrace> \the \@fpstype = 24?}%
1487 </trace>
1488 <latexrelease | fltrace> \else
1489 <latexrelease | fltrace> \@flsettextmin
1490 <*trace>
1491 <latexrelease | fltrace> \fl@trace{text-so-far: Opt (top of col)}%
1492 </trace>
1493 <latexrelease | fltrace> \@reqcolroom \ht\@currbox
1494 <*trace>
1495 <latexrelease | fltrace> \fl@trace{float size:
1496 <latexrelease | fltrace> \the \@reqcolroom (addtonextcol)}%
1497 <latexrelease | fltrace>
1498 </trace>
1499 <latexrelease | fltrace> \advance \@reqcolroom \@textmin
1500 <*trace>
1501 <latexrelease | fltrace> \fl@trace{colroom =
1502 <latexrelease | fltrace> \the \@colroom (addtonextcol)}%
1503 <latexrelease | fltrace> \fl@trace{reqcolroom =
1504 <latexrelease | fltrace> \the \@reqcolroom (addtonextcol)}%

```

```

1505 </trace>
1506 <latexrelease | fltrace> \ifdim \@colroom>\@reqcolroom
1507 <latexrelease | fltrace> \@flsetnum \@colnum
1508 <latexrelease | fltrace> \ifnum \@colnum>\z@
1509 <latexrelease | fltrace> \@bitor \@currtype \@deferlist
1510 <*trace>
1511 <latexrelease | fltrace> \fl@trace{deferlist: \@deferlist:
1512 <latexrelease | fltrace> (addtonextcol-before)}%
1513 </trace>
1514 <latexrelease | fltrace> \if@test
1515 <*trace>
1516 <latexrelease | fltrace> \fl@trace{type already on list:
1517 <latexrelease | fltrace> defer (addtonextcol)}%
1518 </trace>
1519 <latexrelease | fltrace> \else
1520 <*trace>
1521 <latexrelease | fltrace> \fl@trace{sent to addtotoporbot
1522 <latexrelease | fltrace> (addtonextcol)}%
1523 </trace>
1524 <latexrelease | fltrace> \@addtotoporbot
1525 <latexrelease | fltrace> \fi
1526 <latexrelease | fltrace> \fi
1527 <*trace>
1528 <latexrelease | fltrace> \else
1529 <latexrelease | fltrace> \fl@trace{Fail---no room: fl box ht:
1530 <latexrelease | fltrace> \the \ht \@currbox (addtonextcol)}%
1531 </trace>
1532 <latexrelease | fltrace> \fi
1533 <latexrelease | fltrace> \fi
1534 <latexrelease | fltrace> \fi
1535 <latexrelease | fltrace> \if@insert
1536 <latexrelease | fltrace> \else
1537 <*trace>
1538 <latexrelease | fltrace> \fl@trace{put back on deferlist
1539 <latexrelease | fltrace> (addtonextcol)}%
1540 </trace>
1541 <latexrelease | fltrace> \@cons \@deferlist \@currbox
1542 <*trace>
1543 <latexrelease | fltrace> \fl@trace{deferlist: \@deferlist:
1544 <latexrelease | fltrace> (addtonextcol-after)}%
1545 </trace>
1546 <latexrelease | fltrace> \fi
1547 <*trace>
1548 <latexrelease | fltrace> \fl@trace{End of addtonextcol --
1549 <latexrelease | fltrace> locally counts:}%
1550 <latexrelease | fltrace> \fl@trace{col: \the \@colnum.
1551 <latexrelease | fltrace> top: \the \@topnum. bot: \the \@botnum.}%
1552 </trace>
1553 <latexrelease | fltrace> \endgroup
1554 <*trace>
1555 <latexrelease | fltrace> \fl@trace{End of addtonextcol --
1556 <latexrelease | fltrace> globally counts:}%
1557 <latexrelease | fltrace> \fl@trace{col: \the \@colnum.
1558 <latexrelease | fltrace> top: \the \@topnum. bot: \the \@botnum.}%

```

```

1559 </trace>
1560 <latexrelease | fltrace>}%
1561 <latexrelease | fltrace>\EndIncludeInRelease

```

(End definition for \@addtonextcol.)

\@addtodblcol Lots of changes.

```

1562 <latexrelease | fltrace>\IncludeInRelease{2015/01/01}%
1563 <latexrelease | fltrace> {\@addtodblcol}{float order in 2-column}%
1564 <*2ekernel | latexrelease | fltrace>
1565 \def\@addtodblcol{%
1566   \begingroup
1567   <*trace>
1568   \fl@trace{***Start addtodblcol}%
1569   </trace>
1570   \@insertfalse
1571   \@setfloatypecounts
1572   \@getfpsbit \tw@
1573   <*trace>
1574   \fl@trace{fpstype \ifodd \@tempcnta OK \else not \fi dbltop:
1575                                           \the \@fpstype}%
1576   </trace>
1577   \ifodd\@tempcnta
1578     \@flsetnum \@dbltopnum
1579     \ifnum \@dbltopnum>\z@
1580       \@tempswafalse
1581       \ifdim \@dbltoproom>\ht\@currbox
1582         \@tempwattrue
1583   <*trace>
1584     \fl@trace{Space OK: \@dbltoproom =
1585               \the \@dbltoproom > \the \ht \@currbox
1586               (dbltoproom)}%
1587   </trace>
1588   \else
1589   <*trace>
1590     \fl@trace{fpstype: \the \@fpstype (addtodblcol)}%
1591   </trace>
1592     \ifnum \@fpstype<\sist@@n
1593   <*trace>
1594     \fl@trace{BANG float ignoring \@dbltoproom}%
1595     \fl@trace{\@spaces \@dbltoproom = \the \@dbltoproom.
1596               Ht float: \the \ht \@currbox-BANG}%
1597   </trace>

```

Need to check that there is room on the page, using the local value of \@textmin to make the necessary adjustment to \@dbltoproom.

```

1598     \advance \@dbltoproom \@textmin
1599   <*trace>
1600     \fl@trace{Local value of texmin: \the\@textmin}%
1601     \fl@trace{\@spaces space on page = \the \@dbltoproom.
1602               Ht float: \the \ht \@currbox-BANG}%
1603   </trace>
1604     \ifdim \@dbltoproom>\ht\@currbox
1605       \@tempwattrue

```

```

1606 <*trace>
1607         \fl@trace{Space OK BANG: space on page =
1608                 \the \@dbltoproom > \the \ht \@currbox}%
1609     \else
1610         \fl@trace{fpstype: \the \@fpstype}%
1611         \fl@trace{Fail---no room dbltoproom-BANG?:}%
1612         \fl@trace{\@spaces space on page = \the \@dbltoproom.
1613                 Ht float: \the \ht \@currbox}%
1614 </trace>
1615     \fi
1616     \advance \@dbltoproom -\@textmin
1617 <*trace>
1618     \else
1619         \fl@trace{fpstype: \the \@fpstype}%
1620         \fl@trace{Fail---no room dbltoproom-ORD?:}%
1621         \fl@trace{\@spaces \@dbltoproom = \the \@dbltoproom.
1622                 Ht float: \the \ht \@currbox}%
1623 </trace>
1624     \fi
1625     \fi
1626     \if@tempswa
1627         \@bitor \@currtype \@deferlist
1628 <*trace>
1629         \fl@trace{(dbl)deferlist: \@deferlist: (before)}%
1630 </trace>
1631     not in fixfloats?
1632     \@testwrongwidth\@currbox
1633 <*trace>
1634         \fl@trace{type already on list: (dbl)defer}%
1635 </trace>
1636     \else
1637         \@tempdima -\ht\@currbox
1638         \advance\@tempdima
1639             -\ifx \@dbltoplist\@empty \dbltextfloatsep \else
1640                 \dblfloatsep \fi
1641         \global \advance \@dbltoproom \@tempdima
1642         \global \advance \@colht \@tempdima
1643         \global \advance \@dbltopnum \m@ne
1644         \@cons \@dbltoplist \@currbox
1645 <*trace>
1646         \fl@trace{dbltopnum (after) = \the \@dbltopnum}%
1647         \fl@trace{***Success: dbltop}%
1648 </trace>
1649         \@inserttrue
1650     \fi
1651 \fi
1652 <*trace>
1653 \else
1654     \fl@trace{Fail: dbltopnum = \the \@dbltopnum: fpstype
1655             \the \@fpstype=ORD?}%
1656     \ifnum \@fpstype<\sist@n
1657         \fl@trace{ERROR: !t float not successful (addtodblcol)}%

```

```

1658         \fi
1659     </trace>
1660     \fi
1661     \fi
1662     \if@insert
1663     \else
1664     <*trace>
1665         \fl@trace{put on deferlist}%
1666     </trace>
1667     \@cons\@deferlist\@currbox
1668     <*trace>
1669         \fl@trace{(dbl)deferlist: \@deferlist: (after)}%
1670     </trace>
1671     \fi
1672     <*trace>
1673         \fl@trace{End of addtodblcol -- locally count:}%
1674         \fl@trace{ dbltop: \the \@dbltopnum.}%
1675     </trace>
1676     \endgroup
1677     <*trace>
1678         \fl@trace{End of addtodblcol -- globally count:}%
1679         \fl@trace{dbltop: \the \@dbltopnum.}%
1680     </trace>
1681     }%
1682     </2ekernel | latexrelease | fltrace>
1683     <latexrelease | fltrace>\EndIncludeInRelease
1684     <latexrelease | fltrace>\IncludeInRelease{0000/00/00}%
1685     <latexrelease | fltrace> {\@addtodblcol}{float order in 2-column}%
1686     <latexrelease | fltrace>\def\@addtodblcol{%
1687     <latexrelease | fltrace> \begingroup
1688     <*trace>
1689     <latexrelease | fltrace> \fl@trace{***Start addtodblcol}%
1690     </trace>
1691     <latexrelease | fltrace> \insertfalse
1692     <latexrelease | fltrace> \@setfloattypecounts
1693     <latexrelease | fltrace> \@getfpsbit \tw@
1694     <*trace>
1695     <latexrelease | fltrace> \fl@trace{fpstype \ifodd \@tempcnta OK
1696     <latexrelease | fltrace> \else not \fi dbltop: \the \@fpstype}%
1697     </trace>
1698     <latexrelease | fltrace> \ifodd\@tempcnta
1699     <latexrelease | fltrace> \flsetnum \@dbltopnum
1700     <latexrelease | fltrace> \ifnum \@dbltopnum>\z@
1701     <latexrelease | fltrace> \@tempwafalse
1702     <latexrelease | fltrace> \ifdim \@dbltoproom>\ht\@currbox
1703     <latexrelease | fltrace> \@tempwattrue
1704     <*trace>
1705     <latexrelease | fltrace> \fl@trace{Space OK: \@dbltoproom =
1706     <latexrelease | fltrace> \the \@dbltoproom > \the \ht \@currbox
1707     <latexrelease | fltrace> (dbltoproom)}%
1708     </trace>
1709     <latexrelease | fltrace> \else
1710     <*trace>
1711     <latexrelease | fltrace> \fl@trace{fpstype: \the \@fpstype (addtodblcol)}%

```

```

1712 </trace>
1713 <latexrelease | fltrace> \ifnum \@fpstype<\sist@@n
1714 <*trace>
1715 <latexrelease | fltrace> \fl@trace{BANG float ignoring \@dbltoproom}%
1716 <latexrelease | fltrace> \fl@trace{\@spaces \@dbltoproom =
1717 <latexrelease | fltrace> \the \@dbltoproom.
1718 <latexrelease | fltrace> Ht float: \the \ht \@currbox-BANG}%
1719 </trace>

```

Need to check that there is room on the page, using the local value of \@textmin to make the necessary adjustment to \@dbltoproom.

```

1720 <latexrelease | fltrace> \advance \@dbltoproom \@textmin
1721 <*trace>
1722 <latexrelease | fltrace> \fl@trace{Local value of texmin: \the\@textmin}%
1723 <latexrelease | fltrace> \fl@trace{\@spaces space on page =
1724 <latexrelease | fltrace> \the \@dbltoproom.
1725 <latexrelease | fltrace> Ht float: \the \ht \@currbox-BANG}%
1726 </trace>
1727 <latexrelease | fltrace> \ifdim \@dbltoproom>\ht\@currbox
1728 <latexrelease | fltrace> \@tempswatrue
1729 <*trace>
1730 <latexrelease | fltrace> \fl@trace{Space OK BANG: space on page =
1731 <latexrelease | fltrace> \the\@dbltoproom > \the\ht\@currbox}%
1732 <latexrelease | fltrace> \else
1733 <latexrelease | fltrace> \fl@trace{fpstype: \the \@fpstype}%
1734 <latexrelease | fltrace> \fl@trace{Fail---no room dbltoproom-BANG?:}%
1735 <latexrelease | fltrace> \fl@trace{\@spaces space on page =
1736 <latexrelease | fltrace> \the \@dbltoproom.
1737 <latexrelease | fltrace> Ht float: \the \ht \@currbox}%
1738 </trace>
1739 <latexrelease | fltrace> \fi
1740 <latexrelease | fltrace> \advance \@dbltoproom -\@textmin
1741 <*trace>
1742 <latexrelease | fltrace> \else
1743 <latexrelease | fltrace> \fl@trace{fpstype: \the \@fpstype}%
1744 <latexrelease | fltrace> \fl@trace{Fail---no room dbltoproom-ORD?:}%
1745 <latexrelease | fltrace> \fl@trace{\@spaces \@dbltoproom =
1746 <latexrelease | fltrace> \the \@dbltoproom.
1747 <latexrelease | fltrace> Ht float: \the \ht \@currbox}%
1748 </trace>
1749 <latexrelease | fltrace> \fi
1750 <latexrelease | fltrace> \fi
1751 <latexrelease | fltrace> \if@tempswa
1752 <latexrelease | fltrace> \@bitor \@currtype \@dbldeferlist
1753 <*trace>
1754 <latexrelease | fltrace> \fl@trace{dbldeferlist:
1755 <latexrelease | fltrace> \@dbldeferlist: (before)}%
1756 </trace>
1757 <latexrelease | fltrace> \if@test
1758 <*trace>
1759 <latexrelease | fltrace> \fl@trace{type already on list: dbldefer}%
1760 </trace>
1761 <latexrelease | fltrace> \else
1762 <latexrelease | fltrace> \@tempdima -\ht\@currbox

```



```

1763 <latexrelease | fltrace> \advance\@tempdima
1764 <latexrelease | fltrace> -\ifx \@dbltoplist\@empty
1765 <latexrelease | fltrace> \dbltextfloatsep
1766 <latexrelease | fltrace> \else \dblfloatsep \fi
1767 <latexrelease | fltrace> \global \advance \@dbltoproom \@tempdima
1768 <latexrelease | fltrace> \global \advance \@colht \@tempdima
1769 <latexrelease | fltrace> \global \advance \@dbltopnum \m@ne
1770 <latexrelease | fltrace> \@cons \@dbltoplist \@currbox
1771 <*trace>
1772 <latexrelease | fltrace> \fl@trace{dbltopnum (after) =
1773 <latexrelease | fltrace> \the \@dbltopnum}%
1774 <latexrelease | fltrace> \fl@trace{***Success: dbltop}%
1775 </trace>
1776 <latexrelease | fltrace> \@inserttrue
1777 <latexrelease | fltrace> \fi
1778 <latexrelease | fltrace> \fi
1779 <*trace>
1780 <latexrelease | fltrace> \else
1781 <latexrelease | fltrace> \fl@trace{Fail: dbltopnum = \the \@dbltopnum:
1782 <latexrelease | fltrace> fpstype \the \@fpstype=ORD?}%
1783 <latexrelease | fltrace> \ifnum \@fpstype<\sist@@n
1784 <latexrelease | fltrace> \fl@trace{ERROR: !t float not successful
1785 <latexrelease | fltrace> (addtodblcol)}%
1786 <latexrelease | fltrace> \fi
1787 </trace>
1788 <latexrelease | fltrace> \fi
1789 <latexrelease | fltrace> \fi
1790 <latexrelease | fltrace> \if@insert
1791 <latexrelease | fltrace> \else
1792 <*trace>
1793 <latexrelease | fltrace> \fl@trace{put on dbldeferlist}%
1794 </trace>
1795 <latexrelease | fltrace> \@cons\@dbldeferlist\@currbox
1796 <*trace>
1797 <latexrelease | fltrace> \fl@trace{dbldeferlist: \@dbldeferlist: (after)}%
1798 </trace>
1799 <latexrelease | fltrace> \fi
1800 <*trace>
1801 <latexrelease | fltrace> \fl@trace{End of addtodblcol -- locally count:}%
1802 <latexrelease | fltrace> \fl@trace{ dbltop: \the \@dbltopnum.}%
1803 </trace>
1804 <latexrelease | fltrace> \endgroup
1805 <*trace>
1806 <latexrelease | fltrace> \fl@trace{End of addtodblcol -- globally count:}%
1807 <latexrelease | fltrace> \fl@trace{dbltop: \the \@dbltopnum.}%
1808 </trace>
1809 <latexrelease | fltrace>}%
1810 <latexrelease | fltrace>\EndIncludeInRelease

```

(End definition for \@addtodblcol.)

\@addmarginpar

```

1811 <*2ekernel>
1812 \def\@addmarginpar{\@next\@marbox\@currlist{\@cons\@freelist\@marbox

```

```

1813 \cons\@freelist\@currbox}\@latexbug\@tempcnta\@ne
1814 \if@twocolumn
1815   \if@firstcolumn \@tempcnta\m@ne \fi
1816 \else
1817   \if@mparswitch
1818     \ifodd\c@page \else\@tempcnta\m@ne \fi
1819   \fi
1820   \if@reversemargin \@tempcnta -\@tempcnta \fi
1821 \fi
1822 \ifnum\@tempcnta <\z@ \global\setbox\@marbox\box\@currbox \fi
1823 \@tempdima\@mparbottom
1824 \advance\@tempdima -\@pageht
1825 \advance\@tempdima\ht\@marbox
1826 \ifdim\@tempdima >\z@
1827   \@latex@warning@no@line {Marginpar on page \thepage\space moved}%
1828 \else
1829   \@tempdima\z@
1830 \fi
1831 \global\@mparbottom\@pageht
1832 \global\advance\@mparbottom\@tempdima
1833 \global\advance\@mparbottom\dp\@marbox
1834 \global\advance\@mparbottom\marginparpush
1835 \advance\@tempdima -\ht\@marbox

```

Putting box movement inside the ‘marbox’:

```

1836 \global\setbox \@marbox
1837   \vbox {\vskip \@tempdima
1838         \box \@marbox}%
1839 \global \ht\@marbox \z@
1840 \global \dp\@marbox \z@

```

Sticking (rather than gluing:-) the ‘marbox’ to the line above, changed vskip to kern:

```

1841 \kern -\@pagedp
1842 \nointerlineskip
1843 \hb@xt@\columnwidth
1844   {\ifnum \@tempcnta >\z@
1845     \hskip\columnwidth \hskip\marginparsep
1846   \else
1847     \hskip -\marginparsep \hskip -\marginparwidth
1848   \fi
1849   \box\@marbox \hss}%

```

For this reason the following code can vanish:

```

\ nobreak           %% No longer needed.  CAR92/12
\ vskip -\@tempdima %% No longer needed.  CAR92/12

1850 \nointerlineskip
1851 \hbox{\vrule \@height\z@ \@width\z@ \@depth\@pagedp}}

```

(End definition for \@addmarginpar.)

1.1.1 Kludgeins

This part of the file is part of the implementation of the following two new commands for L^AT_EX2_ε.

`\enlargethispage{<dim>}`

Adds <dim> to the height of the current column only. On the printed page the bottom of this column is extended downwards by exactly <dim> without having any effect on the placement of the footer; this may result in an overprinting.

`\enlargethispage*{<dim>}`

Similar to `\enlargethispage` but it tries to squeeze the column to be printed in as small a space as possible, ie it uses any shrinkability in the column. If the column was not explicitly broken (e.g. with `\pagebreak`) this may result in an overfull box message but except for this it will come out as expected (if you know what to expect).

The star form of this command is dedicated to Leslie Lamport, the other we need for ourselves (FMi, CAR).

These commands may well have unwanted effects if used soon before a `\clearpage`: please give keep them clear of such places.

`\@kludgeins` The insert which makes T_EX do a lot of the necessary work. All we need to put into it is the amount by which the pagegoal should be changed.

```
1852 \newinsert \@kludgeins
1853 \global\dimen\@kludgeins \maxdimen
1854 \global\count\@kludgeins 1000
```

(End definition for \@kludgeins.)

`\enlargethispage` The user command.

```
\enlargethispage* 1855 \gdef \enlargethispage {%
1856     \@ifstar
1857     {%
1858     <*trace>
1859         \fl@trace{Enlarging page height * }%
1860     </trace>
1861         \@enlargepage{\hbox{\kern\p@}}}%
1862     {%
1863     <*trace>
1864         \fl@trace{Enlarging page height exactly---}%
1865     </trace>
1866         \@enlargepage\@empty}%
1867 }
```

(End definition for \enlargethispage and \enlargethispage.)*

`\@enlargepage` This actually inserts the insert, after checking for extreme values of the change.

```
1868 \gdef\@enlargepage#1#2{%
1869 <*trace>
1870     \fl@trace{\@spaces\@spaces by #2}%
1871 </trace>
1872     \@tempskipa#2\relax
1873     \ifdim \@tempskipa>.5\maxdimen
1874         \@latex@error{Suggested\space extra\space height\space
1875             (\the\@tempskipa)\space dangerously\space
1876             large}\@eha
1877     \else
1878         \ifdim \vsize<.5\maxdimen
```

```

1879 <*trace>
1880 \fl@trace {Kludgeins added--pagegoal before: \the\pagegoal}%
1881 </trace>
1882 \@bsphack
1883 \insert\@kludgeins{#1\vskip-\@tempskipa}%
1884 \@esphack
This next bit is for tracing only:
1885 <*trace>
1886 \ifvmode \par
1887 \fl@trace {Kludgeins added--pagegoal after: \the \pagegoal}%
1888 \fi
1889 </trace>
1890 \else
1891 \latex@error{Page\space height\space already\space
1892 too\space large}\@eha
1893 \fi
1894 \fi
1895 }
1896 </2ekernel>

```

(End definition for \@enlargepage.)

1.1.2 Float control

This part implements controllable floats and other changes to the float mechanism.

It provides, at the document level, the following command for inclusion in L^AT_EX2_ε.

`\suppressfloats`

This suppresses all further floats on the current page.

With an optional argument it suppresses only floats only in certain positions on the current page.

[t] suppresses only floats at the top of the page [b] suppresses only floats at the bottom of the page

It also enables the use of an extra specifier, !, in the location optional argument of a float. If this is present then, just for this particular float, whenever it is processed by the float mechanism the following are ignored:

- all restrictions on the number of floats which can appear;
- all explicit restrictions on the amount of space which should (not) be occupied by floats and/or text.

The mechanism will still attempt to ensure that pages are not overfull.

These specifiers override, for the single float, the suppression commands described above.

In its current form, it also supplies a reasonably exhaustive, and somewhat baroque, means of tracing some aspects of the float mechanism.

More tracing.

`\fl@trace` Set-up tracing for floats independent of other tracing as it produces mega-output. Default
`\tracefloatsoff` is no tracing.

```

1897 \tracefloats <*\fltrace>
1898 \fl@traceval \def \fl@tracemessage #1{{\let\@elt\@empty\typeout{LaTeX2e: #1}}}
1899 \tracefloatvals \def \tracefloats{\let \fl@trace \fl@tracemessage}
1900 \fl@tracemessage \def \tracefloatsoff {\let \fl@trace \@gobble}
1901 \tracefloatsoff
1902 \def \fl@traceval #1{\fl@trace{\string #1 = \the #1}}
1903 \IncludeInRelease{2015/01/01}{\tracefloatvals}%
1904 {trace float vals}%
1905 \def \tracefloatvals{%

```

As `\@dblfloatplacement` sets `\f@depth` it needs to be run inside a group, otherwise the float placement will test for the wrong value.⁴¹

```

1906 \begingroup
1907 \@dblfloatplacement
1908 \@floatplacement
1909 \fl@trace{***Float placement parameters:}%
1910 \fl@traceval\@colnum
1911 \fl@traceval\@colroom
1912 \fl@traceval\@topnum
1913 \fl@traceval\@toproom
1914 \fl@traceval\@botnum
1915 \fl@traceval\@botroom
1916 \fl@traceval\@fpmin
1917 \fl@trace{\string\textfraction = \textfraction}%
1918 \fl@traceval\@dbltopnum
1919 \fl@traceval\@dbltoproom
1920 \fl@trace{\string\textfraction = \textfraction}%
1921 \fl@trace{toplist: \@toplist}%
1922 \fl@trace{botlist: \@botlist}%
1923 \fl@trace{midlist: \@midlist}%
1924 \fl@trace{deferlist: \@deferlist}%
1925 \fl@trace{dbltoplist: \@dbltoplist}%
1926 %FMI \fl@trace{dbldeferlist: \@dbldeferlist}%
1927 \endgroup
1928 }
1929 \EndIncludeInRelease
1930 \IncludeInRelease{0000/00/00}{\tracefloatvals}%
1931 {trace float vals}%
1932 \def \tracefloatvals{%
1933 \begingroup
1934 \@dblfloatplacement
1935 \@floatplacement
1936 \fl@trace{***Float placement parameters:}%
1937 \fl@traceval\@colnum
1938 \fl@traceval\@colroom
1939 \fl@traceval\@topnum
1940 \fl@traceval\@toproom
1941 \fl@traceval\@botnum
1942 \fl@traceval\@botroom
1943 \fl@traceval\@fpmin

```

⁴¹This is a somewhat questionable design.

```

1944 \fl@trace{\string\textfraction = \textfraction}%
1945 \fl@traceval\@dbltopnum
1946 \fl@traceval\@dbltoproom
1947 \fl@trace{\string\textfraction = \textfraction}%
1948 \fl@trace{toplist: \@toplist}%
1949 \fl@trace{botlist: \@botlist}%
1950 \fl@trace{midlist: \@midlist}%
1951 \fl@trace{deferlist: \@deferlist}%
1952 \fl@trace{dbltoplist: \@dbltoplist}%
1953 % next line only in old releases
1954 \fl@trace{dbldeferlist: \@dbldeferlist}%
1955 \endgroup
1956 }
1957 \EndIncludeInRelease

```

We need to make sure that `fltrace` comes before `flafter` to make the tracing work.

```

1958 \ifpackageloaded{flafter}
1959 {
1960   \PackageWarningNoLine{fltrace}{Load 'fltrace' before 'flafter'\MessageBreak
1961     Attempting to recover by reloading 'flafter'}%
1962   \expandafter\let\csname ver@flafter.sty\endcsname\relax
1963   \def\reserved@a#1{%
1964     \expandafter\let\csname string#1+flafter+IIR\endcsname\relax}%
1965   \reserved@a\@addtocurcol
1966   \reserved@a\@addtonextcol
1967   \RequirePackage{flafter}}{}
1968 \fltrace

```

As the code for `flafter` will contain tracing calls so that it works in conjunction with `fltrace` we need to provide a dummy definition for `\fl@trace` in that package.

```

1969 \*flafter
1970 \providecommand\fl@trace[1]{}
1971 \flafter

```

(End definition for \fl@trace and others.)

\suppressfloats Float suppression commands: these set the relevant counter globally to zero. Thus they
\@flstop are overridden for a particular float by an `!` specifier.

```

1972 \*2kernel
1973 \def \suppressfloats {%
1974   \ifnextchar [%
1975     \@flstop
1976     {\global \@colnum \z@}%
1977   }

```

Maybe this should be a loop over `#1`?

```

1978 \def \@flstop [#1]{%
1979   \if t#1%
1980     \global \@topnum \z@
1981   \fi
1982   \if b#1%
1983     \global \@botnum \z@
1984   \fi
1985 }

```

(End definition for `\suppressfloats` and `\flstop`.)

Manipulation of float placement and type; both their strings and the corresponding count registers.

`\@fpstype` First a new count register to go with `\@currtype`.
`\@reqcolroom` Then a new skip register, for information needed to remove the `\@maxsep` conservatism: it is possible that this could use a temporary register.
`\@textfloatsheight` Finally a dimension register to hold the total height of in-text floats on the current page. This is needed to implement a major change in the functionality of `\@addtocurcol` which is, nevertheless, a bug fix. It is not local and therefore cannot be a temporary register.

```
1986 \newcount \@fpstype
1987 \newdimen \@reqcolroom
1988 \newdimen \@textfloatsheight
1989 \endkernel
```

(End definition for `\@fpstype`, `\@reqcolroom`, and `\@textfloatsheight`.)

`\@fpsadddefault` Adds the default placement to what is already there.
Should not need to change this, but could do it as follows:

```
def \@fpsadddefault {%
  \@temptokena \expandafter\expandafter\expandafter
               {\csname fps@\@capytype \endcsname}%
  \edef \reserved@a {\the\@temptokena}%
  \@onelevel@sanitize \reserved@a
  \edef \@fps {\@fps\reserved@a}%

1990 \endkernel | fltrace)
1991 \def \@fpsadddefault {%
1992 \tracing
1993   \fl@trace{fps changed from: \@fps}%
1994 \tracing
1995   \edef \@fps {\@fps\csname fps@\@capytype \endcsname}%
1996   \@latex@warning {%
1997     No positions in optional float specifier.\MessageBreak
1998     Default added (so using '\@fps')}%
1999 }
```

(End definition for `\@fpsadddefault`.)

`\@setfloattypecounts` Sets counters `\@fpstype` and `\@currtype`.
BANG == bit4 of `\count\@currbox` = 0.

```
2000 \def \@setfloattypecounts {%
2001   \@currtype \count\@currbox
2002   \@fpstype \count\@currbox
2003   \divide\@currtype\@xxxii \multiply\@currtype\@xxxii
2004   \advance \@fpstype -\@currtype
2005 \tracing
2006   \fl@trace{(mod 32) fpstype: \the \@fpstype}%
2007   \fl@trace{(mult of 32) currtype: \the \@currtype}%
2008 % Tracing only: but some should be changed into real errors/warnings?
```

```

2009 \ifnum \@fpstype<\sist@n
2010 \ifnum \@fpstype=\z@
2011 \fl@trace{ERROR: no PLACEMENT, fpstype = \the \@fpstype = 0?}%
2012 \fi
2013 \ifnum \@fpstype=\@ne
2014 \fl@trace{WARNING: only h, fpstype = \the \@fpstype = 1?}%
2015 \fi
2016 \fl@trace{BANG float}%
2017 \else
2018 \ifnum \@fpstype=\sist@n
2019 \fl@trace{ERROR: no PLACEMENT, fpstype = \the \@fpstype = 16?}%
2020 \fi
2021 \ifnum \@fpstype=17
2022 \fl@trace{WARNING: only h, fpstype = \the \@fpstype = 17?}%
2023 \fi
2024 \fl@trace{ORD float}%
2025 \fi
2026 \</trace>
2027 }
2028 \</2ekernel | fltrace>

```

(End definition for \@setfloattypescounts.)

Macros for getting, testing and setting bits of the fps.

\@getfpsbit Sets \@tempcnta to required bit of \count\@currbox.

```

2029 \<*2ekernel>
2030 \def \@getfpsbit {%
2031 \boxfpsbit \@currbox
2032 }

```

(End definition for \@getfpsbit.)

\@boxfpsbit Used above.

```

2033 \def \@boxfpsbit #1#2{%
2034 \@tempcnta \count#1%
2035 \divide \@tempcnta #2\relax
2036 }

```

(End definition for \@boxfpsbit.)

\@testfp New definition of the float page test.

```

2037 \def \@testfp #1{%
2038 \@boxfpsbit #18\relax % Really ‘#1 8’ for human readers!
2039 \ifodd \@tempcnta
2040 \else
2041 \@testtrue
2042 \fi
2043 }

```

(End definition for \@testfp.)

`\@setfpsbit` Sets required bit of `\@tempcnta` (to 1).

```
2044 \def \@setfpsbit #1{%
2045   \@tempcntb \@tempcnta
2046   \divide \@tempcntb #1\relax
2047   \ifodd \@tempcntb
2048     \else
2049       \advance \@tempcnta #1\relax
2050     \fi
2051 }
2052 \</2ekernel>
```

(End definition for \@setfpsbit.)

`\@resethfps` Globally adds t as a possible location for an h or !h only placement: this must be done using the count.

Although it will leave `\@fpstype` set to 17 even if it was originally 1, this does not matter since it is the last thing in `\@addtocurcol`.

```
2053 \<*2ekernel|fltrace>
2054 \def \@resethfps {%
2055   \let\reserved@a\@empty
2056   \ifnum \@fpstype=\@ne
2057     \def \reserved@a {!}%
2058     \@fpstype 17
2059   \fi
2060   \ifnum \@fpstype=17
2061     \global \advance \count\@currbox \tw@
2062     \@latex@warning@no@line {%
2063       '\reserved@a h' float specifier changed to '\reserved@a ht'}%
2064   \<*trace>
2065     \fl@trace{%
2066       't' added to '\reserved@a h'- new Count: \the \count\@currbox}%
2067   \</trace>
2068   \fi
2069 }
```

(End definition for \@resethfps.)

Special stuff for BANG floats.

`\@flsetnum` Ignores any zero float counter value in case BANG.

It uses a local assignment to the normally global counter: a bit naughty, perhaps?

These assignments are safe so long as the counter involved is only consulted once (i.e. only for the 'bang float') with the changed value. This is the case within `\@addtocurcol` because it is used only once within a call of the output routine (which forms a group).

For `\@addtonextcol` this is achieved by putting a group around its code; this is needed because it is called (by `\@startcolumn`) for each float which was on the deferlist. Almost identical considerations pertain to `\@addtodblcol`. There may be more efficient ways to handle this, but the group seems to be the simplest.

```
2070 \def \@flsetnum #1{%
2071   \<*trace>
2072     \fl@trace{fpstype: \the \@fpstype (flsetnum \string#1)}%
2073   \</trace>
2074   \ifnum \@fpstype<\sxt@@n
```

```

2075     \ifnum #1=\z@
2076     <*trace>
2077         \fl@trace{BANG float resetting \string#1 to 1}%
2078     </trace>
2079         #1\@ne
2080     \fi
2081 \fi
2082 <*trace>
2083     \fl@trace{#1 (before) = \the #1}%
2084 </trace>
2085 }

```

(End definition for \@flsetnum.)

\@flsettextmin This ignores \textfraction space restriction in case BANG.

```

2086 \def \@flsettextmin {%
2087 <*trace>
2088     \fl@trace{fpstype: \the \@fpstype (flsettextmin)}%
2089 </trace>
2090     \ifnum \@fpstype<\sist@n
2091 <*trace>
2092         \fl@trace{BANG ignoring textmin}%
2093 </trace>
2094         \@textmin \z@
2095     \else
2096         \@textmin \textfraction\@colht
2097 <*trace>
2098         \fl@trace{ORD textmin = \the \@textmin}%
2099 </trace>
2100     \fi
2101 }

```

(End definition for \@flsettextmin.)

\@flcheckspace This ignores space restriction in case BANG; this is still slightly conservative since it does not allow for the fact that, if there is no text in the column then \textfloatsep is not needed. Sets @tempswa true if there is room for \@currbox.

```

2102 \def \@flcheckspace #1#2{%
2103     \advance \@reqcolroom
2104     \ifx #2\@empty \textfloatsep \else \floatsep \fi
2105 <*trace>
2106     \fl@trace{colroom = \the \@colroom
2107                                     (flcheckspace \string#1 \string#2)}%
2108     \fl@trace{reqcolroom = \the \@reqcolroom
2109                                     (flcheckspace \string#1 \string#2)}%
2110 </trace>
2111     \ifdim \@colroom>\@reqcolroom
2112         \ifdim #1>\ht\@currbox
2113             \@tempwattrue
2114 <*trace>
2115             \fl@trace{Space OK: #1 = \the #1 > \the \ht \@currbox
2116                                     (flcheckspace \string#1 \string#2)}%
2117 </trace>
2118         \else

```

```

2119 <*trace>
2120     \fl@trace{fpstype: \the \@fpstype
2121                                     (flcheckspace \string#1 \string#2)}%
2122 </trace>
2123     \ifnum \@fpstype<\sixt@n
2124 <*trace>
2125     \fl@trace{BANG float ignoring #1
2126                                     (flcheckspace \string#1 \string#2):}%
2127     \fl@trace{\@spaces #1 = \the #1. Ht float: \the \ht \@currbox
2128                                     BANG}%
2129 </trace>
2130     \@tempwattrue
2131 <*trace>
2132     \else
2133     \fl@trace{Fail---no room (flcheckspace \string#1 \string#2)
2134                                     (fpstype \the \@fpstype=ORD?):}%
2135     \fl@trace{\@spaces #1 = \the #1. Ht float: \the \ht \@currbox
2136                                     ORD?:}%
2137 </trace>
2138     \fi
2139     \fi
2140 <*trace>
2141     \else
2142     \fl@trace{Fail---no room at 2nd test of colroom
2143                                     (flcheckspace \string#1 \string#2)}%
2144 </trace>
2145     \fi
2146 }
2147 </2ekernel | fltrace>

(End definition for \@flcheckspace.)

```

\@flupdates This updates everything when a float is placed.

```

2148 <*2ekernel>
2149 \def \@flupdates #1#2#3{%
2150     \global \advance #1\m@ne
2151     \global \advance \@colnum \m@ne
2152     \@tempdima -\ht\@currbox
2153     \advance \@tempdima
2154     -\ifx #3\@empty \textfloatsep \else \floatsep \fi
2155     \global \advance #2\@tempdima
2156     \global \advance \@colroom \@tempdima
2157     \@cons #3\@currbox
2158 }
2159 </2ekernel>

```

(End definition for \@flupdates.)

Interesting facts about float mechanisms past and present, together with a summary of various features, some unresolved:

1. The value `\textfraction` does not affect the processing of doublecol floats: this seems sensible, but should be documented.
2. `\twocolumn floatplacement` was wrong: `dbl` not needed, `ord` needed.

3. `\@floatplacement` was not called after `\@startdblcol` or `\@topnewpage`. This has been changed; it is clearly a bug fix.
4. The use `\@topnewpage` when `\dblfigrule` is non-trivial produced a rule in the wrong place. This has been fixed by not using `\dblfigrule` when processing the ‘float’ from `\@topnewpage`.
5. If the specifier was just `h` and the float could not be put here, it went on the deferlist and stayed there until a clearpage. It now gets changed to a ‘th’: this is only an error-recovery action, putting just `h` or `!h` should be deprecated.
6. `\@dblmaxsep` was ‘the maximum of `\dblfloatsep` and `\dbltextfloatsep`’. But it was never used! Now gone completely, like `\@maxsep`.
7. After an `h` float is put on a page, it was counted as text when applying the `\textfraction` test; this is possibly too big a change although it is a bug fix?
8. Two consecutive `h` floats are separated by twice `\intextsep`: this could be changed to one by use of `\addvspace`, OK? Note that it would also mean that less space is put in if an `h` float immediately follows other spaces. This is also possibly too big a change, at least for compatibility mode? Or it may be simply wrong! It has not been changed.
9. Now `\@addtocurcol` checks first for just `p` fps. I think that this is an increase in efficiency, but maybe the coding should be made even more efficient.
10. `\@tryfcolumn` now tests if the list is empty first, otherwise lots of wasted time! Thus this test has been removed from `\@startcolumn`. As Frank pointed out, this makes `\@startcolumn` less efficient. But it is now the same as `\@startdblcolumn`: I can see no reason why they should be different, but which is best?
11. Why is `\@colroom` set in `\@doclearpage`?
12. Footnotes. Check what `\clearpage` does when footnotes are left over. Footnotes are not put on float pages and, also, `\@addtonextcol` ignores the existence of held-over footnotes in deciding what floats can go on the page. Not changed.
13. `\clearpage` can still lose non-boxes, at least when floats are involved. It also moves some to the ‘wrong page’, but this may be a coding problem.
14. The `!` option makes it necessary to check in `\output` that there is enough room left on the page after adding a float. (This would have been necessary anyway if anyone set `\@textmin` too close to zero! A similar danger existed also if the text in a `\twocolumn[text]` entity gets too large.) The current implementation of this also makes the normal case a little less efficient, OK? Not enough room means, at present, less than `\baselineskip`, with a warning: is this OK? Should it be made generic (another parameter)?
15. There are four possibilities for supporting this:
`\twocolumn[\maketitle more text]`
One is to change `\maketitle` slightly to allow this. Another is to change `\@topnewpage` so that more than one `\twocolumn[]` command is allowed; in this case `\maketitle\twocolumn[more text]` will work. The former is more robust

from the user's viewpoint, but makes the code for `\maketitle` rather ad hoc (maybe it is already?). Another is to misuse the global `twocolumn` flag locally within `\@topnewpage`. Yet another is to move the column count register from the `multicol` package into the kernel. This has been done.

16. Where should the reinserts be put to maximise the probability that footnotes come out on the correct page? Or should we go for as much compatibility as possible (but see next item)?
17. Should we continue to support (as much as possible) `\samepage`? Some of its intended functionality is now advertised as being provided by `\enlargethispage`. Use of either is likely to result in wrongly placed footnotes, marginals, etc. Which should have priority: obeying the pagination instructions, or correct placement of notes/marginalia?
18. Is the adjustment of space to cause shrinking in the kludge-* case correct? Should it be limited to 0pt?
19. Is the setting of `\boxmaxdepth` in `makecol` and friends needed? It only has any effect if `\@textbottom` ends with a box or rule, in which case the `vskip` to allow for its depth should also be added. If it is kept, it should probably be the last thing in the box. It has now been removed.

It would perhaps be better to document that `\@textbottom` and `\@texttop` must have natural height 0pt.
20. I cannot see why the `vskip` adjustment for the depth is needed if `boxmaxdepth` is used to ensure that there is never a too deep box.
21. The value of `\boxmaxdepth` should be explicitly set whenever necessary: it is too risky to assume that it has any particular value. Care is needed in deciding what to set it to.

It is interesting to note that the value of `\boxmaxdepth` is unique in being read before the local settings for the box group are reset; all other parameter settings which affect the box construction use their values outside the box group.
22. Should `\@maxdepth` store the setting of `\maxdepth` from `lplain`? Or should we provide a proper interface to class files for setting these?

An analysis of various other macros.

`\@opcol` should do `\@floatplacement`, but where? Right at the end, since it always occurs at the start of a column.

```
\def\@opcol{%
% Why is this done first?
\global \@mparbottom \z@
\if@twocolumn
\@outputdblcol
\else
\@outputpage
% This is not needed since it is done at the end of
% |\@outputpage|:
\global \@colht \textheight
\fi}
```

Only tracing has been added to these.

```

2160 <latexrelease | fltrace>\IncludeInRelease{2017/01/01}%
2161 <latexrelease | fltrace> {\@makefcolumn}{negative height floats}%
2162 <*2ekernel | fltrace | latexrelease>
2163 \def\@makefcolumn #1{%
2164   \begingroup
2165   \@fpmin -\maxdimen
2166   \let \@testfp \@gobble
2167   \@tryfcolumn #1%
2168   \endgroup
2169 <*trace>
2170   \if@fcolmade
2171     \fl@trace{PAGE: in \string\clearpage
2172               \if@twocolumn ---twocolumn\fi---}%
2173     \fl@trace{----- float column/page completed from \string#1}%
2174     \fi
2175 </trace>
2176 }
2177 <latexrelease | fltrace>\EndIncludeInRelease
2178 <latexrelease | fltrace>\IncludeInRelease{0000/00/00}%
2179 <latexrelease | fltrace> {\@makefcolumn}{negative height floats}%
2180 <latexrelease | fltrace>\def\@makefcolumn #1{%
2181 <latexrelease | fltrace>   \begingroup
2182 <latexrelease | fltrace>   \@fpmin \z@
2183 <latexrelease | fltrace>   \let \@testfp \@gobble
2184 <latexrelease | fltrace>   \@tryfcolumn #1%
2185 <latexrelease | fltrace>   \endgroup
2186 <*trace>
2187 <latexrelease | fltrace>   \if@fcolmade
2188 <latexrelease | fltrace>     \fl@trace{PAGE: in \string\clearpage
2189 <latexrelease | fltrace>               \if@twocolumn ---twocolumn\fi---}%
2190 <latexrelease | fltrace>     \fl@trace{----- float column/page completed
2191 <latexrelease | fltrace>               from \string#1}%
2192 <latexrelease | fltrace>   \fi
2193 </trace>
2194 <latexrelease | fltrace> }
2195 <latexrelease | fltrace>\EndIncludeInRelease
2196 </2ekernel | fltrace | latexrelease>

```

This will line up the last baselines in the two columns provided they are constructed in the normal way: i.e. ending in a skip of minus the original depth, with `\@textbottom` adding nothing.

Thus again it is essential for `\@textbottom` to have depth 0pt.

```

2197 <latexrelease | fltrace>\IncludeInRelease{2015/01/01}%
2198 <latexrelease | fltrace> {\@outputdblcol}{2 column marks}%
2199 <*2ekernel | fltrace | latexrelease>

```

This is just a change to the single command `\@outputdblcol` so that it saves mark information for the first column and restores it in the second column.

```

2200 \def\@outputdblcol{%
2201   \if@firstcolumn
2202     \global\@firstcolumnfalse

```

Save the left column

```
2203 \global\setbox\@leftcolumn\copy\@outputbox
2204 \fl@trace{PAGE: first column boxed}%
```

Remember the marks from the first column

```
2205 \splitmaxdepth\maxdimen
2206 \vbadness\maxdimen
```

In case of `\enlargethispage` we will have infinite negative glue at the bottom of the page (coming from `\vss`) and that will earn us an error message if we `\vsplit` to get at the marks. So we need to remove the last glue (if any) at the end of `\@outputbox` as we are only interested in marks that change doesn't matter.

```
2207 \setbox\@outputbox\vbox{\unvbox\@outputbox\unskip}%
2208 \setbox\@outputbox\vsplit\@outputbox to\maxdimen
```

One minor difference from the current `fixmarks` package, pass the marks through a token register to stop any `#` tokens causing an error in a `\def`.

```
2209 \toks@\expandafter{\topmark}%
2210 \xdef\@firstcoltopmark{\the\toks@}%
2211 \toks@\expandafter{\splitfirstmark}%
2212 \xdef\@firstcolfirstmark{\the\toks@}%
```

This test does not work if truly empty marks have been inserted, but \LaTeX marks should always have (at least) two brace groups. (Except before the first mark is used, when the marks are empty, but that is OK here.)

```
2213 \ifx\@firstcolfirstmark\@empty
2214 \global\let\@setmarks\relax
2215 \else
2216 \gdef\@setmarks{%
2217 \let\firstmark\@firstcolfirstmark
2218 \let\topmark\@firstcoltopmark}%
2219 \fi
```

End of change

```
2220 \else
2221 \global\@firstcolumntrue
2222 \setbox\@outputbox\vbox{%
2223 \hb@xt@\textwidth{%
2224 \hb@xt@\columnwidth{\box\@leftcolumn \hss}%
2225 \hfil
```

The color of the `\vrule` should be `\normalcolor` as to not inherit the color from the column.

```
2226 {\normalcolor\vrule \@width\columnseprule}%
2227 \hfil
2228 \hb@xt@\columnwidth{\box\@outputbox \hss}}}%
2229 \fl@trace{PAGE: second column also boxed}%
2230 \@combinedblfloats
```

Override current first and top with those of first column if necessary

```
2231 \setmarks
```

End of change

```
2232 \outputpage
2233 \fl@trace{PAGE: two column page completed}%
2234 \begingroup
```

```

2235 \dblfloatplacement
2236 \startdblcolumn
2237 \@whiles\if@fcolmade \fi{\@outputpage
2238 <fltrace> \fl@trace{PAGE: double float page completed}%
2239 \startdblcolumn}%
2240 \endgroup
2241 \fi}%

2242 <latexrelease | fltrace>\EndIncludeInRelease
2243 <latexrelease | fltrace>\IncludeInRelease{0000/00/00}%
2244 <latexrelease | fltrace> {\@outputdblcol}{2 column marks}%
2245 <latexrelease | fltrace>\def\@outputdblcol{%
2246 <latexrelease | fltrace> \if@firstcolumn
2247 <latexrelease | fltrace> \global \@firstcolumnfalse
2248 <latexrelease | fltrace> \global \setbox\@leftcolumn \box\@outputbox
2249 <*trace>
2250 <latexrelease | fltrace> \fl@trace{PAGE: first column boxed}%
2251 </trace>
2252 <latexrelease | fltrace> \else
2253 <latexrelease | fltrace> \global \@firstcolumntrue
2254 <latexrelease | fltrace> \setbox\@outputbox \vbox {%
2255 <latexrelease | fltrace> \hb@xt@\textwidth {%
2256 <latexrelease | fltrace> \hb@xt@\columnwidth {%
2257 <latexrelease | fltrace> \box\@leftcolumn \hss}%
2258 <latexrelease | fltrace> \hfil
2259 <latexrelease | fltrace> {\normalcolor\vrule
2260 <latexrelease | fltrace> \@width\columnseprule}%
2261 <latexrelease | fltrace> \hfil
2262 <latexrelease | fltrace> \hb@xt@\columnwidth {%
2263 <latexrelease | fltrace> \box\@outputbox \hss}%
2264 <latexrelease | fltrace> }%
2265 <latexrelease | fltrace> }%
2266 <*trace>
2267 <latexrelease | fltrace> \fl@trace{PAGE: second column also boxed}%
2268 </trace>
2269 <latexrelease | fltrace> \@combinedblfloats
2270 <latexrelease | fltrace> \@outputpage
2271 <*trace>
2272 <latexrelease | fltrace> \fl@trace{PAGE: two column page completed}%
2273 </trace>
2274 <latexrelease | fltrace> \begingroup
2275 <latexrelease | fltrace> \dblfloatplacement
2276 <latexrelease | fltrace> \startdblcolumn

```

This loop could be replaced by an `\expandafter` tail recursion in `\startdblcolumn`.

```

2277 <latexrelease | fltrace> \@whiles\if@fcolmade \fi
2278 <latexrelease | fltrace> {\@outputpage
2279 <*trace>
2280 <latexrelease | fltrace> \fl@trace{PAGE: double float page completed}%
2281 </trace>
2282 <latexrelease | fltrace> \startdblcolumn}%
2283 <latexrelease | fltrace> \endgroup
2284 <latexrelease | fltrace> \fi
2285 <latexrelease | fltrace> }%

```



```

2286 <latexrelease | fltrace>\EndIncludeInRelease
2287 </2ekernel | fltrace | latexrelease>

```

1.1.3 Float placement parameters

The main purpose of this section is to ensure that all the float-placement parameters which need to be set in a class file or package have been declared. It also describes their use and sets values for them which are reasonable for typical documents using US letter or A4 sized paper.

Limits for the placement of floating objects

\c@topnumber This counter holds the maximum number of floats that can appear at the top of a text page or column.

```

2288 <*2ekernel>
2289 \newcount\c@topnumber
2290 \setcounter{topnumber}{2}

(End definition for \c@topnumber.)

```

\topfraction This macro holds the maximum proportion (as a decimal number) of a text page or column that can be occupied by floats at the top.

```

2291 \newcommand\topfraction{.7}

(End definition for \topfraction.)

```

\c@bottomnumber This counter holds the maximum number of floats that can appear at the bottom of a text page or column.

```

2292 \newcount\c@bottomnumber
2293 \setcounter{bottomnumber}{1}

(End definition for \c@bottomnumber.)

```

\bottomfraction This macro holds the maximum proportion (as a decimal number) of a text page or column that can be occupied by floats at the bottom.

```

2294 \newcommand\bottomfraction{.3}

(End definition for \bottomfraction.)

```

\c@totalnumber This counter holds the maximum number of floats that can appear on any text page or column.

```

2295 \newcount\c@totalnumber
2296 \setcounter{totalnumber}{3}

(End definition for \c@totalnumber.)

```

\textfraction This macro holds the minimum proportion (as a decimal number) of a text page or column that must be occupied by text.

```

2297 \newcommand\textfraction{.2}

(End definition for \textfraction.)

```

\floatpagefraction This macro holds the minimum proportion (as a decimal number) of a page or column that must be occupied by floating objects before a ‘float page’ is produced.

```

2298 \newcommand\floatpagefraction{.5}

```

(End definition for `\floatpagefraction`.)

`\c@dbltopnumber` This counter holds the maximum number of double-column floats that can appear on the top of a two-column text page.

2299 `\newcount\c@dbltopnumber`

2300 `\setcounter{dbltopnumber}{2}`

(End definition for `\c@dbltopnumber`.)

`\dbltopfraction` This macro holds the maximum proportion (as a decimal number) of a two-column text page that can be occupied by double-column floats at the top.

2301 `\newcommand\dbltopfraction{.7}`

(End definition for `\dbltopfraction`.)

`\dblfloatpagefraction` This macro holds the minimum proportion (as a decimal number) of a page that must be occupied by double-column floating objects before a ‘double-column float page’ is produced.

2302 `\newcommand\dblfloatpagefraction{.5}`

(End definition for `\dblfloatpagefraction`.)

Floats on a text page

`\floatsep` When a floating object is placed on a page with text, these parameters control the separation between the float and the other objects on the page. These parameters are used for both one-column mode and single-column floats in two-column mode. They are all rubber lengths.

`\floatsep` is the space between adjacent floats that are placed at the top or bottom of the text page or column.

`\textfloatsep` is the space between the main text and floats at the top or bottom of the page or column.

`\intextsep` is the space between in-text floats and the text.

2303 `\newskip\floatsep`

2304 `\newskip\textfloatsep`

2305 `\newskip\intextsep`

2306 `\setlength\floatsep {12\p@ \@plus 2\p@ \@minus 2\p@}`

2307 `\setlength\textfloatsep{20\p@ \@plus 2\p@ \@minus 4\p@}`

2308 `\setlength\intextsep {12\p@ \@plus 2\p@ \@minus 2\p@}`

(End definition for `\floatsep`, `\textfloatsep`, and `\intextsep`.)

`\dblfloatsep` When double-column floats (floating objects that span the whole `\textwidth`) are placed at the top of a text page in two-column mode, the separation between the float and the text is controlled by `\dblfloatsep` and `\dbltextfloatsep`. They are rubber lengths.

`\dblfloatsep` is the space between adjacent double-column floats placed at the top of the text page.

`\dbltextfloatsep` is the space between the main text and double-column floats at the top of the page.

2309 `\newskip\dblfloatsep`

2310 `\newskip\dbltextfloatsep`

2311 `\setlength\dblfloatsep {12\p@ \@plus 2\p@ \@minus 2\p@}`

2312 `\setlength\dbltextfloatsep{20\p@ \@plus 2\p@ \@minus 4\p@}`

(End definition for `\dblfloatsep` and `\dbltextfloatsep`.)

Floats on their own page or column

`\@fptop` When floating objects are placed on a separate page or column, called a ‘float page’, the layout of the page is controlled by these parameters, which are rubber lengths.

`\@fpsep` At the top of the page `\@fptop` is inserted; typically this supplies some stretchable whitespace. At the bottom of the page `\@fpbot` is inserted. Between adjacent floats `\@fpsep` is inserted.

These parameters are used for all floating objects on a ‘float page’ in one-column mode, and for single-column floats in two-column mode.

Note that at least one of the two parameters `\@fptop` and `\@fpbot` should contain a `plus ...fil` so as to fill the remaining empty space.

```
2313 \newskip\@fptop
2314 \newskip\@fpsep
2315 \newskip\@fpbot
2316 \setlength\@fptop{0\p@ \@plus 1fil}
2317 \setlength\@fpsep{8\p@ \@plus 2fil}
2318 \setlength\@fpbot{0\p@ \@plus 1fil}
```

(End definition for \@fptop, \@fpsep, and \@fpbot.)

`\@dblftop` Double-column ‘float pages’ in two-column mode use similar parameters.

```
\@dblfpsep 2319 \newskip\@dblftop
\@dblfpbot 2320 \newskip\@dblfpsep
2321 \newskip\@dblfpbot
2322 \setlength\@dblftop{0\p@ \@plus 1fil}
2323 \setlength\@dblfpsep{8\p@ \@plus 2fil}
2324 \setlength\@dblfpbot{0\p@ \@plus 1fil}
```

(End definition for \@dblftop, \@dblfpsep, and \@dblfpbot.)

`\topfigrule` The macros can be used to put in rules between floats and text; whatever they insert should be vertical mode material which takes up zero space.

```
\botfigrule 2325 \let\topfigrule=\relax
\dblfigrule 2326 \let\botfigrule=\relax
2327 \let\dblfigrule=\relax
2328 \endkernel
```

(End definition for \topfigrule, \botfigrule, and \dblfigrule.)

File W

lthyphen.dtx

This file contains the code for loading hyphenation patterns into L^AT_EX. Most of this will end up in a file called `hyphen.ltx`. If you wish to customize your L^AT_EX system in respect of hyphenation patterns, write a file `hyphen.cfg`. If this file exists, it will be loaded instead of `hyphen.ltx`. See the comments below for additional information.

To produce the printed version of this file the following code is used. It can be extracted with the DOCSTRIP program, or one can run this file directly through L^AT_EX 2_ε.

```
1 <*driver>
2 \documentclass{ltxdoc}
3 \begin{document}
4 \DocInput{lthyphen.dtx}
5 \end{document}
6 </driver>
```

The default file `hyphen.ltx` loads hyphenation patterns for US english. If you want to load additional or other hyphenation patterns, you should create a file `hyphen.cfg`. This is best done by starting from `hyphen.ltx`.

For backward compatibility, the default file, `hyphen.ltx`, first tries to load the file `hyphen.tex`. If this file exists, an information message is issued and the appropriate defaults for T_EX's internal parameters are set: `\language` is initialized to 0, and `\lefthyphenmin` and `\righthyphenmin` to 2 and 3, respectively, to disallow x- or -xx breaks.

```
7 <*default>
8 \InputIfFileExists{hyphen.tex}%
9   {\message{Loading hyphenation patterns for US english.}}%
10   \language=0
11   \lefthyphenmin=2 \righthyphenmin=3 }%
```

Otherwise, since we cannot do anything without any hyphenation patterns, an error message is printed and the IniT_EX run is terminated by invoking `\@@end` (which is the L^AT_EX 2_ε name for T_EX's `\end` primitive).

```
12   {\errhelp{The configuration for hyphenation is incorrectly
13             installed.^^J%
14             If you don't understand this error message you need
15             to seek^^Jexpert advice.}%
16   \errmessage{OOPS! I can't find any hyphenation patterns for
17             US english.^^J \space Think of getting some or the
18             latex2e setup will never succeed}\@@end}
19 </default>
```

The following example describes the possible contents of a file `hyphen.cfg` that will load both US English and German hyphenation patterns, making the former the default. It sets `\language` to 0 for the US patterns and to 1 for the German patterns. Then `\language` is set to 0 to make this the default and the default values of `\lefthyphenmin` and `\righthyphenmin` are set.

```
language=0
input hyphen % (or \input ushyphen1 if the file has been renamed)
language=1
input ghyph31
```

```
language=0
lefthyphenmin=2
righthyphenmin=3
endinput
```

Another possibility is to use the package `babel`, by Johannes Braams. That package is distributed with a suitable `hyphen.cfg` file.

File X

ltxfinal.dtx

1 Final settings

This section contains the final settings for L^AT_EX. It initializes some debugging and typesetting parameters, sets the default `\catcodes` and `uc/lc` codes, and inputs the hyphenation file.

1.1 Debugging

By default, L^AT_EX shows statistics:

```
1 <*2ekernel>
2 \tracingstats1
```

1.2 Typesetting parameters

```
\@lowpenalty These are penalties used internally.
\@medpenalty 3 \newcount\@lowpenalty
\@highpenalty 4 \newcount\@medpenalty
5 \newcount\@highpenalty
```

(End definition for \@lowpenalty, \@medpenalty, and \@highpenalty.)

```
\newmarks Allocate extended marks types if etex is active. Placed here at the end of the format to
increase compatibility with count allocations in earlier releases.
```

```
6 </2ekernel>
7 <*2ekernel | latexrelease>
8 <latexrelease>\IncludeInRelease{2015/01/01}%
9 <latexrelease>          {\newmarks}{Extended Allocation}%
10 \ifx\marks\@undefined\else
11 \def\newmarks{%
12   \e@alloc\marks \e@alloc@chardef{\count256}\m@ne\e@alloc@top}
13 \fi
14 </2ekernel | latexrelease>
15 <latexrelease>\EndIncludeInRelease
16 <latexrelease>\IncludeInRelease{0000/00/00}%
17 <latexrelease>          {\newmarks}{Extended Allocation}%
18 <latexrelease>\let\newmarks\@undefined
19 <latexrelease>\EndIncludeInRelease
20 <*2ekernel>
```

(End definition for \newmarks.)

```
\newXeTeXintercharclass Allocate \XeTeXintercharclass types if xetex is active. previously defined in xetex.ini.
\XeTeXalloc@intercharclass
XeTeXalloc@intercharclass@top 21 </2ekernel>
22 <*2ekernel | latexrelease>
23 <latexrelease>\IncludeInRelease{2015/01/01}%
24 <latexrelease>          {\newXeTeXintercharclass}{Extended Allocation}%
```

Classes allocated 1 to 4094 (or 254 on older xetex) (In earlier XeLaTeX versions 1, 2 and 3 were pre-set for CJK).

```

25 \ifx\XeTeXcharclass\@undefined
26 \else
27 \ifdim\the\XeTeXversion\XeTeXrevision\p@>0.99993\p@
28 \chardef\Xe@alloc@intercharclass@top=4095
29 \else
30 \chardef\Xe@alloc@intercharclass@top=255
31 \fi
32 \def\newXeTeXintercharclass{%
33 \Xe@alloc\XeTeXcharclass
34 \chardef\Xe@alloc@intercharclass\m@ne\Xe@alloc@intercharclass@top}
35 \fi
36 </2ekernel | latexrelease>
37 <latexrelease>\EndIncludeInRelease
38 <latexrelease>\IncludeInRelease{0000/00/00}%
39 <latexrelease> \{newXeTeXintercharclass\}{Extended Allocation}%
40 <latexrelease> \ifx\XeTeXcharclass\@undefined
41 <latexrelease> \else
42 <latexrelease> \def\Xe@alloc@#1#2#3#4#5{\global\advance#1\@ne
43 <latexrelease> \Xe@ch@ck#1#4#2%
44 <latexrelease> \allocationnumber#1%
45 <latexrelease> \global#3#5\allocationnumber
46 <latexrelease> \wlog{\string#5=\string#2\the\allocationnumber}}
47 <latexrelease> \def\Xe@ch@ck#1#2#3{%
48 <latexrelease> \ifnum#1<#2\else
49 <latexrelease> \errmessage{No room for a new #3}%
50 <latexrelease> \fi}
51 <latexrelease> \def\newXeTeXintercharclass{%
52 <latexrelease> \Xe@alloc@\Xe@alloc@intercharclass
53 <latexrelease> \XeTeXcharclass\chardef\@cclv}
54 <latexrelease> \fi
55 <latexrelease>\EndIncludeInRelease
56 <*/2ekernel | latexrelease>
57 <latexrelease>\IncludeInRelease{2016/02/01}%
58 <latexrelease> {\Xe@alloc@intercharclass}{Start of XeTeX class allocator}%
59 \ifx\XeTeXcharclass\@undefined
60 \else
61 \countdef\Xe@alloc@intercharclass=257
62 \Xe@alloc@intercharclass=\z@
63 \fi
64 </2ekernel | latexrelease>
65 <latexrelease>\EndIncludeInRelease
66 <latexrelease>\IncludeInRelease{2015/01/01}%
67 <latexrelease> {\Xe@alloc@intercharclass}{Start of XeTeX class allocator}%
68 <latexrelease> \ifx\XeTeXcharclass\@undefined
69 <latexrelease> \else
70 <latexrelease> \Xe@alloc@intercharclass=\thr@@
71 <latexrelease> \fi
72 <latexrelease>\EndIncludeInRelease
73 <latexrelease>\IncludeInRelease{0000/00/00}%
74 <latexrelease> {\Xe@alloc@intercharclass}{Start of XeTeX class allocator}%
75 <latexrelease> \ifx\XeTeXcharclass\@undefined

```

```

76 <latexrelease> \else
77 <latexrelease> \newcount\xe@alloc@intercharclass
78 <latexrelease> \xe@alloc@intercharclass=\thr@@
79 <latexrelease> \fi
80 <latexrelease>\EndIncludeInRelease
81 <*2ekernel>

```

(End definition for \newXeTeXintercharclass, \xe@alloc@intercharclass, and \e@alloc@intercharclass@top.)

trace_stack_levels Now define the Lua function to emulate \tracingstacklevels and install it in the input_level_string callback.

```

82 </2ekernel>
83 <*2ekernel|latexrelease>

```

In latexrelease mode we always remove the function from the callback, then add the correct version later.

```

84 <latexrelease>\ifx\directlua\@undefined
85 <latexrelease>\else
86 <latexrelease> \directlua{%
87 <latexrelease>   if luatexbase.callbacktypes['input_level_string'] and %
88 <latexrelease>     luatexbase.in_callback('input_level_string','tracingstacklevels') then
89 <latexrelease>       luatexbase.remove_from_callback('input_level_string','tracingstacklevels')
90 <latexrelease>     end}%
91 <latexrelease>\fi
92 <latexrelease>\IncludeInRelease{2021/06/01}{trace_stack_levels}%
93 <latexrelease>      {Lua trace_stack_levels function}%
94 \ifx\directlua\@undefined
95 \else
96 <*2ekernel>
97   \expanded{%
98     \everyjob{\the\everyjob
99       \noexpand%\directlua
100 </2ekernel>
101   \directlua{%
102     local function trace_stack_levels (input_ptr)
103       local tracingstacklevels = tex.count.tracingstacklevels
104       if tex.tracingmacros > 0 or input_ptr < tracingstacklevels then
105         if tracingstacklevels > 0 then
106           if input_ptr < tracingstacklevels then
107             return "\string\n\string~" .. string.rep(".",input_ptr)
108           else
109             return "\string~\string~"
110           end
111         else
112           return "\string\n"
113         end
114       else
115         return ""
116       end
117     end
118 <latexrelease>   if luatexbase.callbacktypes['input_level_string'] then
119 <latexrelease>     luatexbase.add_to_callback('input_level_string',
120 <latexrelease>       trace_stack_levels,'tracingstacklevels')
121 <latexrelease>   end

```



```

122     }%
123 <*2ekernel>
124     }}%
125 </2ekernel>
126 \fi
127 <latexrelease>\EndIncludeInRelease
128 <latexrelease>

```

Then for the full rollback, just do nothing, since the function was already taken out of the rollback above.

```

129 <latexrelease>\IncludeInRelease{0000/00/00}{trace_stack_levels}%
130 <latexrelease>                {Lua trace_stack_levels function}%
131 <latexrelease>% Nothing here
132 <latexrelease>\EndIncludeInRelease
133 </2ekernel | latexrelease>
134 <*2ekernel>

```

(End definition for trace_stack_levels.)

The default values of the picture and \fbox parameters:

```

135 \unitlength = 1pt
136 \fboxsep = 3pt
137 \fboxrule = .4pt

```

The saved value of T_EX's \maxdepth:

```

138 \@maxdepth      = \maxdepth

```

\vsize initialized because a \clearpage with \vsize < \topskip causes trouble. \@colroom and \@colht also initialized because \vsize may be set to them if a \clearpage is done before the \begin{document}

```

139 \vsize = 1000pt
140 \@colroom = \vsize
141 \@colht = \vsize

```

Initialise \textheight \textwidth and page style, to avoid internal errors if they are not set by the class.

```

142 \textheight=.5\maxdimen
143 \textwidth=\textheight
144 \ps@empty

```

1.3 Lccodes for hyphenation

For 7- and 8-bit engines the assumption of T1 encodings is the basis for the hyphenation patterns. That's not the case for the Unicode engines, where the assumption is engine-native working. The common loader system provides access to data from the Unicode Consortium covering not only \lccode but also other related data. The \lccode part of that at least needs to be loaded before hyphenation is tackled: XeT_EX follows the standard T_EX route of building patterns into the format. LuaT_EX doesn't require this data be loaded *here* but it does need to be loaded somewhere. Rather than test for the Unicode engines by name, the approach here is to look for the extended math mode handling both provide: any other engine developed in this area will presumably also provide \Umathcode.

```

145 \ifnum 0%
146   \ifx\Umathcode\@undefined\else 1\fi
147   \ifx\XeTeXmathcode\@undefined\else 1\fi

```

```

148 >\z@
149 \message{ Unicode character data,}
150 \input{load-unicode-data}
151 (/2ekernel)
152 <latexrelease>\IncludeInRelease{2016/02/01}%
153 <latexrelease> {\XeTeXintercharclasses}{XeTeX character classes}%
154 <latexrelease> \ifx\XeTeXinterchartoks\undefined
155 <latexrelease> \else
156 <latexrelease> \begingroup
157 <latexrelease> \chardef\XeTeXcharclassID = 0 %
158 <latexrelease> \chardef\XeTeXcharclassOP = 0 %
159 <latexrelease> \chardef\XeTeXcharclassCL = 0 %
160 <latexrelease> \chardef\XeTeXcharclassEX = 0 %
161 <latexrelease> \chardef\XeTeXcharclassIS = 0 %
162 <latexrelease> \chardef\XeTeXcharclassNS = 0 %
163 <latexrelease> \chardef\XeTeXcharclassCM = 0 %
164 <latexrelease> \input{load-unicode-xetex-classes}
165 <latexrelease> \endgroup
166 <latexrelease> \global\let\xtxHanGlue\undefined
167 <latexrelease> \global\let\xtxHanSpace\undefined
168 <latexrelease> \global\XeTeXinterchartoks 0 1 = {}
169 <latexrelease> \global\XeTeXinterchartoks 0 2 = {}
170 <latexrelease> \global\XeTeXinterchartoks 0 3 = {}
171 <latexrelease> \global\XeTeXinterchartoks 1 0 = {}
172 <latexrelease> \global\XeTeXinterchartoks 2 0 = {}
173 <latexrelease> \global\XeTeXinterchartoks 3 0 = {}
174 <latexrelease> \global\XeTeXinterchartoks 1 1 = {}
175 <latexrelease> \global\XeTeXinterchartoks 1 2 = {}
176 <latexrelease> \global\XeTeXinterchartoks 1 3 = {}
177 <latexrelease> \global\XeTeXinterchartoks 2 1 = {}
178 <latexrelease> \global\XeTeXinterchartoks 2 2 = {}
179 <latexrelease> \global\XeTeXinterchartoks 2 3 = {}
180 <latexrelease> \global\XeTeXinterchartoks 3 1 = {}
181 <latexrelease> \global\XeTeXinterchartoks 3 2 = {}
182 <latexrelease> \global\XeTeXinterchartoks 3 3 = {}
183 <latexrelease> \fi
184 <latexrelease>\EndIncludeInRelease
185 <latexrelease>\IncludeInRelease{0000/00/00}%
186 <latexrelease> {\XeTeXintercharclasses}{XeTeX character classes}%
187 <latexrelease> \ifx\XeTeXinterchartoks\undefined
188 <latexrelease> \else
189 <latexrelease> \input{load-unicode-xetex-classes}
190 <latexrelease> \gdef\xtxHanGlue{\hskip0pt plus 0.1em\relax}
191 <latexrelease> \gdef\xtxHanSpace{\hskip0.2em plus 0.2em minus 0.1em\relax}
192 <latexrelease> \global\XeTeXinterchartoks 0 1 = {\xtxHanSpace}
193 <latexrelease> \global\XeTeXinterchartoks 0 2 = {\xtxHanSpace}
194 <latexrelease> \global\XeTeXinterchartoks 0 3 = {\nobreak\xtxHanSpace}
195 <latexrelease> \global\XeTeXinterchartoks 1 0 = {\xtxHanSpace}
196 <latexrelease> \global\XeTeXinterchartoks 2 0 = {\nobreak\xtxHanSpace}
197 <latexrelease> \global\XeTeXinterchartoks 3 0 = {\xtxHanSpace}
198 <latexrelease> \global\XeTeXinterchartoks 1 1 = {\xtxHanGlue}
199 <latexrelease> \global\XeTeXinterchartoks 1 2 = {\xtxHanGlue}
200 <latexrelease> \global\XeTeXinterchartoks 1 3 = {\nobreak\xtxHanGlue}
201 <latexrelease> \global\XeTeXinterchartoks 2 1 = {\nobreak\xtxHanGlue}

```

```

202 <latexrelease> \global\XeTeXinterchartoks 2 2 = {\nobreak\xtxHanGlue}
203 <latexrelease> \global\XeTeXinterchartoks 2 3 = {\xtxHanGlue}
204 <latexrelease> \global\XeTeXinterchartoks 3 1 = {\xtxHanGlue}
205 <latexrelease> \global\XeTeXinterchartoks 3 2 = {\xtxHanGlue}
206 <latexrelease> \global\XeTeXinterchartoks 3 3 = {\nobreak\xtxHanGlue}
207 <latexrelease> \fi
208 <latexrelease>\EndIncludeInRelease
209 <*2kernel>

```

There is one over-ride that makes sense here (see below for the same for 8-bit engines): setting the lccode for - to itself.

```
210 \lccode'\- ='\- % default hyphen char
```

The alternative is that a “traditional” engine is in use.

```
211 \else
```

We set things up so that hyphenation files can assume that the default (T1) lccodes are in use (at present this also sets up the uccodes). We temporarily define `\reserved@a` to apply `\reserved@c` to all the numbers in the range of its arguments.

```

212 \def\reserved@a#1#2{%
213   \@tempcnta#1\relax
214   \@tempcntb#2\relax
215   \reserved@b
216 }
217 \def\reserved@b{%
218   \ifnum\@tempcnta>\@tempcntb\else
219     \reserved@c\@tempcnta
220     \advance\@tempcnta\@ne
221     \expandafter\reserved@b
222   \fi
223 }

```

Depending on the T_EX version, we might not be allowed to do this for non-ASCII characters.

```

224 \def\reserved@c#1{%
225   \count@=#1\advance\count@ by -"20
226   \uccode#1=\count@
227   \lccode#1=#1
228 }
229 \reserved@a{'\a}{'\z}
230 \reserved@a{"A0}{"BC}
231 \reserved@a{"E0}{"FF}

```

The upper case characters need their `\uccode` and `\lccode` values set, and their `\sfcode` set to 999.

```

232 \def\reserved@c#1{%
233   \count@=#1\advance\count@ by "20
234   \uccode#1=#1
235   \lccode#1=\count@
236   \sfcode#1=999
237 }
238 \reserved@a{'\A}{'\Z}
239 \reserved@a{"80}{"9C}
240 \reserved@a{"C0}{"DF}

```

Well, it would be nice if that were correct, but unfortunately, the Cork encoding contains some odd slots whose ucode or lcode isn't quite what you'd expect.

```

241 \uccode'\^^Y='^I      % dotless i
242 \lccode'\^^Y='^Y      % dotless i
243 \uccode'\^^Z='^J      % dotless j, ae in OT1
244 \lccode'\^^Z='^Z      % dotless j, ae in OT1
245 \lccode'\^^9d='^i      % dotted I
246 \uccode'\^^9d='^9d    % dotted I
247 \lccode'\^^9e='^9e    % d-bar
248 \uccode'\^^9e='^d0    % d-bar

```

Finally here is one that helps hyphenation in the OT1 encoding.

```

249 \lccode'\^^[='^[[      % oe in OT1

```

And we also set the \lccode of \- and \textcompwordmark so that they do not prevent hyphenation in the remainder of the word (as suggested by Lars Helström).

```

250 \lccode'\- ='^-      % default hyphen char
251 \lccode 127=127      % alternate hyphen char
252 \lccode 23 =23       % textcompwordmark in T1

```

End of the conditional to select either Unicode or T1 encoding defaults.

```

253 \fi

```

At this stage, we can install any last-minute expl3 set-up.

```

254 \@expl@finalise@setup@@
255 \def\@expl@finalise@setup@@{}

```

This is as good a place as any to active a few XeTeX-specific settings

```

256 \ifx\XeTeXuseglyphmetrics\@undefined
257 \else
258   \XeTeXuseglyphmetrics=1 %
259   \XeTeXdashbreakstate=1 %
260 \fi

```

1.4 Hyphenation

The following code will be compiled into the format file. It checks for the existence of `hyphen.cfg` in inputs that file if found. Otherwise it inputs `hyphen.ltx`. Note that these are loaded in *before* the \catcodes are set, so local hyphenation files can use 8-bit input.

We try to load the customized hyphenation description file.

```

261 \InputIfFileExists{hyphen.cfg}
262   {\typeout{=====^^J%
263             Local configuration file hyphen.cfg used^^J%
264             =====}%
265   \def\@addtofilelist##1{\xdef\@filelist{\@filelist,##1}}%
266   }
267   {\input{hyphen.ltx}}
268 \let\@addtofilelist\@gobble

```

\l@nohyphenation

```

269 \ifx\l@nohyphenation \@undefined
270 \newlanguage\l@nohyphenation
271 \fi

```

(End definition for \l@nohyphenation.)

`\document@default@language` Default document language. -1 acts as language 0, but used as a flag in `\document` to see if it has been set in the preamble.

```

272 </2ekernel>
273 <*2ekernel | latexrelease>
274 <latexrelease>\IncludeInRelease{2017/04/15}%
275 <latexrelease>      {\document@default@language}{Save language for hyphenation}%
276 \let\document@default@language\m@ne
277 </2ekernel | latexrelease>
278 <latexrelease>\EndIncludeInRelease
279 <latexrelease>\IncludeInRelease{0000/00/00}%
280 <latexrelease>      {\document@default@language}{Save language for hyphenation}%
281 %
282 <latexrelease>\let\document@default@language\@undefined
283 <latexrelease>\EndIncludeInRelease
284 <*2ekernel>

```

(End definition for \document@default@language.)

1.5 Font loading

Fonts loaded during the formatting process might already have changed the `\font@submax` from 0pt to something higher. If so, we put out a bold warning.

```

285 \ifdim \font@submax >\z@
286   \@font@warning{Size substitutions with differences\MessageBreak
287                 up to \font@submax\space have occurred.\MessageBreak
288                 \MessageBreak
289                 Please check the transcript file
290                 carefully\MessageBreak
291                 and redo the format generation if necessary!
292                 \@gobbletwo}%
293   \errhelp{Only stopped, to give you time to
294           read the above message.}
295   \errmessage{}

```

We reset the macro. Otherwise every user will get a warning on every job.

```

296 \def\font@submax{0pt}
297 \fi

```

For pdfTeX preload and enable automatic glyph to Unicode mapping for more reliable copy and paste support.

```

298 </2ekernel>
299 <*2ekernel | latexrelease>
300 <latexrelease>\IncludeInRelease{2021/06/01}%
301 <latexrelease>      {\pdfgentounicode}{Preload glyphtounicode}%
302 \ifx \pdfgentounicode \@undefined \else
303 <*2ekernel>
304   \ifnum 0=0%
305     \ifdefined\pdftexversion
306 % \pdftexversion<140 does not have \pdfgentounicode, so we only check higher values
307     \ifnum \pdftexversion=140 \ifnum\pdftexrevision<22 1\fi\fi
308     \fi
309     \relax
310 </2ekernel>
311   \input glyphtounicode

```

```

312 <*2ekernel>
313   \else
314     \begingroup
315       \everyeof{\noexpand}\endlinechar-1
316       \edef\x{\endgroup
317         \everyjob{\the\everyjob\@@input glyphtounicode }%
318       }\x
319   \fi
320 </2ekernel>
321 \pdfgentounicode=1
322 \fi
323 </2ekernel | latexrelease>
324 <latexrelease>\EndIncludeInRelease

```

When rolling back we can't unload the glyphtounicode mappings, but we can reset `\pdfgentounicode` to ensure that they aren't used.

```

325 <latexrelease>\IncludeInRelease{0000/00/00}%
326 <latexrelease>           {\pdfgentounicode}{Preload glyphtounicode}%
327 <latexrelease>\ifx \pdfgentounicode \@undefined \else
328 <latexrelease> \pdfgentounicode=0
329 <latexrelease>\fi
330 <latexrelease>\EndIncludeInRelease
331 <*2ekernel>

```

1.6 Input encoding

Starting with the 2018 L^AT_EX release default the inputencoding to UTF-8. Unless the format is being used with luatex, xetex, enc_{te}x or ml_{te}x.

This is done in a way largely compatible with older releases: `utf8.def` is input just as if

```
\usepackage[utf8]{inputenc}
```

had been used, however rather than input the whole package a minimal core part just enough to support loading the UTF-8 encoding files is defined here.

If a document re-specifies UTF-8 this is silently ignored.

```

332 </2ekernel>
333 <*2ekernel | latexrelease>

```

Check that a classic 8-bit tex engine is being used (LaTeX or PDFLaTeX).

```

334 <latexrelease>\IncludeInRelease{2018/04/01}%
335 <latexrelease>           {\UTFviii@invalid}{UTF-8 default}%

```

Skip this section in Unicode TeX, or if MLTeX and EncTeX are enabled.

```

336 \ifnum0%
337   \ifx\Umathcode\@undefined\else 1\fi
338   \ifx\mubyte\@undefined\else 1\fi
339   \ifx\charsubdef\@undefined\else 1\fi
340   =\z@
341 \def\saved@space@catcode{10}
342 \let\@inpenc@test\relax
343 \def\IeC{%
344   \ifx\protect\@typeset@protect
345     \expandafter\@firstofone
346   \else

```

```

347     \noexpand\IeC
348     \fi
349 }

    Make characters active for UTF-8 input formats
350 \@tempcnta=1
351 \loop
352   \catcode\@tempcnta=13 %
353   \advance\@tempcnta\@ne %
354   \ifnum\@tempcnta<32 %
355   \repeat %
356   \catcode0=15 % null
357   \catcode9=10 % tab
358   \catcode10=12 % ctrl J
359   \catcode12=13 % ctrl L
360   \catcode13=5 % newline
361   \@tempcnta=128
362   \loop
363     \catcode\@tempcnta=13
364     \advance\@tempcnta\@ne
365     \ifnum\@tempcnta<256
366     \repeat

```

\UseRawInputEncoding Reset 8 bit characters to catcode 12 so the input encoding matches the “Raw” font encoding. Useful for special behaviours, or for compatibility with older L^AT_EX formats.

```

367 \def\UseRawInputEncoding{%
368   \let\inputencodingname\@undefined % revert
369   \let\DeclareFontEncoding\DeclareFontEncoding@saved % revert
370   \let\DeclareUnicodeCharacter\@undefined % revert
371   \@tempcnta=1
372   \loop
373     \catcode\@tempcnta=15 %
374     \advance\@tempcnta\@ne %
375     \ifnum\@tempcnta<32 %
376     \repeat %
377     \catcode0=15 % null
378     \catcode9=10 % tab
379     \catcode10=12 % ctrl J
380     \catcode12=13 % ctrl L
381     \catcode13=5 % newline
382     \@tempcnta=128
383     \loop
384       \catcode\@tempcnta=12
385       \advance\@tempcnta\@ne
386       \ifnum\@tempcnta<256
387       \repeat
388   }

```

(End definition for \UseRawInputEncoding.)

\DeclareFontEncoding@saved Saved version of \DeclareFontEncoding@ before utf8.def modifies it for use in \UseRawInputEncoding above.

```

389 \let\DeclareFontEncoding@saved\DeclareFontEncoding@

```

(End definition for \DeclareFontEncoding@saved.)

```
390 \edef\inputencodingname{utf8}%
391 \input{utf8.def}
392 \let\UTFviii@undefined@err@@\UTFviii@undefined@err
393 \let\UTFviii@invalid@err@@\UTFviii@invalid@err
394 \let\UTFviii@two@octets@@\UTFviii@two@octets
395 \let\UTFviii@three@octets@@\UTFviii@three@octets
396 \let\UTFviii@four@octets@@\UTFviii@four@octets
397 <2kernel>\def\UTFviii@undefined@err#1{\@gobble#1}%
398 <2kernel>\let\UTFviii@invalid@err\string
399 <2kernel>\let\UTFviii@two@octets\string
400 <2kernel>\let\UTFviii@three@octets\string
401 <2kernel>\let\UTFviii@four@octets\string
402 <2kernel>\everyjob\expandafter{\the\everyjob
403 <2kernel>\let\UTFviii@undefined@err\UTFviii@undefined@err@@
404 <2kernel>\let\UTFviii@invalid@err\UTFviii@invalid@err@@
405 <2kernel>\let\UTFviii@two@octets\UTFviii@two@octets@@
406 <2kernel>\let\UTFviii@three@octets\UTFviii@three@octets@@
407 <2kernel>\let\UTFviii@four@octets\UTFviii@four@octets@@
408 <2kernel>}}
409 \let\@inpenc@test\@undefined
410 \let\saved@space@catcode\@undefined
```

For formats not set up for UTF-8 default, set the C0 controls to catcode 15.

```
411 \else
412 \@tempcnta=0
413 \loop
414 \catcode\@tempcnta=15 %
415 \advance\@tempcnta\@ne %
416 \ifnum\@tempcnta<32 %
417 \repeat %
418 \catcode0=15 % null
419 \catcode9=10 % tab
420 \catcode10=12 % ctrl J
421 \catcode12=13 % ctrl L
422 \catcode13=5 % newline
423 \let\UseRawInputEncoding\relax
```

This ends the skipped code in Unicode engines:

```
424 \fi
425 </2kernel | latexrelease>
426 <latexrelease>\EndIncludeInRelease
427 <latexrelease>\IncludeInRelease{0000/00/00}%
428 <latexrelease> \UTFviii@invalid}{UTF-8 default}}%
```

The first block of commands got only introduced in 2019 but we revert all of Unicode support in one go not jump to the intermediate version.

```
429 <latexrelease> \let\UTFviii@two@octets@combine\@undefined
430 <latexrelease> \let\UTFviii@three@octets@combine\@undefined
431 <latexrelease> \let\UTFviii@four@octets@combine\@undefined
432 <latexrelease> \let\UTFviii@two@octets@string\@undefined
433 <latexrelease> \let\UTFviii@three@octets@string\@undefined
434 <latexrelease> \let\UTFviii@four@octets@string\@undefined
435 <latexrelease> \let\UTFviii@two@octets@noexpand\@undefined
```



```

436 <latexrelease> \let\UTFviii@three@octets@noexpand\@undefined
437 <latexrelease> \let\UTFviii@four@octets@noexpand\@undefined

438 <latexrelease>\@tempcnta=0
439 <latexrelease>\loop
440 <latexrelease> \catcode\@tempcnta=15
441 <latexrelease> \advance\@tempcnta\@ne
442 <latexrelease>\ifnum\@tempcnta<32
443 <latexrelease>\repeat %
444 <latexrelease>\catcode9=10 % tab
445 <latexrelease>\catcode10=12 % ctrl J
446 <latexrelease>\catcode12=13 % ctrl L
447 <latexrelease>\catcode13=5 % newline
448 <latexrelease>\@tempcnta=128
449 <latexrelease>\loop
450 <latexrelease>\catcode\@tempcnta=12
451 <latexrelease>\advance\@tempcnta\@ne
452 <latexrelease>\ifnum\@tempcnta<256
453 <latexrelease>\repeat
454 <latexrelease>\let\IeC\@undefined
455 <latexrelease>\def\DeclareFontEncoding@#1#2#3{%
456 <latexrelease> \expandafter
457 <latexrelease> \ifx\csname T@#1\endcsname\relax
458 <latexrelease> \def\cdp@elt{\noexpand\cdp@elt}%
459 <latexrelease> \xdef\cdp@list{\cdp@list\cdp@elt{#1}%
460 <latexrelease> {\default@family}\default@series}%
461 <latexrelease> {\default@shape}}%
462 <latexrelease> \expandafter\let\csname#1-cmd\endcsname\@changed@cmd
463 <latexrelease> \else
464 <latexrelease> \font@info{Redeclaring font encoding #1}%
465 <latexrelease> \fi
466 <latexrelease> \global\@namedef{T@#1}{#2}%
467 <latexrelease> \global\@namedef{M@#1}{\default@M#3}%
468 <latexrelease> \xdef\LastDeclaredEncoding{#1}%
469 <latexrelease> }
470 <latexrelease> \let\UseRawInputEncoding\@undefined
471 <latexrelease> \let\DeclareFontEncoding@saved\@undefined
472 <latexrelease> \let\inputencodingname\@undefined
473 <latexrelease>\EndIncludeInRelease

474 <*2kernel>
475 % \begin{macrocode}
476 %
477 % We temporarily define |\reserved@a| to apply |\reserved@c| to all the
478 % numbers in the range of its arguments.
479 % \begin{macrocode}
480 \def\reserved@a#1#2{%
481 \@tempcnta#1\relax
482 \@tempcntb#2\relax
483 \reserved@b
484 }
485 \def\reserved@b{%
486 \ifnum\@tempcnta>\@tempcntb\else
487 \reserved@c\@tempcnta
488 \advance\@tempcnta\@ne

```

```

489     \expandafter\reserved@b
490     \fi
491 }

```

Set the special catcodes (although some of these are useless, since an error will have occurred if the catcodes have changed). Note that \sim J has catcode ‘other’ for use in warning messages.

```

492 \catcode'\ =10
493 \catcode'\#=6
494 \catcode'\$=3
495 \catcode'\%=14
496 \catcode'\&=4
497 \catcode'\|=0
498 \catcode'\^=7
499 \catcode'\_ =8
500 \catcode'\{=1
501 \catcode'\}=2
502 \catcode'\-=13
503 \catcode'\@=11
504 \catcode'\^^I=10
505 \catcode'\^^J=12
506 \catcode'\^^L=13
507 \catcode'\^^M=5

```

Set the ‘other’ catcodes.

```

508 \def\reserved@c#1{\catcode#1=12\relax}
509 \reserved@c{'\!}
510 \reserved@c{'\"}
511 \reserved@a{'\' }{'\?}
512 \reserved@c{'\[]
513 \reserved@c{'\}
514 \reserved@c{'\' }
515 \reserved@c{'\|}

```

Set the ‘letter’ catcodes.

```

516 \def\reserved@c#1{\catcode#1=11\relax}
517 \reserved@a{'\A }{'\Z}
518 \reserved@a{'\a }{'\z}

```

All the characters in the range 0–31 and 127–255 are illegal, *except* tab (\sim I), nl (\sim J), ff (\sim L) and cr (\sim M).

1.7 Lccodes and uccodes

We now again set up the default (T1) uc/lccodes. The lower case characters need their `\uccode` and `\lccode` values set. Some of this is a repeat of the set-up before loading hyphenation files. Depending on the $\mathrm{T}_{\mathrm{E}}\mathrm{X}$ version, we might not be allowed to do this for non-ASCII characters. For the Unicode engines (Xe $\mathrm{T}_{\mathrm{E}}\mathrm{X}$ and Lua $\mathrm{T}_{\mathrm{E}}\mathrm{X}$) there is no need to do any of this: they use hyphenation data which does not alter any of the set up and so this entire block is skipped.

```

519 \ifnum 0%
520   \ifx\Umathcode\@undefined\else 1\fi
521   \ifx\XeTeXmathcode\@undefined\else 1\fi
522   >\z@
523 \else

```

```

524 \def\reserved@c#1{%
525     \count@=#1\advance\count@ by -"20
526     \uccode#1=\count@
527     \lccode#1=#1
528 }
529 \reserved@a{'\a}{'\z}
530 \reserved@a{"A0}{\BC}
531 \reserved@a{"E0}{\FF}

```

The upper case characters need their \uccode and \lccode values set, and their \sfcode set to 999.

```

532 \def\reserved@c#1{%
533     \count@=#1\advance\count@ by "20
534     \uccode#1=#1
535     \lccode#1=\count@
536     \sfcode#1=999
537 }
538 \reserved@a{'\A}{'\Z}
539 \reserved@a{"80}{\9C}
540 \reserved@a{"C0}{\DF}

```

Well, it would be nice if that were correct, but unfortunately, the Cork encoding contains some odd slots whose uccode or lccode isn't quite what you'd expect.

```

541 \uccode'\^^Y='I % dotless i
542 \lccode'\^^Y='^^Y % dotless i
543 \uccode'\^^Z='J % dotless j, ae in OT1
544 \lccode'\^^Z='^^Z % dotless j, ae in OT1
545 \lccode'\^^9d='i % dotted I
546 \uccode'\^^9d='^^9d % dotted I
547 \lccode'\^^9e='^^9e % d-bar
548 \uccode'\^^9e='^^d0 % d-bar

```

Finally here is one that helps hyphenation in the OT1 encoding.

```

549 \lccode'\^^[='^^[ % oe in OT1
550 \fi % End of reset block for 8-bit engines

```

\MakeUppercase
\MakeUppercase
\@uclclist

And whilst we're doing things with uc/lc tables, here are two commands to upper- and lower-case a string.

Note that this implementation is subject to change! At the moment we're not providing any way to extend the list of uc/lc commands, since finding a good interface is difficult. These commands have some nasty features, such as uppercasing mathematics, environment names, labels, etc. A much better long-term solution is to use all-caps fonts, but these aren't generally available.

```

551 \DeclareRobustCommand{\MakeUppercase}[1]{%
552     \def\i{I}\def\j{J}%
553     \def\reserved@a##1##2{\let##1##2\reserved@a}%
554     \expandafter\reserved@a\@uclclist\reserved@b{\reserved@b\@gobble}%

```

Tell UTF-8 processing to process chars even though we are in an \protected@edef.

```

555     \let\UTF@two@octets@noexpand\@empty
556     \let\UTF@three@octets@noexpand\@empty
557     \let\UTF@four@octets@noexpand\@empty
558     \protected@edef\reserved@a{\uppercase{#1}}%
559     \reserved@a
560 }

```

```

561 \DeclareRobustCommand{\MakeLowercase}[1]{%
562     \def\reserved@a##1##2{\let##2##1\reserved@a}%
563     \expandafter\reserved@a\@uclclist\reserved@b{\reserved@b@gobble}%
564     \let\UTF@two@octets@noexpand\@empty
565     \let\UTF@three@octets@noexpand\@empty
566     \let\UTF@four@octets@noexpand\@empty
567     \protected@edef\reserved@a{\lowercase{#1}}%
568     \reserved@a
569 }}

570 \def\@uclclist{\oe\OE\o\O\ae\AE
571     \dh\DH\dj\DJ\l\L\ng\NG\ss\SS\th\TH}

```

The above code works, but has the nasty side-effect that if you say something like:

```

\markboth{\MakeUppercase\contentsname}
         {\MakeUppercase\contentsname}

```

then the uppercasing is only done to the first letter of the contents name, since the mark expands out to:

```

\mark{\protect\MakeUppercase Table of Contents}
      {\protect\MakeUppercase Table of Contents}

```

In order to get round this, we redefine `\MakeUppercase` and `\MakeLowercase` to grab their argument and brace it. This is a very low-level hack, and is *not* recommended practice! This is an instance of a general problem that makes it unsafe to grab arguments unbraced, and probably needs a more general solution. For the moment though, this hack will do:

```

572 \protected@edef\MakeUppercase#1{\MakeUppercase{#1}}
573 \protected@edef\MakeLowercase#1{\MakeLowercase{#1}}

```

(End definition for `\MakeUppercase`, `\MakeLowercase`, and `\@uclclist`.)

1.8 Applying Patch files

Between major releases, small patches will be distributed in files `ltpatch.ltx` which must be added at this point.

Patch file code removed.

```

574 %\IfFileExists{ltpatch.ltx}
575 % {\typeout{=====^^J%
576 %         Applying patch file ltpatch.ltx^^J%
577 %         =====}}
578 % \def\fmtversion@topatch{unknown}
579 % \input{ltpatch.ltx}
580 % \ifx\fmtversion\fmtversion@topatch
581 % \ifx\patch@level\@undefined
582 % \typeout{^^J^^J^^J%
583 %         !!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!^^J%
584 %         !! Patch file 'ltpatch.ltx' not suitable for this^^J%
585 %         !! version of LaTeX.^^J^^J%
586 %         !! Please check if initex found an old patch file:^^J%
587 %         !! --- if so, rename it or delete it, and redo the^^J%
588 %         !! initex run.^^J%
589 %         !!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!^^J}%

```

The code below adds the ‘patch level’ string to the first `\typeout` in the startup banner.

1.9 Freeing Memory

```

615 \let\reserved@a=@filelist
616 \let\reserved@b=@undefined
617 \let\reserved@c=@undefined
618 \let\reserved@d=@undefined
619 \let\reserved@e=@undefined
620 \let\reserved@f=@undefined

```

\toks

(End definition for \toks.)

626 \errhelp{}

File X: ltfinal.dtx Date: 2021/04/18 Version v2.2o

1.10 Initialise file list

`\@providesfile` Initialise for use in the document. During initex a modified version has been used which leaves debugging information for `latexbug.tex`.

```
627 \def\@providesfile#1[#2]{%  
628   \wlog{File: #1 #2}%  
629   \expandafter\xdef\csname ver@#1\endcsname{#2}%  
630   \endgroup}
```

(End definition for \@providesfile.)

`\@filelist` Reset `\@filelist` so files input while making the format are not listed. The list built up so far may take up a lot of memory and so it is moved to `\reserved@a` where it will be overwritten as soon as almost any L^AT_EX command is issued in a class file. However the `latexbug.tex` program will be able to access this information and insert it into a bug report.

```
631 \let\@filelist\@gobble  
632 \def\@addtofilelist#1{\xdef\@filelist{\@filelist,#1}}%
```

(End definition for \@filelist and \@addtofilelist.)

1.11 Do some temporary work for pre-release

This is a good place to load code that hasn't yet been integrated into the other files ...

1.12 Some last minute initializations ...

Load the first aid set of definitions for external packages that await updates.

```
633 \input{latex2e-first-aid-for-external-files.ltx}
```

1.13 Dumping the format

Finally we make @ into a letter, ensure the format will be in the 'normal' error mode, and dump everything into the format file.

```
634 \makeatother  
635 \errorstopmode  
636 \dump  
637 </2ekernel>
```

Change History

1985-11-04 ltmath.dtx LaTeX2.09		<code>\mathversion</code> : Test if version defined added.	387
General: produce warning message if line extends into margin. Doesn't warn about formula overprinting equation number.	607		
1989-04-10 ltfsbas.dtx v1.0a		1989-04-29 ltfsbas.dtx v1.0i	
General: Starting with version numbers! <code>\ifmmode</code> added in <code>\math@group</code>	376	General: Removed the <code>\halign</code> <code>\noalign</code> correction (wasn't bugfree)	376
1989-04-10 ltfsbas.dtx v1.0b		1989-04-29 ltfsini.dtx v1.0f	
General: <code>\preload@sizes</code> added.	376	General: Corrections to L ^A T _E X tabular env. added.	481
<code>\wrong@fontshape</code> changed to define substitution font/shape macro.	376	1989-05-01 ltfsbas.dtx v1.0j	
1989-04-10 ltfsini.dtx v1.0a		General: Default for <code>\baselinestretch</code> added.	376
General: Starting with version numbers <code>\newif</code> for <code>\@tempwa</code> added since this switch is unknown at the time when this file is read in. (latex.tex is loaded later.) <code>\math@famname</code> changed to <code>\math@version</code>	481	1989-05-22 ltfsbas.dtx v1.0k	
1989-04-14 ltfsbas.dtx v1.0c		General: Lines longer than 72 characters folded.	376
General: More documentation added.	376	1989-05-22 ltfsini.dtx v1.0g	
1989-04-15 ltfsini.dtx v1.0b		General: Lines shortened to 72 characters	481
General: <code>\mathfontset</code> renamed to <code>\mathversion</code>	481	1989-09-14 ltfsbas.dtx v1.0m	
1989-04-19 ltfsbas.dtx v1.0d		General: Global replacement: <code>\group</code> to <code>\mathgroup</code>	376
General: Even more doc.	376	<code>\mathversion</code> : Corrected typo: <code>\endscname</code> to <code>\endcsname</code>	387
1989-04-21 ltfsbas.dtx v1.0e		1989-11-07 ltfsini.dtx v1.0i	
General: Documentation is fun! Parameters of <code>\define@mathalphabet</code> changed.	376	General: All family, series, and shape names abbreviated.	481
1989-04-21 ltfsini.dtx v1.0c		1989-11-08 ltfsbas.dtx v1.0o	
General: Changed to conform to fam.tex.	481	General: First parameter of <code>\define@mathalphabet</code> and <code>\define@mathgroup</code> changed from string to control sequence.	376
1989-04-23 ltfsbas.dtx v1.0f		1989-11-14 ltfsbas.dtx v1.0p	
General: % in <code>\getanddefinefonts</code> added.	376	<code>\math@version</code> : Math version prefix 'mv@' added.	387
1989-04-26 ltfsini.dtx v1.0d		1989-11-19 ltfsbas.dtx v1.0q	
General: <code>\xpt</code> added.	481	<code>\define@newfont</code> : Group added.	389
1989-04-27 ltfsbas.dtx v1.0g		<code>\wrong@fontshape</code> : Instead of calling <code>\family\default@family</code> , etc. we directly set <code>\f@family</code> , etc.	394
General: Documentation revised.	376	1989-11-22 ltfsbas.dtx v1.0r	
1989-04-27 ltfsini.dtx v1.0e		<code>\math@version</code> : <code>\def</code> → <code>\edef</code> for <code>\math@version</code>	387
General: Definitions of L ^A T _E X symbols corrected.	481	1989-11-25 ltfsbas.dtx v1.0s	
1989-04-29 ltfsbas.dtx v1.0h		General: All <code>\edef\font@name</code> changed to <code>\xdef\font@name</code> . Necessary after introduction of <code>\begingroup/\endgroup</code> in v1.0q.	376
General: Documented problem with <code>\halign</code> , and <code>\noalign</code>	376	extra// → + in <code>\extra@def</code>	376

1989-11-26 ltfssbas.dtx v1.0t	1990-01-25 ltfssini.dtx v1.1e
<code>\select@group</code> : <code>\bgroup</code> / <code>\egroup</code>	<code>\nfss@text</code> : Macro added. 501
changed to	1990-01-27 ltfssbas.dtx v1.2d
<code>\begingroup</code> / <code>\endgroup</code> to avoid	<code>\DeclarePreloadSizes</code> : Font identifier
empty Ord atom on math list. . . 396	set to <code>\relax</code> 382
1989-12-02 ltfssini.dtx v1.1b	1990-01-28 ltfssbas.dtx v1.2e
General: <code>\rmmath</code> renamed to	<code>\mathgroup</code> : <code>\newfam</code> let to
<code>\mathrm</code> 481	<code>\new@mathgroup</code> 376
1989-12-03 ltfssini.dtx v1.1c	1990-01-28 ltfssbas.dtx v1.2f
General: Some internal macros	<code>\define@newfont</code> : Added call to
renamed to make them	<code>\curr@fontshape</code> macro to allow
inaccessible. 481	substitution. 389
1989-12-05 ltfssbas.dtx v1.0u	<code>\wrong@fontshape</code> : Warning message
<code>\addto@hook</code> : <code>\addto@hook</code> added. . 401	slightly changed. 394
1989-12-05 ltfssstrc.dtx v1.0u fam.dtx	1990-01-28 ltfssini.dtx v1.2b
<code>\every@math@size</code> : Hook <code>\every@size</code>	<code>\em</code> : Call to <code>\@nomath</code> added. 498
added. 433	1990-02-08 ltfssini.dtx v1.1g
1989-12-13 ltfssstrc.dtx v1.0f	General: Protected the commands
<code>\use@mathgroup</code> : <code>\expandafter</code> added	<code>\family</code> , <code>\series</code> , <code>\shape</code> , <code>\size</code> ,
before final <code>\fi</code> 436	<code>\selectfont</code> , and <code>\mathversion</code> . 481
1989-12-16 ltfssbas.dtx v1.1a	1990-02-16 ltfssbas.dtx v1.2g
<code>\select@group</code> : <code>\relax</code> in front	General: Support for changes of
added. 396	<code>\baselineskip</code> without changing
Now four arguments. 396	the size. 376
Redefinition of alphabet now	<code>\math@version</code> : <code>\@nomath</code> added. . . 387
simpler. 397	1990-02-18 ltfssstrc.dtx v1.0j
Usage of ‘=’ macro added. 397	<code>\selectfont</code> : Redefine unprotected
1989-12-16 ltfssstrc.dtx v1.1a	version <code>\p@selectfont</code> instead of
<code>\selectfont</code> : Changed order of calls. 428	<code>\selectfont</code> 428
<code>\use@mathgroup</code> : Redefinition of	1990-03-14 ltfssstrc.dtx v1.0k
alphabet now simpler. 435	General: Added code for TeX3. 424
Usage of ‘=’ macro added. 435	<code>\extract@font</code> : Added code for
1990-01-18 ltfssstrc.dtx v1.0h	TeX3. 427
General: <code>\tracingfonts</code> meaning	1990-03-30 ltfssbas.dtx v1.2h
changed. 424	<code>\math@egroup</code> : Changed to have one
1990-01-20 ltfssbas.dtx v1.2a	arg. 399
<code>\math@bgroup</code> : Def. placed in this	1990-03-30 ltfssstrc.dtx v1.2h
file. 399	<code>\use@mathgroup</code> : Third argument
<code>\math@egroup</code> : Def. placed in this	removed (see <code>\math@egroup</code>). . . . 435
file. 399	1990-04-01 ltfssbas.dtx v1.2i
<code>\select@group</code> : Def for alph id	General: Code added from
changed. 397	tracefnt.dtx. 376
1990-01-21 ltfssbas.dtx v1.2b	Support for TeX3. 376
<code>\select@group</code> : Code moved to	1990-04-01 ltfssstrc.dtx v1.0l
<code>\use@mathgroup</code> 397	General: Part of code moved to
1990-01-21 ltfssstrc.dtx v1.2b	fam.dtx. 424
<code>\use@mathgroup</code> : Macro added to	<code>\tracingfonts</code> : Check if
allow cleaner interface. 435	<code>\tracingfonts</code> already defined. . 425
1990-01-23 ltfssbas.dtx v1.2c	1990-04-01 ltfssstrc.dtx v1.0o
General: <code>\no@version@warning</code>	<code>\tracingfonts</code> : Check if
renamed to <code>\no@alphabet@error</code> . 376	<code>\tracingfonts</code> defined removed
Macro <code>\no@alphabet@help</code> added 376	again. 425
<code>\no@alphabet@error</code> : Changed to	
error call 376	

1990-04-02 ltfsini.dtx v1.1i	1991-08-14 ltpictur.dtx LaTeX2.09
General: <code>\input</code> of files now handled	General: (RmS) inserted extra braces
by docstrip. 481	around entry for NFSS 675
1990-04-05 ltfsstrc.dtx v1.0m	1991-08-14 ltthm.dtx LaTeX2.09
<code>\selectfont</code> : Call <code>\tracingon</code> only if	<code>\endtheorem</code> : Moved <code>\itshape</code> after
<code>\tracingfonts</code> greater than 3. . . 428	<code>\item</code> to make it work with NFSS 705
1990-05-05 ltfsstrc.dtx v1.0n	1991-08-26 ltfsini.dtx v1.1n
<code>\selectfont</code> : <code>\tracingon</code> with new	<code>\reset@font</code> : Macro introduced . . . 501
syntax. 428	1991-08-26 ltmiscen.dtx LaTeX2.09
1990-06-23 ltfsini.dtx v1.1k	<code>\verbatim</code> : <code>\@par</code> added 592
<code>\nfss@text</code> : Changed to <code>\mbox</code> 501	1991-08-26 ltpictur.dtx LaTeX2.09
1990-06-24 ltfsbas.dtx v1.2j	<code>\endpicture</code> : (RmS & FMi) extra
<code>\DeclarePreloadSizes</code> : Missing	boxing level around <code>\@picbox</code> to
percent added. 382	guard against unboxing in math
1990-06-24 ltfsstrc.dtx v1.0o	mode (proposed by John Hobby) 673
<code>\baselinestretch</code> : Moved to	1991-08-26 ltplain.dtx LaTeX2.09
tracefnt.dtx. 433	<code>\tracingall</code> : Added
<code>\getanddefine@fonts</code> : <code>\Adding</code>	<code>\errorcontextlines=\maxdimen</code> ,
tracing code. 437	suggested by J. Schrod 32
<code>\Macro</code> moved from fam.dtx. . . . 436	1991-09-29 ltboxes.dtx LaTeX2.09
Adding debug code. 436	<code>\mpfootnotetext</code> : (RmS) added
<code>\use@mathgroup</code> : Tracing code added. 436	<code>\reset@font</code> 641
1990-06-30 ltfsbas.dtx v1.2l	1991-09-29 ltfloat.dtx LaTeX2.09
<code>\showhyphens</code> : Macro added. 399	<code>\@footnotetext</code> : (RmS) added
1990-06-30 ltfsstrc.dtx v1.0p	<code>\reset@font</code> 738
<code>\use@mathgroup</code> : Added <code>\relax</code> after	1991-09-29 ltmath.dtx LaTeX2.09
math group number. 436	<code>\eqnnum</code> : RmS: <code>\reset@font</code> added. 607
1990-07-07 ltfsstrc.dtx v1.0q	1991-09-29 ltsect.dtx LaTeX2.09
<code>\getanddefine@fonts</code> : Group number	<code>\dottedtocline</code> : (RmS) added
added to tracing. 437	<code>\reset@font</code> for page number . . 717
<code>\math@egroup</code> : Tracing code added. 436	1991-10-17 ltctrl.dtx LaTeX2.09
<code>\use@mathgroup</code> : Group number	<code>\@tfor</code> : (RmS) <code>\xdef</code> replaced by <code>\def</code>
added to tracing. 436	(See FMi's array.doc) 251
1990-08-27 ltfsstrc.dtx 1.0r	1991-10-25 ltbibl.dtx LaTeX2.09
<code>\type@restoreinfo</code> : Some extra	<code>\citex</code> : added <code>\reset@font</code> ,
tracing info. 432	suggested by Bernd Raichle. . . . 746
1990-08-27 ltfsstrc.dtx v1.0r	1991-11-01 ltfloat.dtx LaTeX2.09
<code>\getanddefine@fonts</code> : Correcting	<code>\footnote</code> : (RmS) Added
missing name after <code>\tracingon</code> . . 436	<code>\let\protect\noexpand</code> in
1991-03-28 ltfsini.dtx v1.1m	<code>\footnote</code> , <code>\footnotemark</code> , and
<code>\copyright</code> : Extra braces added. . . 501	<code>\footnotetext</code> , since <code>\xdef</code> is
1991-03-30 ltfsini.dtx v1.2g	used 738
<code>\newfont</code> : Definition added. 500	1991-11-04 ltlists.dtx LaTeX2.09
<code>\symbol</code> : Definition added. 500	<code>\makelabel</code> : (RmS) added default
1991-07-24 ltmiscen.dtx LaTeX2.09	definition for <code>\makelabel</code> , to
<code>\@verbatim</code> : Added	produce an error message. 626
<code>\penalty\interlinepenalty</code> to	1991-11-04 ltplain.dtx RmS
definition of <code>\par</code> so that	General: Removed <code>\itemitem</code> since
<code>\samepage</code> works 592	never needed/useful with L ^A T _E X. . . 30
1991-08-14 ltmath.dtx LaTeX2.09	1991-11-06 ltbibl.dtx LaTeX2.09
<code>\cases</code> : (RmS) inserted extra braces	<code>\citex</code> : added code to remove a
around entry for NFSS 602	leading blank 746

1991-11-13 ltbib1.dtx LaTeX2.09	avoid conflicts with other channels allocated by <code>\newread</code>	75
<code>\@bibitem</code> : Changed counter <code>enumi</code> to <code>enumiv</code> , as it says in the comment above		746
1991-11-21 ltfssini.dtx v1.1o	<code>\xympar</code> : (RmS) added <code>\global\@ignorefalse</code>	733
<code>\reset@font</code> : Added extra braces for robustness.	<code>\end@float</code> : (RmS) changed <code>\@esphack</code> to <code>\@Esphack</code>	727
Changed to protected version of macro.	1992-03-18 ltlists.dtx 0.0 <code>\trivlist</code> : RmS: added <code>\@nmbrlistfalse</code>	501 622
1991-11-22 ltfloat.dtx LaTeX2.09	1992-03-18 ltmiscen.dtx LaTeX2.09 <code>\begin</code> : Changed <code>\@ignoretrue</code> to <code>\@ignorefalse</code> (as documented)	584
<code>\footnote</code> : (RmS) Added <code>\let\protect\noexpand</code> in <code>\@xfootnote</code> , <code>\@xfootnotemark</code> , and <code>\@xfootnotetext</code>	1992-03-21 ltfssini.dtx v1.2d General: Renamed <code>\text</code> to <code>\nfss@text</code> to make it internal.	738 481
1991-11-22 ltlists.dtx LaTeX2.09	1992-05-12 ltfssbas.dtx v1.3c <code>\extract@alph@from@version</code> : Macro added.	626 398
<code>\@item</code> : (RmS) Changed second call to <code>\makelabel</code> to <code>\unhbox\@tempboxa</code> . Avoids problems with side effects in <code>\makelabel</code> and is more efficient.	<code>\select@group</code> : Added call to <code>\extract@alph@from@version</code>	397
1991-11-27 ltfssbas.dtx v1.3a	1992-07-26 ltfssbas.dtx v1.9a <code>\curr@fontshape</code> :	376
General: All <code>\family</code> , <code>\shape</code> etc. renamed to <code>\fontfamily</code> etc.	<code>\DeclareFontShape</code> : Introduced <code>\DeclareFontShape</code>	389 377
1991-11-27 ltfssini.dtx v1.2a	<code>\define@newfont</code> :	389
General: All <code>\family</code> , <code>\shape</code> etc. renamed to <code>\fontfamily</code> etc.	<code>\math@fonts</code> :	396
1992-01-06 ltfssini.dtx v1.2c	<code>\select@group</code> :	397
General: added <code>slitex</code> code	<code>\split@name</code> : Added splitting into <code>\f@encoding</code>	481 389
1992-01-10 ltbib1.dtx LaTeX2.09	<code>\wrong@fontshape</code> :	394
<code>\@bibitem</code> : Changed <code>\c@enumiv</code> to <code>\value of \@listctr</code>	1992-07-26 ltfssstrc.dtx v2.0b <code>\s@fct@</code> :	746 445
1992-01-10 ltmath.dtx LaTeX2.09	<code>\s@fct@sub</code> : documentation fixes	446
<code>equation</code> : RmS: put <code>\hbox</code> around <code>\@eqnnum</code> to typeset the equation number in text mode (as in the <code>eqnarray</code> env.)	<code>\selectfont</code> :	607 429
1992-01-10 ltthm.dtx LaTeX2.09	<code>\try@simple@size</code> :	439
<code>\@othm</code> : (RmS) Check for existence of theorem environment	<code>\try@size@range</code> :	443
1992-01-14 ltbib1.dtx LaTeX2.09	<code>\use@mathgroup</code> :	436
<code>\@biblabel</code> : removed <code>\hfill</code>	1992-08-14 ltbib1.dtx LaTeX2.09 <code>\@citex</code> : added missing argument braces around <code>\hbox</code> , found by Ed Sznyter	749 746
1992-01-14 ltsect.dtx 0.0	1992-08-14 ltboxes.dtx LaTeX2.09 <code>\endminipage</code> : (RmS) replaced <code>\vskip</code> - <code>\lastskip</code> by <code>\unskip</code> (proposed by FMi)	715 640
<code>\@starttoc</code> : (RmS) added <code>\immediate</code> to <code>\openout</code> as all <code>\write</code> commands are also executed <code>\immediate</code>	1992-08-17 ltbib1.dtx LaTeX2.09 <code>\@citex</code> : simplified code for removing leading blanks in citation key (proposed by Frank Jensen and Kresten Krab Thorup)	746 746
1992-02-26 ltbib1.dtx LaTeX2.09	1992-08-19 ltsect.dtx 0.0 <code>\@xsect</code> : (RmS) corrected bug: stretch and shrink in argument to <code>\hskip</code>	
<code>\@lbibitem</code> : Added <code>\hfill</code> to restore left-alignment of bibliography labels in alpha style		
1992-03-18 ltdefns.dtx LaTeX2.09		
General: (RmS) changed input channel from 0 to <code>\@inputcheck</code> to		

previously not negated	711	<code>\footnote:</code> (RmS) Changed all to 'def'protect'noexpand'protect'noexpand	738
1992-08-19 ltthm.dtx LaTeX2.09		1992-12-03 ltffsini.dtx v?	
<code>\@othm:</code> (RmS) Changed error message to complain about undefined counter	704	<code>\hexnumber@:</code> Make it accept counters.	501
1992-08-20 ltffsini.dtx v1.4b		1993-03-08 preload.dtx v2.0b	
<code>\@setsize:</code> Added <code>\@currsize.</code> . . .	500	General: Added 12pt preloads	527
1992-08-24 ltdefs.dtx LaTeX2.09		1993-03-18 ltffsbas.dtx v2.0c	
<code>\@ifnextchar:</code> (Rms) <code>\@ifnextchar</code> didn't work if its first argument was an equal sign.	97	General: Changed all <code>\@tempdima</code> in <code>\@tempdimb</code> to avoid killing <code>\numberline</code>	376
1992-08-24 ltmiscen.dtx LaTeX2.09		1993-03-18 ltfsstrc.dtx v2.1b	
<code>\begin:</code> Added code to <code>\begin</code> to remember line number. Used by <code>\@badend</code> to display position of non-matching <code>\begin.</code>	584	General: Changed all <code>\@tempdima</code> in <code>\@tempdimb</code> to avoid killing <code>\numberline</code>	424
<code>\verb:</code> Changed <code>\verb</code> and <code>\@sverb</code> to work correctly in math mode . . .	597	Changed all <code>\@tempdimb</code> in <code>\@tempdimx</code> to avoid killing <code>\numberline</code>	424
1992-08-25 ltsect.dtx LaTeX2.09		1993-03-18 ltfsstrc.dtx v2.1c	
<code>\@sect:</code> (FMi) replaced explicit setting of <code>\@svsec</code> by call to <code>\@seccntformat</code>	710	<code>\DeclareSizeFunction:</code> Added all args to avoid blanks problems . .	442
1992-09-18 ltlists.dtx LaTeX2.09		1993-04-09 lterror.dtx v1.0e	
<code>\item:</code> (RmS) Added warning if <code>\item</code> is used in math mode	624	<code>\@latexerr:</code> Mention The Companion	257
1992-09-18 lttab.dtx LaTeX2.09		1993-04-11 lterror.dtx v1.0f	
<code>\@array:</code> Changed <code>\par</code> to <code>\@empty</code> to avoid starting new row e.g. after <code>\hline</code>	658	<code>\@latexerr:</code> Remove setting of <code>errorcontextlines</code>	257
1992-09-19 ltfsstrc.dtx v2.0c		1993-05-05 ltftncmd.dtx v2.0b	
<code>\try@simple@size:</code>	439	General: Removed all LaTeX related cmds	530
1992-09-21 ltffsini.dtx v1.4d		1993-05-16 ltffsbas.dtx v2.0e	
<code>\not@math@alphabet:</code> Macro defined. .	499	<code>\showhyphens:</code> Use <code>\reset@font</code> . .	399
1992-09-22 ltffsbas.dtx v1.91a		1993-07-16 ltfsstrc.dtx v2.1h	
General: Introduced <code>\tf@size</code> for math size.	376	General: Changed layout of info messages	424
1992-09-22 ltfsstrc.dtx v2.1a		1993-07-17 ltoutenc.dtx 1.0d	
<code>\getanddefine@fonts:</code> Introduced <code>\tf@size</code> for math size.	437	General: changed <code>\catcoding @</code> . . .	323
1992-11-13 ltffsini.dtx v?		1993-08-03 ltmiscen.dtx LaTeX2.09	
<code>\hexnumber@:</code> Made expandable. . . .	501	<code>\enddocument:</code> Changed redefinition of <code>\global</code> to redefinition of <code>\@setckpt.</code>	578
1992-11-23 ltcounts.dtx LaTeX2.09		1993-08-05 ltpictur.dtx LaTeX2.09	
<code>\stepcounter:</code> Replaced <code>{}</code> in <code>\stepcounter</code> by <code>\begingroup</code> <code>\endgroup</code> to avoid adding an empty ord in math mode	368	<code>\circle:</code> (RMS) Added error message if <code>\circle</code> is used in math mode. .	695
1992-11-26 ltboxes.dtx LaTeX2.09		1993-08-05 ltsect.dtx LaTeX2.09	
<code>\@mpfootnotetext:</code> (RmS) added protection for <code>\edef</code>	641	<code>\@sect:</code> (RmS) Made sure that <code>\protect</code> works correctly in expansion of <code>\the</code> counter	710
1992-11-26 ltfloat.dtx LaTeX2.09		1993-08-05 ltspac.dtx LaTeX2e	
<code>\@footnotetext:</code> (RmS) added protection for <code>\edef</code>	738	<code>\@hspace:</code> (RmS) Removed superfluous <code>\leavevmode</code> in <code>\@hspace</code> and <code>\@hspace,</code> as suggested by CAR.	296

1993-08-05 lttab.dtx latex2e		
<code>\tabular*</code> : Replaced		
<code>\expandafter\def</code> by <code>\@namedef</code> .	657	
1993-08-06 ltbibl.dtx LaTeX2.09		
<code>\@citex</code> : Moved writing to .aux file in		
loop over citation keys so that		
leading blanks are removed there		
as well.	746	
1993-08-13 ltoutenc.dtx 1.0f		
General: Protected against active @		
sign.	323	
1993-08-13 preload.dtx v2.0c		
General: Added <code>\relax</code> at end of font		
names.	528	
1993-08-16 ltoutenc.dtx 1.0g		
General: Needs space after <code>\string</code>	323	
1993-08-18 ltfsdcl.dtx v2.0e		
<code>\new@mathversion</code> : Exchanged names		
of encodings in warning message of		
<code>\SetSymbolFont</code> .	463	
1993-09-02 ltfsstrc.dtx v2.1i		
General: Corrected name of sgen size		
function.	424	
1993-09-03 ltmiscen.dtx LaTeX2.09		
<code>\verbatim@nolig@list</code> : Replaced		
<code>\@noligs</code> by extensible list	597	
1993-09-07 ltmiscen.dtx LaTeX2.09		
<code>\verb@balance@group</code> : (RmS)		
Changed definition of <code>\verb</code> so		
that it detects a missing second		
delimiter.	596	
1993-09-08 ltmiscen.dtx LaTeX2.09		
<code>\enddocument</code> : Added warning in case		
of undefined references.	578	
1993-09-15 ltfsbas.dtx v2.0g		
<code>\DeclareFontEncoding</code> : Corrected:		
<code>\default@T</code> to <code>\default@M</code> .	380	
1993-09-15 ltfsstrc.dtx v2.1j		
General: Corrected spelling of		
<code>\noexpand</code> .	424	
1993-09-19 lterror.dtx LaTeX2.09		
<code>\@invalidchar</code> : (RmS) Error message		
for invalid input characters.	260	
1993-11-02 ltmath.dtx LaTeX2.09		
General: RmS: Corrected description		
of <code>\@eqnset</code> , moved <code>\@eqnset</code>		
accordingly and removed extra		
<code>\tabskip</code> assignment.	607	
1993-11-03 ltmath.dtx LaTeX2e		
General: RmS: Initialized <code>\everycr</code> to		
empty	607	
1993-11-03 ltpictur.dtx LaTeX2.09		
General: (RmS) changed <code>\halign</code> to		
<code>\ialign</code> to initialize <code>\tabskip</code> and		
<code>\everycr</code>	675	
1993-11-11 ltfsini.dtx v2.1a		
<code>\normalfont</code> : Macro added	501	
1993-11-11 ltfsstrc.dtx v2.2a		
General: Option concept added for		
LaTeX2e	424	
1993-11-14 ltclass.dtx v0.2a		
<code>\@currentx</code> : Name changed from		
<code>\@currentextension</code>	759	
<code>\@reset@ptions</code> : macro added	783	
<code>\AtEndDocument</code> : Included extension		
in the generated macro name for		
package and class hooks.	784	
<code>\documentstyle</code> : Added		
<code>\RequirePackage</code>		
<code>\@unusedoptionlist</code> stuff.	773	
<code>\load@onefilewithoptions</code> : Moved		
resetting of <code>\default@ds</code> , <code>\ds@</code> and		
<code>\@declaredoptions</code> here, from the		
end of <code>\ProcessOptions</code> .	778	
<code>\NeedsTeXFormat</code> : made more robust		
for alternative syntax for other		
formats.	775	
<code>\ProcessOptions*</code> : Optimize ‘empty		
option’ code.	770	
Stop adding the global option list		
inside class files.	770	
1993-11-14 ltdefs.dtx v0.2a		
<code>\g@addto@macro</code> : Made global	102	
1993-11-15 ltclass.dtx v0.2b		
<code>\documentstyle</code> : Modified to match		
<code>\ProcessOption*</code>	773	
<code>\ProcessOptions*</code> : Star form added.	770	
1993-11-17 ltclass.dtx v0.2c		
<code>\@fileswith@ptions</code> : Macro added	783	
<code>\@badrequireerror</code> : Macro added	785	
<code>\@twoloadclasserror</code> : Macro added	785	
<code>\CurrentOption</code> : Name changed from		
<code>\@curroption</code>	759	
<code>\DeclareOption*</code> : Error checking		
added	769	
<code>\load@onefilewithoptions</code> : Added		
trap for two <code>\LoadClass</code>		
commands.	780	
<code>\NeedsTeXFormat</code> : Name changed from		
<code>\NeedsFormat</code>	775	
<code>\ProcessOptions*</code> : restoring		
<code>\@fileswith@ptions</code> added.	770	
1993-11-18 ltclass.dtx v0.2d		
<code>\documentstyle</code> : Modified		
<code>\RequirePackage</code> stuff.	773	
<code>\ExecuteOptions</code> : Use		
<code>\CurrentOption</code> not <code>\reserved@a</code>	773	

<code>\NeedsTeXFormat: \fmtname</code>		<code>\newcommand: Macro reimplemented</code>	
<code>\fmtversion not \@...</code>	775	and extended	76
1993-11-21 ltfiles.dtx LaTeX2e		<code>\renewcommand: Macro reimplemented</code>	
<code>\@missingfileerror: Stop infinite</code>		and extended	79
looping on <code>\@er@ext</code>	316	<code>\renewenvironment: Macro</code>	
1993-11-21 ltmiscen.dtx v0.9a		reimplemented and extended	80
<code>\@verbatim: use \verbatim@font</code>		<code>\two@digits: Macro added</code>	73
instead of <code>\tt</code>	593	1993-11-23 ltoutput.dtx v0.1a	
<code>\verb: Use \verbatim@font instead of</code>		<code>\paperheight: Register added</code>	864
<code>\tt.</code>	597	<code>\paperwidth: Register added</code>	864
<code>\verbatim@font: Macro added</code>	593	1993-11-23 ltoutput.dtx v0.1c	
1993-11-22 ltclass.dtx v0.2f		<code>\@enlargepage: Command added</code>	908
<code>\@fileswithoptions: Made the</code>		<code>\@kludgeins: Insert added</code>	908
default <code>[]</code> not <code>[\@unknownversion]</code>	776	<code>\@makecol: Command changed</code>	875
<code>\@ifl@ter: Added //00 so parsing</code>		<code>\@specialoutput: Command changed</code>	869
never produces a runaway		<code>\enlargethispage*: Commands</code>	
argument.	764	added	908
General: <code>\@unknownversion</code> removed	754	1993-11-24 ltfntcmd.dtx v2.1a	
<code>\load@onefilewithoptions: Made the</code>		<code>\maybe@ic@: Use \t@st@ic</code>	535
initial version <code>[]</code> not		<code>\t@st@ic: Macro added</code>	536
<code>[\@unknownversion]</code>	778	1993-11-24 ltfssini.dtx v2.1a	
1993-11-22 ltdefs.dtx LaTeX2e		General: Removed <code>\xpt</code> stuff	501
<code>\@minus: Macro added</code>	74	1993-11-24 ltlogos.dtx LaTeX2e	
<code>\@plus: Macro added</code>	74	<code>\LaTeX: Macro changed</code>	298
<code>\CheckCommand: Macro added</code>	81	1993-11-28 ltclass.dtx v0.2h	
<code>\providecommand: Macro added</code>	81	<code>\@twoclasseserror: Macro added</code>	786
1993-11-22 lterror.dtx LaTeX2e		General: Assorted commands now in	
<code>\c@errorcontextlines: Macro added</code>	257	the kernel removed.	758
1993-11-22 ltfiles.dtx LaTeX2e		Directory syntax checking moved to	
<code>\listfiles: Removed checking for</code>		<code>dircheck.dtx</code>	758
<code>\@unknownversion</code>	318	Primitive filenames now terminated	
1993-11-22 ltlength.dtx LaTeX2e		by space not <code>\relax.</code>	758
<code>\@settodim: Macro added</code>	374	<code>\endfilecontents: Don't globally</code>	
<code>\@settopoint: Macro added</code>	375	allocate a write stream (always use	
<code>\settodepth: Macro added</code>	374	15)	786
<code>\settoheight: Macro added</code>	374	1993-11-28 ltfiles.dtx LaTeX2e	
1993-11-22 ltlogos.dtx LaTeX2e		<code>\@missingfileerror: Use filename</code>	
<code>\LaTeXe: Macro added</code>	298	parser from <code>dircheck</code>	316
1993-11-23 ltclass.dtx v0.2g		1993-11-29 ltoutput.dtx v1.0b	
<code>\@use@option: Name changed from</code>		<code>\@makecol: \@makespecialcolbox</code>	
<code>\@executeoption</code>	772	added	875
General: Various macros now moved		<code>\@makespecialcolbox: Command</code>	
to latex.tex.	758	added	877
Warnings and errors now directly		1993-11-29 ltplain.dtx LaTeX2e	
coded.	758	General: All accents in decimals;	
1993-11-23 ltdefs.dtx LaTeX2e		suggested by Paul Taylor	31
<code>\@argdef: Macro added</code>	77	1993-11-30 ltoutput.dtx v1.0c	
<code>\@ifundefined: Redefined to remove a</code>		<code>\fl@tracemessage: Commands added</code>	910
trailing <code>\fi</code>	96	1993-12-01 fontdef.dtx v2.1a	
<code>\@newcommand: Macro added</code>	77	General: Update for LaTeX2e	506
<code>\@newenv: Macro interface changed</code>	80	1993-12-01 ltoutput.dtx v1.0e	
<code>\@xargdef: Macro interface changed</code>	77	<code>\@reinserts: Command added</code>	877
<code>\@yargd@f: Avoid \@?@? token</code>	78	1993-12-03 ltboxes.dtx v0.1a	
Macro interface changed	78	<code>\@argsbox: macro removed</code>	643

<code>\@begin@tempboxa</code> : macro added . . .	631	1993-12-05 ltfloat.dtx LaTeX2e	
<code>\@end@tempboxa</code> : macro added	631	<code>\@dblfloatplacement</code> : Command	
<code>\@iirsbox</code> : redefined to support		changed	729
<code>\height</code>	643	<code>\@xfloat</code> : Command changed	723
<code>\@imakebox</code> : macro modified	631	1993-12-05 ltoutput.dtx v1.0f	
<code>\@iirsbox</code> : redefined to support		<code>\@addtobot</code> : Command changed . . .	889
<code>\height</code>	643	<code>\@addtocurcol</code> : Command changed	891
<code>\@isavebox</code> : color support	634	<code>\@addtodblcol</code> : Command changed	902
extra group	634	<code>\@addtonextcol</code> : Command changed	898
<code>\@isavepicbox</code> : extra group	634	<code>\@addtotoporbot</code> : Command changed	890
<code>\@makebox</code> : default changed from x to		<code>\@boxfpsbit</code> : Command added . . .	913
c	631	<code>\@flcheckspace</code> : Command added .	915
<code>\@makepicbox</code> : macro modified	632	<code>\@flsetnum</code> : Command added	914
<code>\@savebox</code> : default c not x	634	<code>\@flsettextmin</code> : Command added .	915
<code>\bm@b</code> : macros added	631	<code>\@flstop</code> : Commands added	911
<code>\endlrbox</code> : macro added	635	<code>\@flupdates</code> : Command added . . .	916
<code>\fbox</code> : extra group	635	<code>\@fpsadddefault</code> : Command added	912
<code>\lrbox</code> : color support	634	<code>\@getfpsbit</code> : Command added . . .	913
macro added	634	<code>\@opcol</code> : Command changed	875
<code>\makebox</code> : modified	630	Hook added	875
<code>\mbox</code> : extra group	631	<code>\@outputpage</code> : Command changed .	879
<code>\minipage</code> : Redefined to support extra		<code>\@resethfps</code> : Command added . . .	914
optional arguments	639	<code>\@setfloattyperecounts</code> : Command	
<code>\newsavebox</code> : Pass the whole of arg 1		added	912
to <code>\@ifdefinable</code>	633	<code>\@setfpsbit</code> : Command added . . .	914
<code>\parbox</code> : Redefined to support extra		<code>\@shipoutsetup</code> : Command added .	879
optional arguments	637	<code>\@startcolumn</code> : Command changed	885
<code>\raisebox</code> : redefined to support		<code>\@startdblcolumn</code> : Command	
<code>\height</code>	642	changed	885
<code>\sbox</code> : color support	634	<code>\@testfp</code> : Command added	913
extra group	634	<code>\@textfloatsheight</code> : Commands	
<code>\set@color</code> : color support	633	added	912
macro added	633	<code>\@topnewpage</code> : Commands changed	867
1993-12-03 ltclass.dtx v0.2i		<code>\@tryfcolumn</code> : Command changed .	886
<code>\@cls@pkg</code> : Name changed to avoid		<code>\@writesetup</code> : <code>\@startpagehook</code>	
clash with output routine.	785	added	879
General: <code>\@onlypreamble</code> : Many		<code>\@output</code> : Command changed	869
commands declared.	758	1993-12-06 ltclass.dtx v0.2k	
Removed obsolete		<code>\ExecuteOptions</code> : Preserve	
<code>\@documentclass</code>	758	<code>\CurrentOption</code>	773
1993-12-03 lterror.dtx v1.0b		1993-12-06 ltoutput.dtx v1.0f	
<code>\@latexerr</code> : Set		<code>\@specialoutput</code> : Unboxing of 255	
<code>\@errorcontextlines</code> to -1 . . .	257	added to rescue writes	869
1993-12-03 ltfssini.dtx v2.1a		1993-12-06 ltoutput.dtx v1.0g	
General: update for LaTeX2e	481	<code>\@topnewpage</code> : <code>\@floatplacement</code>	
1993-12-04 ltfilehook.dtx v0.9b		placement bug fixed	867
<code>\unqu@tefilef@und</code> : Macro added .	813	1993-12-07 ltclass.dtx v0.2l	
1993-12-04 ltfiles.dtx v0.9b		<code>\ProvidesFile</code> : Macro added	767
<code>\@iinput</code> : Macro reimplemented . .	315	1993-12-07 ltclass.dtx v0.2m	
<code>\@input</code> : Macro reimplemented . .	316	<code>\load@onefilewithoptions</code> : Reset	
<code>\IfFileExists@</code> : Macro added . . .	312	<code>\CurrentOption</code>	778
<code>\input</code> : Macro reimplemented . . .	315	1993-12-07 ltoutenc.dtx 1.1	
		General: Protected all special	
		characters with <code>\string</code>	323

1993-12-07 ltoutenc.dtx v1.1	1993-12-11 ltmath.dtx v0.9g
General: Made all character numbers	General: Added a group around the
decimal. 320	first argument of <code>\frac</code> to prevent
Removed a lot of equal signs and	changes (for example font changes)
the like. 320	from modifying the contents of the
1993-12-08 ltboxes.dtx v0.1b	second argument. 607
<code>\@begin@tempboxa</code> : Extra braces for	1993-12-11 ltoutenc.dtx v1.2a
color support (braces removed	General: Corrected for <code>tlenc</code> , <code>math</code> . 320
from other macros) 631	1993-12-11 ltsect.dtx LaTeX2e
<code>\@irsbox</code> : fix typo 643	<code>\author</code> : Added default 706
<code>\@parboxto</code> : <code>\endgraf</code> added due to	<code>\title</code> : Added default 706
extra group in <code>\@begin@tempboxa</code> 638	1993-12-11 ltxref.dtx LaTeX2e
<code>\lrbox</code> : move <code>\@endpfalse</code> out of the	<code>\setref</code> : Macro added 574
inner group 634	<code>\pageref</code> : Macro reimplemented . . . 574
1993-12-08 ltfntcmd.dtx v2.1b	<code>\ref</code> : Macro reimplemented 574
General: Macros <code>\rm</code> , <code>\bf</code> and <code>\sf</code>	1993-12-12 ltoutput.dtx v1.0h
moved to <code>classes.dtx</code> 538	<code>\@cflb</code> : <code>boxmaxdepth</code> setting moved 883
1993-12-08 ltlists.dtx LaTeX2e	defs changed to <code>lets</code> 883
<code>\@item</code> : use <code>\sbox</code> to support colour 625	<code>\@cflt</code> : name changed 883
1993-12-08 ltspace.dtx LaTeX2e	<code>\@docclearpage</code> : defs changed to
<code>\@bsphack</code> : Command reimplemented 286	<code>lets</code> 874, 875
Command reimplemented; late	<code>\@makecol</code> : defs changed to <code>lets</code> . . . 876
birthday present for Chris 286	<code>\@resetthfps</code> : Warnings added:
<code>\@vbsphack</code> : Command added 289	minimal 914
1993-12-09 ltboxes.dtx v0.1c	<code>\@startdblcolumn</code> : defs changed to
<code>\@irsbox</code> : fix another typo 643	<code>lets</code> 885, 886
1993-12-09 ltclass.dtx v0.2n	<code>\@topnewpage</code> : braces removed 867
<code>\documentstyle</code> : input 209	<code>\@tryfcolumn</code> : defs changed to <code>lets</code> . 886
compatibility file. 773	<code>\fl@tracemessage</code> : Commands
1993-12-09 ltfiles.dtx v0.9e	changed 910
<code>\document</code> : Hook added 301	1993-12-13 ltclass.dtx v0.2o
1993-12-09 ltmiscen.dtx v0.9e	General: Removed setting
<code>\enddocument</code> : Hook added 578	<code>\errorcontextlines</code> (now in
1993-12-10 ltoutenc.dtx v1.2	<code>latex.tex</code>) 758
General: Added source code for	<code>\documentstyle</code> : compatibility file
<code>tlenc.sty</code> 320	now <code>latex209.sty</code> 773
1993-12-11 ltfntcmd.dtx v3.0a	<code>\usepackage</code> : Fixed error handling . 775
General: Complete reworking of all	1993-12-13 ltdirchk.dtx v0.2a
text commands, using just one	General: on the ‘docstrip’ pass, do not
creator function 530	check <code>openin</code> path 10
italic correction now put in front of	<code>\IfFileExists</code> : Removed interactive
penalty before glue 530	prompting for current directory
newcommands replaced by defs . . 530	syntax 10
newfontswitch command corrected	<code>\strip@prefix</code> : modified, name
and changed 530	changed from <code>\stripmeaning</code> 5
<code>\DeclareTextFontCommand</code> : Macro	1993-12-13 ltlists.dtx latex2e
changed 532	<code>\trivlist</code> : Initialised <code>\@itemlabel</code> 622
<code>\emph</code> : Macro changed 533	1993-12-13 ltmiscen.dtx v0.9h
<code>\fix@penalty</code> : Macro added 536	<code>\@noligs</code> : Readded <code>\@noligs</code> 597
<code>\maybe@ic</code> : Macro name changed . . 535	<code>\@verbatim</code> : Readded <code>\@noligs</code> . . . 593
<code>\maybe@ic@</code> : Macro and name	Removed optional argument of
changed 535	<code>\item</code> 592
<code>\sw@slant</code> : Macro changed 536	<code>center</code> : Removed optional argument
<code>\textup</code> : Macros changed 533	of <code>\item</code> 590

<code>flushleft</code> : Removed optional argument of <code>\item</code>	591	Removed the catcode hackery, since the file is only read as a package in the preamble, and removed all the messages on the screen, which just confuse users. Replaced them by the appropriate <code>\ProvidesPackage</code> commands. Added <code>XXXenc</code>	323
<code>flushright</code> : Removed optional argument of <code>\item</code>	591		
1993-12-13 <code>ltoutenc.dtx</code> v1.2b			
General: Corrected file name in driver code.	320		
1993-12-13 <code>lftab.dtx</code> latex2e		1993-12-17 <code>ltoutenc.dtx</code> v1.3	
<code>\tabbing</code> : Removed optional argument of <code>\item</code>	652	General: Added	
1993-12-14 <code>ltoutput.dtx</code> v1.0i		<code>\EncodingSpecificAccent</code> , <code>\EncodingSpecificAccentedLetter</code> and <code>\EncodingSpecificCommand</code>	320
General: Section added to declare all parameters	922	Made Rokicki's encoding a proper encoding scheme rather than a variant of OT1.	320
1993-12-15 <code>ltboxes.dtx</code> v0.1d		1993-12-17 <code>ltoutput.dtx</code> v1.0j	
<code>\iminipage</code> : Changed default from 'c' to 's'	640	<code>\@opcol</code> : Hook removed	875
<code>\iparbox</code> : Changed default from 'c' to 's'	637	<code>\@specialoutput</code> : Page room test added	870
<code>\minipage</code> : Changed default from 'c' to 's'	639	<code>\@topnewpage</code> : check for vsize too small added	867
extra space removed.	639	Page room test added	869
<code>\parbox</code> : Changed default from 'c' to 's'	637	<code>\@writesetup</code> : —and then removed	879
1993-12-15 <code>ltclass.dtx</code> v0.2p		<code>\fl@tracemessage</code> : tracefloatvals made a document command	910
General: Removed extra 's' from <code>@@warnings</code>	758	1993-12-17 <code>ltpage.dtx</code> LaTeX2e	
1993-12-16 <code>ltlogos.dtx</code> LaTeX2e		<code>\mark</code> : Removed init <code>\mark</code> at begin document, since it doesn't work. . . .	752
<code>\LaTeXe</code> : Extended logo by DPC	298	<code>\rightmark</code> : Stopgap solution to mark <code>\leftmark</code> and <code>\rightmark</code> work without initializing mark until the problem is solved.	752
1993-12-16 <code>ltmath.dtx</code> v0.9i		1993-12-18 <code>ltoutenc.dtx</code> 1.3b	
<code>\@eqnocr</code> : use <code>\refstepcounter</code> instead of shortcut	609	General: Fixed typos with <code>\ProvidesPackage</code> lines. Added the <code>\NeedsTeXFormat</code> line. Added the last argument to <code>\DeclareEncoding</code> . Moved the use of the encodings to after their declaration.	323
General: use <code>\refstepcounter</code> instead of shortcut	607	Replaced the missing last argument to <code>\DeclareFontEncoding</code>	336, 338
1993-12-16 <code>ltmiscen.dtx</code> v0.9i		1993-12-18 <code>ltoutenc.dtx</code> 1.3c	
General: <code>\literal</code> added	597	General: Rewrote for the new syntax of <code>\EncodingSpecific</code>	336, 338
1993-12-16 <code>ltpage.dtx</code> LaTeX2e		Split <code>\EncodingSpecificAccent</code> up into <code>\EncodingSpecific</code> and <code>\DeclareAccent</code>	324
<code>\mark</code> : Init <code>\mark</code> at begin document	752	1993-12-18 <code>ltoutenc.dtx</code> v1.3a	
1993-12-16 <code>ltspace.dtx</code> LaTeX2e		General: Replaced OT3 by XXX	320
<code>\@bsphack</code> : Corrected optimisation :-)	286	1993-12-18 <code>ltoutenc.dtx</code> v1.3b	
1993-12-16 <code>lftab.dtx</code> latex2e		General: Corrected typos.	320
<code>\@xhline</code> : Measure from middle of vertical rules	667	Replaced the missing last argument to <code>\DeclareFontEncoding</code>	320
1993-12-17 <code>ltclass.dtx</code> v0.2q			
<code>\@documentclasshook</code> : Macro added	758		
<code>\@fileswithoptions</code> : Add <code>\@compatibility</code> hook	776		
<code>\documentstyle</code> : Match Alan's new code.	773		
1993-12-17 <code>ltoutenc.dtx</code> 1.3			
General: Added this section	324		
Removed all the hackery for use in <code>\DeclareFontEncoding</code> , and redid everything using <code>\DeclareTextFoo</code>	336, 338		

1993-12-18 ltoutenc.dtx v1.3c	1994-01-17 ltclass.dtx v0.2s
General: A new syntax, separating accent-definitions from encoding-specific definitions, and allowing encoding-specific <code>\chardef</code> , <code>\let</code> , etc. 320	<code>\@fileswithoptions</code> : Modify to reduce parameter stack usage . . . 776
Rewrote for the new syntax of <code>\EncodingSpecific</code> 320	General: Added many more <code>\@onlypreamble</code> commands 758
1993-12-18 ltoutenc.dtx v1.3d	Wrapped long lines to column 72 758
General: Some T1 stuff had drifted into the OT1 file. 320	<code>\load@onefile@withoptions</code> : Modify to reduce parameter stack usage 781
1993-12-18 ltpage.dtx LaTeX2e	1994-01-17 ltfiles.dtx LaTeX2e
<code>\sloppy</code> : Added <code>\emergencystretch</code> 752	<code>\listfiles</code> : New Version, adds ‘tex’ if needed, and lines up columns . 318
1993-12-19 ltclass.dtx v0.2r	1994-01-17 ltfssbas.dtx v2.1a
<code>\endfilecontents</code> : Different message when ignoring a file 786	General: New math font setup 376
1993-12-19 ltfntcmd.dtx v3.0b	<code>\curr@math@size</code> : New math font setup 388
General: <code>\@pdef</code> command added . . 530	<code>\everydisplay</code> : New math font setup 388
Added by ASAJ. 538	<code>\everymath</code> : New math font setup . 388
Made <code>\@newfontswitch</code> produce an error if command already exists, and added <code>\@renewfontswitch</code> , ASAJ 530	<code>\frozen@everydisplay</code> : New math font setup 388
Other tidying 530	<code>\frozen@everymath</code> : New math font setup 388
Some more tidying done 530	<code>\math@version</code> : New math font setup 387
Untidying added, so this is now a TEMPORARY version. 530	1994-01-17 ltfssini.dtx v2.1e
Wording changes by CAR. 538	<code>\not@math@alphabet</code> : Message changed 499
<code>\DeclareOldFontCommand</code> : Corrected and tidied 537	1994-01-17 ltfssstrc.dtx v2.3a
<code>\DeclareTextFontCommand</code> : Corrected and tidied 532	General: New math font setup 424
1993-12-19 ltspace.dtx LaTeX2e	<code>\check@mathfonts</code> : New math font setup 434
<code>\@bsphack</code> : There seem to be problems with selfmade birthday presents . 287	<code>\glb@currsize</code> : New math font setup 432
1993-12-20 ltdefs.dtx LaTeX2e	<code>\restglb@settings</code> : New math font setup 435
<code>\@reargdef</code> : Kept old version of <code>\@reargdef</code> , for array.sty 78	1994-01-18 ltbibl.dtx LaTeX2e
1993-12-20 ltfiles.dtx v0.9m	<code>\bibliography</code> : Use <code>\@input@</code> so include files are listed. 747
<code>\@obsoletefile</code> : Added this command, removed <code>@oldfilewarning</code> 318	1994-01-18 ltclass.dtx v0.2t
1994-01-05 fontdef.dtx v2.1d	<code>\@ifclassloaded</code> : Fix typo <code>\@pkgetension</code> 762
General: Removed nf prefix from file names. 508	1994-01-18 ltfilehook.dtx v0.9p
1994-01-13 ltmath.dtx v0.9o	<code>\unqu@tefilef@und</code> : New Definition 813
<code>\@eqncr</code> : correcting 0.9i 609	1994-01-18 ltfiles.dtx v0.9p
General: correcting 0.9i 607	<code>\@iffileonpath</code> : Macro added 314
1994-01-14 ltdirchk.dtx v0.2d	<code>\@input</code> : do not use a different definition for <code>\input@path</code> 316
<code>\IfFileExists</code> : Close the texsys.aux output stream 10	<code>\@input@</code> : Macro added 316
1994-01-15 ltfiles.dtx v0.9o	<code>\IfFileExists@</code> : New Definition . . 312
<code>\document</code> : move <code>\@preamblecmds</code> after document hook 303	<code>\includeonly</code> : Use <code>\@input@</code> so include files are listed. 307
	1994-01-18 ltfssini.dtx v2.1f
	<code>\not@math@alphabet</code> : Message corrected 499
	1994-01-18 ltmiscen.dtx v0.9p
	<code>\@verbatim</code> : Add <code>\global\@inlabelfalse</code> 592

Only add <code>\penalty</code> if in hmode .	592	<code>\restgbl@settings</code> : Correct trace info placement	435
1994-01-19 fontdef.dtx v2.1e		1994-01-27 ltfntcmd.dtx v3.1a	
General: Added missing setting for symbols in bold version.	513	<code>\nocorrlist</code> : Only <code>.</code> , used as default for cm fonts	537
1994-01-19 ltdirchk.dtx v0.2e		1994-01-29 ltclass.dtx v0.2v	
<code>\IfFileExists</code> : name changed from <code>\test</code>	9	<code>\@unprocessedoptions</code> : Macro added.	785
<code>\input@path</code> : No longer check that an empty group is in the path	10	<code>\load@onefile@withoptions</code> : All options raise error if no <code>\ProcessOptions</code> appears	781
<code>\strip@prefix</code> : name changed from <code>\strip@meaning</code> , to match NFSS.	5	1994-01-31 ltdefs.dtx v0.2w	
1994-01-19 ltmath.dtx v1.0n classes		<code>\g@addto@macro</code> : Use toks register to avoid ‘hash’ problems	102
<code>\mathindent</code> : Deferred setting of <code>\mathindent</code>	610	1994-01-31 ltfiles.dtx v0.9t	
1994-01-20 ltdirchk.dtx v0.2f		<code>\document</code> : set <code>\@normalsize</code> or <code>\normalsize</code> if necessary	302
General: <code>\@copytexsys</code> and the <code>texsys.new</code> file removed	8	1994-01-31 ltfntcmd.dtx v3.1b	
Modify all of <code>ltxcheck</code>	13	General: <code>\@normalsize</code> no longer defined	530
<code>\IfFileExists</code> : <code>\@copytexsys</code> removed	10	1994-02-01 ltpage.dtx LaTeX2e	
1994-01-21 ltclass.dtx v0.2u		<code>\pagestyle</code> : (DPC) Modify to get nicer error message	750
<code>\documentstyle</code> : compatibility file now <code>latex209.def</code>	773	<code>\thispagestyle</code> : (DPC) Modify to get nicer error message	751
1994-01-21 ltdirchk.dtx v0.2g		1994-02-02 ltclass.dtx v0.2x	
General: Improve documentation, reorganize <code>docstrip</code> module	1	<code>\load@onefile@withoptions</code> : Only run the hook and options check if the file was loaded.	781
<code>\filename@parse</code> : Minor changes, and add Mac version (<code>:</code>)	11	1994-02-03 ltoutput.dtx v1.0k	
<code>\today</code> : Name changed from <code>\stamp</code> , to save memory	9	<code>\@makespecialcolbox</code> : correct mistakes in the documentation	878
1994-01-21 ltfloat.dtx LaTeX2e		1994-02-07 ltclass.dtx v0.2y	
<code>\xfloat</code> : Added missing percent characters.	723	<code>\@fileswithoptions</code> : Run <code>\@compatibility</code> on the first class to start (not the first to finish)	776
1994-01-21 ltmiscen.dtx v0.9s		<code>\@ifclasswith</code> : Add extra <code>,s</code> so ‘two’ is not matched with ‘twocolumn’	765
<code>\verbatim@font</code> : Removed unnecessary category code hackery.	593	<code>\ProcessOptions*</code> : Add extra <code>,s</code> so ‘two’ is not matched with ‘twocolumn’	770
1994-01-24 ltdirchk.dtx v0.2h		1994-02-07 ltfssbas.dtx v2.1c	
<code>\IfFileExists</code> : Stop testing once <code>texsys.aux</code> has been found	9	<code>\DeclareFontEncoding</code> : revert catcode settings earlier	380
1994-01-24 ltpage.dtx LaTeX2e		<code>\DeclareFontShape@</code> : revert catcode settings earlier	377
<code>\pagestyle</code> : (DPC) Complain if <code>pagestyle</code> is undefined.	750	1994-02-08 ltoutput.dtx v1.0k	
1994-01-25 ltdirchk.dtx v0.2i		<code>\@makespecialcolbox</code> : <code>boxmaxdepth</code> setting added	878
General: Protect against looping on <code>\@input</code> and <code>\@end</code>	2	<code>boxmaxdepth</code> setting removed	877
1994-01-25 ltfssbas.dtx v2.1b		General: Documentation and tasks tidied.	852
<code>\math@version</code> : Corrections for math setup	387		
1994-01-25 ltmath.dtx LaTeX2e			
<code>\bordermatrix</code> : Removed <code>\p@renwd</code>	602		
1994-01-26 ltfsssrc.dtx v2.3c			
<code>\check@mathfonts</code> : Correct trace info placement	434		

1994-02-10 ltclass.dtx v0.2z	1994-03-07 ltboxes.dtx v0.1a
\@documentclasshook: Changed the name from \@compatibility to \@documentclasshook, and added the check for whether \@normalsize has been defined. ASAJ. 758	\@mpfootnotetext: Extra group for color 641
\@fileswithoptions: Renamed \@compatibility to \@documentclasshook. ASAJ. . . 776	1994-03-07 ltboxes.dtx v1.0a
1994-02-10 ltfssbas.dtx v2.1d	General: Unify format with other Kernel files 630
\addto@hook: Made \addto@hook long. 401	1994-03-07 ltdefs.dtx v1.0a
1994-02-10 ltfsscmp.dtx v2.1d	\@italiccorr: Macro added 74
\scan@fontshape: scan away stuff after pt 450	1994-03-07 ltfiles.dtx v1.0a
1994-02-22 ltfssini.dtx v2.1g	General: Initial version, split from latex.dtx 299
General: Correct error message 503	Long lines wrapped to 72 columns 299
1994-02-24 ltfssbas.dtx v2.1e	1994-03-07 ltfinal.dtx v0.1a
\DeclareFontShape: Separate restoration of catcodes for fd cmds 377	General: Add code from the old dump.dtx 941
\define@newfont: Separate restoration of catcodes for fd cmds 390	Initial version, split from latex.dtx 927
\nfss@catcodes: Separate restoration of catcodes for fd cmds 390	move code here from lhyphen.dtx 933
1994-02-25 ltdirchk.dtx v0.2j	Remove oldcomments environment 927
General: Remove need for drv file 1	use \InputIfFileExists not \IfFileExists 933
1994-03-01 ltdirchk.dtx v0.2k	1994-03-07 ltfloat.dtx v1.0a
General: Add unstripped module, so that dircheck.dtx may be used with initex 1	\@endfloatbox: (DPC) Extra group for colour 728
1994-03-02 ltboxes.dtx v0.1e	\@footnotetext: (DPC) Extra group for colour 738
General: Add 2kernel module 630	\@xfloat: (DPC) Extra group for colour 724
Remove need for drv file 630	1994-03-07 lthyphen.dtx v0.1c
1994-03-02 ltclass.dtx v0.3a	General: move the 2kernel code to ltfinal.dtx 925
General: Remove need for driver file 758	1994-03-07 ltlength.dtx v1.0a
1994-03-03 ltboxes.dtx v0.1f	\@settodim: (DPC) Extra group for colour 374
\@irsbox: Replaced a missing \else 643	1994-03-07 ltlists.dtx v1.0a
1994-03-04 ltfloat.dtx v1.0a	General: Initial version, split from latex.dtx 614
General: Initial version, split from latex.dtx 719	Long lines wrapped to 72 columns 614
1994-03-04 ltsect.dtx v1.0a	1994-03-07 ltpage.dtx v1.0a
General: Initial version, split from latex.dtx 706	General: Initial version, split from ltherest.dtx 750
1994-03-04 lttab.dtx v1.0a	1994-03-07 ltpictur.dtx v0.1a
General: Initial version, split from latex.dtx 645	General: Initial version, split from latex.dtx 670
1994-03-04 ltvers.dtx v1.0a	Long lines wrapped to 72 columns 670
General: Initial version, split from latex.dtx 36	1994-03-07 ltsect.dtx v1.0a
	\@hangfrom: (DPC)Extra groups for colour 713
	1994-03-07 lttab.dtx v1.0a
	General: Long lines wrapped to 72 columns 645
	1994-03-08 ltclass.dtx v0.3b
	General: Modify driver code into ‘new style’ 758

1994-03-08 ltdirchk.dtx v1.0a		<code>\listfiles</code> : Reset <code>\@addtofilelist</code> at begin document	318
General: Reorganize driver module into ‘new style’	1		
1994-03-08 ltplain.dtx v1.0a		1994-03-13 ltssbas.dtx v2.1g	
General: Remove need for a driver file.	14	General: add 2kernel module to omit repeated code	376
1994-03-10 ltssbas.dtx v2.2f		1994-03-13 ltssdcl.dtx v2.1c	
<code>\math@egroup</code> : Changed		General: add 2kernel module to omit repeated code	454
<code>\begingroup/\endgroup</code> to <code>\bgroup/\egroup</code>	399	1994-03-14 ltboxes.dtx v1.0b	
1994-03-11 ltssdcl.dtx v2.1b		<code>\@isavebox</code> : Use <code>\color@setgroup</code>	634
<code>\DeclareSymbolFontAlphabet@</code> :		<code>\@isavepicbox</code> : Use	
Added check against use of alphabet switch outside of math mode.	479	<code>\color@setgroup</code>	634
<code>\SetMathAlphabet@</code> : Changed		<code>\color@begingroup</code> : macro added for color support	633
parameter template in temporary macro to catch check add below.	468	<code>\color@endgroup</code> : macro added for color support	633
1994-03-12 ltclass.dtx v0.3c		<code>\lrbox</code> : Use <code>\color@setgroup</code>	634
General: Change name from docclass to ltclass	758	<code>\sbox</code> : Use <code>\color@setgroup</code>	634
<code>\ProvidesFile</code> : Add <code>\wlog</code>	767	1994-03-14 ltfloat.dtx 1.0c	
<code>\ProvidesPackage</code> : Add <code>\wlog</code>	765	<code>\xympar</code> : (DPC) Use	
use <code>\@gtempa</code>	765	<code>\color@begingroup</code>	733
1994-03-12 ltdefs.dtx v1.0b		1994-03-14 ltfloat.dtx v1.0c	
<code>\@reargdef</code> : New defn, in terms of <code>\@yargdef</code>	78	<code>\@endfloatbox</code> : (DPC) Use	
<code>\@yargd@f</code> : Name changed from <code>\XXX@argdef</code>	78	<code>\color@endgroup</code>	728
1994-03-12 ltdirchk.dtx v1.0b		<code>\@footnotetext</code> : (DPC) Use	
General: Change name from dircheck.dtx	1	<code>\color@begingroup</code> , add <code>\endgraf</code>	738
Minor edits to the typeouts in ltcheck	1	<code>\@savemarbox</code> : (DPC) Use	
1994-03-12 ltfloat.dtx v1.0b		<code>\color@begingroup</code>	732
<code>\@savemarbox</code> : (DPC) Extra group for colour	732	<code>\@xfloat</code> : (DPC) Use	
<code>\@xympar</code> : (DPC) Extra bgroup for colour	733	<code>\color@begingroup</code>	724
1994-03-12 ltplain.dtx v1.0b		1994-03-15 ltfiles.dtx LaTeX2e	
General: Name changed from lplain. The end of an era	14	<code>\@missingfileerror</code> : Quit on x or X just like a real error	316
1994-03-12 ltplain.dtx v1.0e		1994-03-15 ltfntcmd.dtx v3.2a	
General: Replaced remaining width, height, depth by L ^A T _E X macro names to save tokens.	14	General: Adapted to mass formatting	530
1994-03-13 ltcntrl.dtx v1.0c		Changed <code>\/</code> to <code>\@@italiccorr</code>	530
<code>\@tfor</code> : (DPC) Add <code>\@tf@r</code> so a single group is correctly treated.	251	Removed <code>\@renewfontswitch</code>	530
1994-03-13 ltfilehook.dtx v0.3b		Removed defs of short-forms and all sizes except <code>\normalize</code>	530
<code>\unqu@tefilef@und</code> : Use new cmd <code>\@addtofilelist</code>	813	1994-03-15 ltoutput.dtx v1.0l	
1994-03-13 ltfiles.dtx LaTeX2e		<code>\@addtocurcol</code> : Changed <code>\advspace</code> to <code>\vskip</code>	893, 897
<code>\@addtofilelist</code> : Macro added	318	<code>\@combinedblfloats</code> : Removed boxmaxdepth setting.	884
		<code>\@makecol</code> : <code>\maxdepth</code> changed to <code>\@maxdepth</code>	876
		Removed boxmaxdepth setting.	876
		<code>\@makespecialcolbox</code> : Removed boxmaxdepth setting.	878
		<code>\@topnewpage</code> : Corrected and amended warning message	868
		Warning added: it should be improved	869

General: Added some warnings when page gets full of top floats.	852	1994-03-29 ltcounts.dtx v1.0c	General: Create file from parts of ltmiscen and ltherest.	366
Driver added and further tidying.	852	1994-03-29 ltlength.dtx v1.0c	General: Create file ltcntlen from parts of ltmiscen and ltherest.	374
Removed duplicated code and corrected docstrip options.	852	1994-03-29 ltmiscen.dtx v1.0d	General: Remove counter macros to ltcntlen	577
Some boxmaxdepth settings removed.	852	1994-03-29 ltpageno.dtx v1.0c	General: Create file ltcntlen from parts of ltmiscen and ltherest.	571
1994-03-16 ltclass.dtx v0.3f		1994-03-29 ltxref.dtx v1.0c	General: Create file ltcntlen from parts of ltmiscen and ltherest.	572
General: Add pkgindoc package . . .	802	1994-03-31 ltbibl.dtx v1.0a	General: Initial version of ltidxbib.dtx, split from ltherest.dtx	745
1994-03-16 ltfiles.dtx LaTeX2e		1994-03-31 ltidxglo.dtx v1.0a	General: Initial version of ltidxbib.dtx, split from ltherest.dtx	742
\listfiles: Move this code directly into \document	318	1994-04-09 ltcounts.dtx v1.0d	\@newctr: \@nocnterr now has counter name argument	367
1994-03-16 ltfiles.dtx v1.0c			\addtocounter: \@nocnterr now has counter name argument	367
\document: (DPC) directly add file list settings	303		\setcounter: \@nocnterr now has counter name argument	367
1994-03-16 ltmiscen.dtx v1.0b			\stepcounter: Use \addtocounter to have name checked	368
\@verbatim: Remove			1994-04-09 ltthm.dtx v1.0b	
\global\@inlabelfalse again. . .	592		\@othm: Use standard counter error message (FMi)	704
1994-03-28 ltalloc.dtx v1.0d			1994-04-11 ltclass.dtx v0.3g	
General: Redefinition of ‘new’ allocations removed.	246		\endfilecontents: Add star form, don’t write \endinput at the end of the file.	786
1994-03-28 ltdirchk.dtx v1.0d			\ProvidesFile: Protect against weird catcodes.	767
General: Improve documentation . . .	1		1994-04-11 ltfsbas.dtx v2.1h	
1994-03-28 lterror.dtx v1.0d			General: Added \defaultscriptratio and \defaultscriptscriptratio. ASAJ.	376
\@invalidchar: (DPC) Comment out (use catcode15 instead)	260		\defaultscriptratio: Macro added	399
General: Remove test for \inputlineno undefined.	257		\defaultscriptscriptratio: Macro added	399
1994-03-28 ltfiles.dtx v1.0d			1994-04-12 ltboxes.dtx v1.0c	
\document: (DPC) Use \normalsize not \@normalsize	302		General: Remove \@acci, now defined in ltplain.dtx	638
(DPC) remove			Remove \@dischyph, now defined in ltinit.dtx	638
\@normalsize check	302		1994-04-12 ltdefs.dtx v1.0g	
1994-03-28 ltfloat.dtx v1.0b			\@dischyph: Define \@dischyph, was previously in ltboxes.dtx	100
\@caption: Use \normalsize not \@normalsize	722			
General: Split further from ltherest.dtx	719			
1994-03-28 ltlists.dtx v1.0b				
General: Improve documentation . .	613			
1994-03-28 ltmiscen.dtx v1.0c				
General: Improve Documentation . .	577			
1994-03-28 ltplain.dtx v1.0c				
\newlanguage: Remove some \outer declarations.	17			
1994-03-28 ltsect.dtx v1.0b				
General: Split further from ltherest.dtx	706			
1994-03-28 lttab.dtx v1.0b				
General: Improve documentation . .	645			
1994-03-28 ltthm.dtx v1.0a				
General: Initial version, split from latex.dtx	702			

1994-04-12 ltplain.dtx v1.0d		<code>\fontsubfuzz</code> : Changed dimen to macro	443
General: Define <code>\@acci</code>	31		
1994-04-12 ltvers.dtx v1.0b		<code>\subst@size</code> : <code>\font@submax</code> and <code>\fontsubfuzz</code> now macros	444
General: Have version info generated automatically.	36		
1994-04-14 ltfontcmd.dtx v3.2b		1994-04-19 ltpage.dtx v1.0b	
General: Macros renamed to non-private forms, JB	530	General: Improve documentation	750
<code>\DeclareOldFontCommand</code> : Renamed from <code>\@newfontswitch</code>	537	1994-04-20 ltfontcmd.dtx v3.3a	
1994-04-15 ltboxes.dtx v1.0d		General: Documentation up-dated	530
<code>\@isavebox</code> : Added missing percent character.	634	New implementation of <code>\nocorr</code>	530
1994-04-17 ltcounts.dtx v1.0e		<code>\check@nocorr@</code> : Macros added	534
<code>\@newctr</code> : Use <code>\@nocounterr</code> instead of <code>\@nocnterr</code>	367	<code>\maybe@ic@</code> : <code>\nocorr</code> etc removed from list of tokens to check, leaving only punctuation characters	535
<code>\addtocounter</code> : Use <code>\@nocounterr</code> instead of <code>\@nocnterr</code>	367	1994-04-20 ltmiscen.dtx v1.0e	
<code>\setcounter</code> : Use <code>\@nocounterr</code> instead of <code>\@nocnterr</code>	367	<code>\@enddocument@kernel@warnings</code> : Changed logic for producing warning messages	580
1994-04-17 lterror.dtx v1.0h		1994-04-21 ltboxes.dtx v1.0e	
<code>\@nocounterr</code> : New name for error message, old error message (without arg) kept	258	<code>\@iiiminipage</code> : Extra <code>\bgroup</code> for color	640
1994-04-17 ltthm.dtx v1.0c		<code>\@mpfootnotetext</code> : Extra <code>\endgraf</code> for color	641
<code>\@othm</code> : Use new std counter error message (FMi)	704	<code>\endminipage</code> : Extra <code>\egroup</code> for color	640
1994-04-18 ltfinal.dtx v0.1b		1994-04-21 ltfinal.dtx v0.1c	
General: Initialise <code>\textheight</code> , <code>\textwidth</code> and page style	930	General: Added comments, set the catcodes of 128–255.	927
1994-04-18 ltfloat.dtx v1.0d		1994-04-22 ltssini.dtx v2.1g	
<code>\@footnotetext</code> : (DPC) Remove Colour support	738	<code>\not@math@alphabet</code> : Message changed again	499
<code>\@savemarbox</code> : (DPC) Remove Colour support	732	1994-04-23 ltfinal.dtx v0.1d	
1994-04-18 ltssbas.dtx v2.1i		General: Check that <code>\font@submax</code> is still zero	927
General: Macro <code>\no@alphabet@help</code> removed again	376	1994-04-24 ltoutput.dtx v1.0m	
<code>\calculate@math@sizes</code> : Changed message to log only	399	<code>\@resetfyps</code> : Number 2 changed to <code>\tw@</code>	914
<code>\no@alphabet@error</code> : Use std LaTeX error macro	376	Warning changed	914
1994-04-18 ltssdcl.dtx ???		<code>\@specialoutput</code> : Message changed to give more info and ‘top’ removed	870
<code>\DeclareMathAlphabet</code> : Pass correct arg (2 not 3)	466	<code>\@topnewpage</code> : Message changed to give more info	869
1994-04-18 ltssdcl.dtx v2.1d		Warning message removed as it will be generated later	868
General: Removed surplus <code>\no@alphabet@error</code> (see fam.dtx)	454	General: Changed <code>\@normalsize</code> to <code>\normalsize</code> .	852
1994-04-18 ltssstrc.dtx v2.3d		Corrected unverbbed commands in documentation.	852
General: Changed to new error/warning scheme	424	Removed some long lines and other aesthetic changes.	852
<code>\font@submax</code> : Changed dimen to macro	443	Warning messages changed/corrected.	852
		1994-04-24 ltpictur.dtx v0.1b	
		General: Removed surplus spaces after <code>\hbox to</code> in several cases	670

1994-04-25 ltclass.dtx v0.3h	Warning changed to info message in
General: Removed spurious extra 's at	<code>\@protecteddef</code> 530
the end of error messages 758	1994-04-30 ltoutput.dtx v1.0n
1994-04-25 ltfloat.dtx v1.0e	<code>\@activechar@info</code> :
<code>\@largefloatcheck</code> : Changed warning	<code>\@activechar@warning</code> changed to
message to give more info 728	<code>\@activechar@info</code> 879
Command added 728	<code>\@combinedblfloats</code> : Removed rule in
General: Changed warning messages 719	topnewpage case 884
Removed obsolete tracing code .. 719	<code>\@emptycol</code> : Empty column action
1994-04-27 ltfstrc.dtx v2.3e	added: <code>\@emptycol</code> 867
General: Corrected item that was	<code>\@flsetnum</code> : Rogue space removed . 914
forgotten in last change. 424	<code>\@specialoutput</code> : Cut-off point
1994-04-28 lterror.dtx v1.0j	changed to <code>2\baselineskip</code> 870
<code>\@inmatherr</code> : Macro added 260	Empty column action added:
1994-04-28 lterror.dtx v1.1c	<code>\@emptycol</code> 870
<code>\@inmatherr</code> : Replaced <code>\noexpand</code>	Extra empty column added for
with <code>\protect</code> 260	twocolumn case 870
1994-04-28 ltssdcl.dtx v2.1e	Extra empty column added for
General: Removed all <code>\uppercase</code> in	twocolumn case (wrong, see
hex num parsing macros 454	below) 870
1994-04-28 ltlists.dtx v1.0c	<code>\@topnewpage</code> : Added setting of
<code>\item</code> : Replaced <code>\@ltxnomath</code> by	<code>\col@number</code> 867, 868
<code>\@inmatherr</code> 624	Cut-off point changed to
1994-04-28 ltpictur.dtx v0.1c	<code>3\baselineskip</code> 869
<code>\@multiput</code> : (DPC) Macro added .. 674	Empty column action added:
General: bezier curves added 697	<code>\@emptycol</code> 869
<code>\multiput</code> : (DPC) Ignore spaces	Message changed for Frank 869
between)(..... 673	General: <code>\@activechar@warning</code>
<code>\picture</code> : (DPC) Ignore spaces before	changed to an info message. 852
(..... 672	Added <code>\col@number</code> 852
1994-04-28 ltplain.dtx v1.0g	Documentation tidied. 852
General: Turn off overfull box tracing	Empty column action added. 852
in log 25	Fixed bug from <code>\dblfigrule</code> with
1994-04-29 ltclass.dtx v1.0a	<code>\@topnewpage</code> 852
General: Change version number to 1	Full of floats action improved. 852
(no other change) 758	<code>\col@number</code> : Added <code>\col@number</code> . 864
1994-04-29 ltmiscen.dtx v1.0f	<code>\onecolumn</code> : Added setting of
<code>\@verbatim</code> : <code>\leavevmode</code> added .. 593	<code>\col@number</code> 866
Change to <code>\everypar</code> added 593	1994-05-01 lterror.dtx v1.0k
1994-04-29 ltoutenc.dtx 1.4a	<code>\@latexerr</code> : (CAR) Added draft
General: Removed	<code>\@latexinfo</code> 257
<code>\EncodingSpecific</code> . Renamed all	1994-05-01 ltoutenc.dtx 1.4a
the commands. Added	General: Added the <code>\a</code> command. .. 332
<code>\DeclareTextGlyph</code> and	Added the <code>\SaveAtCatcode</code> and
<code>\UndeclareTextCommand</code> 324	<code>\RestoreAtCatcode</code> commands. . 336
Removed Rokicki's OT1 variant	Removed the uc/lc table settings,
encoding. Moved the driver to the	since the T1 uc/lc table is now the
top. 323	default. 344
1994-04-30 ltfntcmd.dtx v3.3b	Rewrote for the new syntax. . 336, 338
General: Documentation up-dated and	1994-05-01 ltoutenc.dtx v1.4a
tidied 530	General: Removed Rokicki's
Prefix frag@ changed to frag in	encoding. 320
<code>\@protecteddef</code> 530	Renamed the commands, removed
Title changed 530	the <code>\EncodingSpecific</code> command.

Turned all slots into decimal.		1994-05-03 ltmiscen.dtx v1.0h	
Added <code>\a</code>	320	<code>\@centercr</code> : <code>\@badcrerr</code> replaced by	
1994-05-02 ltcntrl.dtx v1.0l		<code>\@nolnerr</code>	589
<code>\@break@tfor</code> : Macro added (from		1994-05-03 lttab.dtx v1.0d	
ltfiles.dtx)	251	<code>\@endpbox</code> : Use <code>\@finalstrut</code> based	
1994-05-02 ltdefs.dtx v1.1f		on depth of <code>\@arstrutbox</code>	668
<code>\renewcommand</code> : Removed surplus		1994-05-04 ltclass.dtx v1.0b	
<code>\space</code> in error	79	<code>\NeedsTeXFormat</code> : Changed wording of	
<code>\renewenvironment</code> : Removed surplus		the warning	775
<code>\space</code> in error	80	1994-05-04 lterror.dtx v1.0m	
1994-05-02 ltfiles.dtx v1.0f		<code>\@badcrerr</code> : Error message removed	260
<code>\@iffilenamepath</code> : <code>\@break@loop</code>		1994-05-05 ltbibl.dtx v1.0c	
renamed to <code>\@break@tfor</code>	314	<code>\@citex</code> : Set switch for warning and	
<code>\@obsoletefile</code> : Make		end of run.	746
<code>\@onlypreamble</code>	318	<code>\nocite</code> : Do not write page number in	
1994-05-02 ltfinal.dtx v0.1e		<code>\nocite</code> warning message.	747
General: Added setting the ‘letter’		Set switch for warning and end of	
catcodes.	939	run.	747
Added setting the ‘other’ catcodes.	939	1994-05-05 ltfinal.dtx v0.1g	
Added setting the special catcodes.	939	General: Added empty errhelp.	927
Made slot 127 illegal	939	<code>\errhelp</code> : Set error help empty.	942
Set all the catcodes	927	1994-05-05 ltfntcmd.dtx v3.3c	
1994-05-02 ltfinal.dtx v0.1f		<code>\@math@egroup</code> : Corrected	
General: Set the catcode of control-J.	939	<code>\@fontswitch</code> and added saved	
1994-05-02 ltmiscen.dtx v1.0g		versions	537
General: Changed 91 to 1991 and		General: Corrected <code>\@fontswitch</code>	530
moved some bits	577	1994-05-05 ltmiscen.dtx v1.0i	
1994-05-02 ltoutput.dtx v1.0o		General: Removed braces from	
<code>\@resetfhps</code> : Code shortened	914	<code>ifnextchar</code> and <code>ifstar</code> arguments	577
General: Code of <code>\@resetfhps</code>		1994-05-07 lttab.dtx v1.0c	
shortened.	852	<code>\@maxtab</code> : Changed <code>\@firsttab</code> to	
1994-05-03 ltbibl.dtx v1.0b		<code>\chardef</code>	649
<code>\nocite</code> : Make <code>\nocite</code> issue a		Changed <code>\@maxtab</code> to <code>\chardef</code>	649
warning for an undefined citation		General: Removed definition of <code>\+</code>	645
key.	747	Removed surplus braces from	
1994-05-03 ltfinal.dtx v0.1f		<code>\@ifnextchar</code> constructs	645
General: Set the catcode of control-J		1994-05-08 ltfntcmd.dtx v3.3d	
to be ‘other’, for use in messages.	927	General: Removed	
1994-05-03 ltfloat.dtx v1.0f		<code>\@undefinedfonterror</code>	530
General: (CAR) Added		<code>\normalsize</code> : Removed	
<code>\@largefloatcheck</code>	719	<code>\@undefinedfonterror</code>	538
Removed unnecessary braces from		1994-05-09 ltfntcmd.dtx v3.3f	
arguments of <code>\@ifnextchar</code>	719	General: Replaced all <code>\next</code> by	
<code>\end@dblfloat</code> : <code>\@largefloatcheck</code>		<code>\@let@token</code> and undo change	
added	727	3.3e, whatever that was.	530
<code>\end@float</code> : (CAR) Added		1994-05-10 ltdefs.dtx v1.0n	
<code>\@largefloatcheck</code>	726	General: (ASAJ) Added	
1994-05-03 ltfssdcl.dtx v2.1f		<code>\DeclareProtectedCommand</code>	73
General: Renamed		Added <code>\DeclareProtectedCommand</code>	82
<code>\@DeclareMathDelimiter</code> to		Removed braces around	
<code>\@DeclareMathDelimiter</code>	454	<code>\@ifundefined</code> argument. ASAJ.	79
1994-05-03 ltlists.dtx v1.0d		<code>\makeatother</code> : Added <code>\makeatletter</code>	
<code>\@item</code> : <code>\hskip</code> changed to <code>\kern</code>	625	and <code>\makeatother</code> ASAJ.	100
<code>\item</code> : Removed superfluous braces	624		

1994-05-10 lterror.dtx v1.0n		
\@latexerr: (ASAJ) Added extra blank lines to \@latexerr.	257	
1994-05-10 ltmiscen.dtx v1.0j		
\@sverb: Slight change in error message text.	595	
1994-05-11 ltboxes.dtx v1.0f		
\@begin@tempboxa: Use new		
\color@setgroup concept.	631	
\@iiiminipage: Use new		
\color@setgroup concept.	640	
\@mpfootnotetext: Use new		
\color@setgroup concept.	641	
Use new \normalcolor and		
\@finalstrut.	641	
General: Superfluous braces removed from several commands	630	
\color@setgroup: macro added for color support	633	
\endminipage: Use new		
\color@setgroup concept.	640	
1994-05-11 ltclass.dtx v1.0c		
\endfilecontents: Add checks for form feed and tab	786	
1994-05-11 ltdirchk.dtx v1.0e		
General: Add \ProvidesFile as used in fd files.	4	
1994-05-11 lterror.dtx v1.0o		
\@latexerr: (ASAJ) Removed one of the extra blank lines to		
\@latexerr.	257	
1994-05-11 ltlogos.dtx v1.0o		
\LaTeX: Use		
\DeclareProtectedCommand.		
ASAJ.	298	
\LaTeXe: Use		
\DeclareProtectedCommand.		
ASAJ.	298	
1994-05-11 ltoutenc.dtx 1.5a		
General: Made T1 and OT1 generate packages rather than def files.		
Renamed the ‘package’ module to ‘teststy’.	323	
1994-05-11 ltoutenc.dtx v1.5a		
General: Reimplemented		
\DeclareTextCommand using		
\@changed@cmd and		
\DeclareProtectedCommand. . . .	324	
Renamed the commands again.		
Made the encoding part of the command syntax. Added the		
\DeclareTextCommand interface.		
Used		
\DeclareProtectedCommand. . . .	320	
\DeclareTextAccent: Reimplemented using \DeclareTextCommand. . . .	326	
1994-05-11 ltspace.dtx v1.0o		
\hspace: Use		
\DeclareRobustCommand. ASAJ. . .	295	
1994-05-12 ltboxes.dtx v1.0g		
\@finalstrut: macro added	643	
\fbbox: New definition, merged with		
\framebox	635	
\framebox: Merged \fbbox and		
\framebox	636	
\normalcolor: macro added for color support	633	
1994-05-12 ltdefns.dtx v1.0p		
General: (ASAJ) Fixed a bug with \relax which was using \@gobble before defining it.	73	
Fixed a bug with \relax which was using \@gobble before defining it.	82	
1994-05-12 ltffsbas.dtx v2.1j		
General: New baselinestretch concept	376	
Replaced hand-protected commands by \DeclareRobustCommand defs	376	
\@linespread: New macro	386	
\fontencoding: Use		
\DeclareRobustCommand.	384	
\fontfamily: Use		
\DeclareRobustCommand.	385	
\fontseries: Use		
\DeclareRobustCommand.	385	
\fontshape: Use		
\DeclareRobustCommand.	385	
\fontsize: Redefined to use		
\set@fontsize	386	
\linespread: New macro	386	
\mathversion: Use		
\DeclareRobustCommand.	387	
1994-05-12 ltffsdcl.dtx v2.1g		
General: Allow \relax as undefined command	454	
Allow \relax’ed cmds to be declared	454	
1994-05-12 ltffssini.dtx v2.1i		
General: Moved \fontencoding to fam.dtx	481	
Moved \fontfamily to fam.dtx . . .	481	
Moved \fontseries to fam.dtx . . .	481	
Moved \fontshape to fam.dtx . . .	481	
Moved \fontsize to fam.dtx . . .	481	
Moved \mathversion to fam.dtx . .	481	
Moved \selectfont to tracefnt.dtx	481	

1994-05-12 ltfsstrc.dtx v2.3f	1994-05-13 ltfinal.dtx v1.0h
\selectfont: Use	General: Added output enc stuff . . . 941
\DeclareRobustCommand 428	1994-05-13 ltfloat.dtx v1.0g
1994-05-12 ltoutenc.dtx 1.5a	\@footnotetext: (DPC) Add new
General: Removed the	style colour support:
\SaveAtCatcode and	\normalcolor 738
\RestoreAtCatcode commands. . 336	(DPC) Use \@finalstrut 738
Rewrote for the new syntax. . 336, 338	\@xfloat: (DPC) Use \normalcolor 724
1994-05-12 ltoutput.dtx v1.0p	1994-05-13 ltfntcmd.dtx v3.3g
\@writsetup: \normalcoloradded 879	General: Replaced \@protecteddef by
General: \normalcoloradded in	\DeclareRobustCommand 530
various places (DPC). 852	1994-05-13 ltfssbas.dtx v2.1k
1994-05-13 ltboxes.dtx v1.0h	General: Remove File identification
\@arrayparboxrestore: New accent	‘typeout’ 376
system, use \let not \def 639	1994-05-13 ltfssbas.dtx v2.1l
1994-05-13 ltcounts.dtx v1.0f	\DeclareFontEncoding: Init encoding
General: Removed \@lalph 370	change command 380
Removed \@lalph 370	\define@newfont: Use \@input@ for fd
1994-05-13 ltdefns.dtx v1.0q	files 390
General: (ASAJ) Renamed	1994-05-13 ltfssdcl.dtx v2.1h
\DeclareProtectedCommand to	General: Removed file identification
\DeclareRobustCommand.	typeout 454
Removed \@if@short@command. . . 73	1994-05-13 ltfssini.dtx v2.1j
(ASAJ) Replaces \space by ‘ ’ in	General: Removed file identification
\csname. 73	typeout 481
Renamed	1994-05-13 ltfsstrc.dtx v2.3g
\DeclareProtectedCommand to	General: Removed typeouts as
\DeclareRobustCommand.	\ProvidesPackage writes to log. 424
Removed \@if@short@command.	1994-05-13 ltoutenc.dtx v1.5b
Moved to after the definition of	General: Added \[, \} and \\$. . . . 320
\@gobble. 82	Renamed
1994-05-13 ltdefns.dtx v1.0r	\DeclareProtectedCommand to
General: (ASAJ) Added logging	\DeclareRobustCommand. 320
message to	Replaces \space by ‘ ’ in \csname. 320
\DeclareProtectedCommand. . . . 73	1994-05-13 ltpictur.dtx v0.1d
Added logging message to	General: Removed surplus braces from
\DeclareProtectedCommand. . . . 82	\@if.. constructions 670
1994-05-13 ltdefns.dtx v1.0s	1994-05-13 lttab.dtx v1.0d
General: (ASAJ) Added	\@contfield: Colour support 651
\@backslashchar. 73	\@startfield: Colour support 651
(ASAJ) Coded \@ifdefinable more	\@stopfield: Colour support 651
efficiently. 73	\@a: moved to ltoutenc 649
Coded more efficiently, thanks to	1994-05-14 fontdef.dtx v2.1f
FMi. 79	General: Removed .def files. 508
1994-05-13 ltfiles.dtx LaTeX2e	1994-05-14 ltfssbas.dtx v2.1m
\listfiles: Stop \listfiles being	\enc@update: Macro added 385
run twice 318	1994-05-14 ltfssbas.dtx v2.1n
1994-05-13 ltfiles.dtx v1.0g	General: Set defaults for all \f@...
\document: Added execution of	\DeclareErrorFont: Don’t set
\every@size 302	\f@encoding 393
1994-05-13 ltfinal.dtx v0.1h	\DeclareFontEncoding: Log if
General: Added package otlenc, and	encoding is redeclared 380
defined \@acci, \@accii and	Only init enc change cmd when new
\@acciii. 927	encoding 380

1994-05-14 ltffssini.dtx v2.1k	1994-05-16 ltlogos.dtx v1.1a
General: Init error font just before	General: (ASAJ) Split from ltinit.dtx. 298
checking for fontdef.cfg 503	1994-05-16 ltmath.dtx v1.0k
\reset@font: Remove surplus braces 501	\ensuremath: Use
1994-05-14 ltffsstrc.dtx v2.3h	\DeclareRobustCommand and add
\selectfont: Added \enc@update . 430	extra braces in math mode 609
1994-05-14 ltoutenc.dtx 1.5d	1994-05-16 ltoutenc.dtx 1.5h
General: Moved the driver to the top. 323	General: \pounds was still using u
1994-05-14 ltoutenc.dtx v1.5c	rather than ui shape. 336
General: Added the fontenc package 363	1994-05-16 ltoutenc.dtx v1.5f
Added the fontenc package. 320	General: enc files now have uc
Fixed a bug which caused an	encoding name parts (FMi) 320
infinite loop if \f@encoding was	Revert code so that the encoding
incorrectly set. 320, 324	given is used in
Moved fontsmpl to its own dtx file. 320	\DeclareTextCommand (FMi) . . . 320
1994-05-14 ltoutenc.dtx v1.5d	1994-05-16 ltoutenc.dtx v1.5g
General: Rewrote	General: Made fontenc.sty use the new
\DeclareTextCommand to define its	mixed-case encoding files. 320
argument to use the current	Removed the lowercasing of the
encoding by default, rather than	filename. 363
the encoding provided to	1994-05-16 ltoutenc.dtx v1.5h
\DeclareTextCommand. . . . 320, 324	General: Added \NG, \ng, \TH, \th,
Tidied up the documentation. . . . 320	\DH, \dh, \DJ and \dj. 320
1994-05-14 ltoutenc.dtx v1.5e	Added \r (ring accent) and \k
General: Replaced \ENC@cmd by	(ogonek) accents. 320
\ENC-cmd. 320	Fixed a bug with \pounds. 320
1994-05-15 ltffsbas.dtx v2.1o	Removed \P from the OT1
General: encoding cmds changed to	definitions file. 320
enc-cmd 376	1994-05-16 ltoutenc.dtx v1.5i
1994-05-16 fontdef.dtx v2.1g	General: Fixed a bug with \d. 320
General: Removed	1994-05-16 ltoutput.dtx v1.0q
\DeclareFontEncoding for ot1 and	\@writesetup: Changed setting of
t1 and input .def files instead . . 508	accents (FMi): with the new
1994-05-16 ltalloc.dtx v1.1a	encoding setup they can use \let.
General: (ASAJ) Split from ltinit.dtx. 246	It could also use the new internal
1994-05-16 ltcntrl.dtx v1.0a	commands? 880
General: (ASAJ) Split from ltinit.dtx. 248	General: Changed setting of accents
1994-05-16 ltdefns.dtx v1.1a	(FMi). 852
General: (ASAJ) Split from ltinit.dtx. 73	1994-05-16 ltpar.dtx v1.1a
1994-05-16 lterror.dtx v1.1a	General: (ASAJ) Split from ltinit.dtx. 262
General: (ASAJ) Completely new	1994-05-16 ltplain.dtx v1.0h
error interface. 252	General: Comment out encoding
(ASAJ) Split from ltinit.dtx. . . . 252	specific commands 30
1994-05-16 ltfinal.dtx v1.0i	Remove \@acci and friends again . 31
General: moved output enc stuff to	Remove unnecessary def for \item 30
lfonts 941	\loop: Use Kabelschacht method . . . 28
1994-05-16 ltffsbas.dtx v2.1p	\m@th: Remove unnecessary space . . . 29
\fontsize: Pass \baselinestretch not	1994-05-16 ltspac.dtx v1.1a
\f@linespread 386	General: (ASAJ) Split from ltinit.dtx. 279
\linespread: Remove surplus braces 386	1994-05-17 ltclass.dtx v1.0e
1994-05-16 ltffssini.dtx v2.1m	\@use@option: Execute option after
\@acciii: Define saved versions of	removing from list, not before . . 772
accents 504	

1994-05-17 ltdefs.dtx 1.1b	1994-05-19 ltcounts.dtx v1.1a
General: (ASAJ) Added the	General: Extracted file from ltcntlen. 366
<code>\@protect@...</code> commands. 83	1994-05-19 ltdefs.dtx v1.1d
1994-05-17 ltdefs.dtx v1.1b	General: (RmS) Added definitions for
General: (ASAJ) Added definitions for	<code>\@namedef</code> and <code>\@nameuse</code> again. . 73
<code>protect.</code> 73	1994-05-19 ltfinal.dtx v0.1k
(ASAJ) Removed warnings and	General: Removed <code>\makeat...</code> 927
logging to lterror.dtx. 73	1994-05-19 ltidxglo.dtx v1.1a
Added the discussion of protected	General: Initial version of ltidxglo.dtx,
commands, defined the values that	split from ltidxbib.dtx 742
<code>\protect</code> should have. 83	1994-05-19 ltlength.dtx v1.1a
1994-05-17 ltdefs.dtx v1.1c	General: Extract file ltlength from
General: (ASAJ) Redid definitions for	ltcntlen. 374
<code>protect.</code> 73	1994-05-19 ltpageno.dtx v1.1a
1994-05-17 lterror.dtx v1.1b	General: Extract file ltpageno from
General: (ASAJ) Moved error stuff	ltcntlen. 571
from ltdefs.dtx. 252	1994-05-19 ltplain.dtx v0.1k ltfinal
1994-05-17 ltssini.dtx v2.1n	<code>\showoutput</code> : used <code>\maxdimen</code> not
<code>\copyright</code> : Really add extra braces 501	99999 32
<code>\nfss@text</code> : Added braces to allow	<code>\showoverfull</code> : used <code>\@one</code> not 1 31
use in subscripts 501	1994-05-19 ltxref.dtx v1.1a
1994-05-17 ltmath.dtx v1.0i	General: Extract file ltxref from
General: Replaced <code>\let</code> by <code>\gdef</code> , for	ltcntlen. 572
indirect definition. 605	1994-05-20 ltdefs.dtx v1.1e
1994-05-17 ltoutenc.dtx v1.5j	General: Changed command name
General: Added braces to <code>\pounds</code> so	from <code>\@checkcommand</code> to
it works as a subscript. 320	<code>\CheckCommand.</code> 73
1994-05-18 ltdefs.dtx 1.1c	<code>\CheckCommand</code> : Changed name from
General: (ASAJ) Renamed the	<code>\@checkcommand</code> to
commands, and removed one	<code>\CheckCommand.</code> 81
which is no longer needed. 83	1994-05-20 lterror.dtx v1.1c
1994-05-18 ltdefs.dtx v1.1c	General: (ASAJ) Added
General: Redid the discussion and	<code>\@latex@info@no@line.</code> 252
definitions, in line with the	(ASAJ) Added missing full stops. 252
proposed new setting of <code>\protect</code>	(ASAJ) Fixed a bug with
in the output routine. 83	<code>\@inmatherr.</code> 252
1994-05-18 ltfinal.dtx v0.1j	1994-05-20 ltfinal.dtx v0.1l
General: Corrected the lccode for	General: Use new font warning
<code>d-bar.</code> 927	commands 934
1994-05-18 ltlogos.dtx v1.1b	1994-05-20 ltfloat.dtx v1.0h
General: (ASAJ) Added the \TeX	<code>\@endfloatbox</code> : Restore outer value of
logo. 298	<code>@nobreak</code> switch. 728
(ASAJ) Made the $\LaTeX 2_{\epsilon}$ logo use	<code>\outer@nobreak</code> : Macro added:
the text font ‘2’ rather than the	default is to do nothing. 728
math font ‘2’. 298	1994-05-20 ltfntcmd.dtx v3.3h
1994-05-18 ltoutenc.dtx v1.5k	General: Use new error commands . 530
General: Made dotted-i produce ‘i’. . 320	1994-05-20 ltfsbas.dtx v2.1q
Removed braces from <code>\pounds</code> and	General: Use new error commands . 376
<code>\dollar.</code> 320	1994-05-20 ltfsstrc.dtx v2.3i
Replaced <code>\defaultencoding</code> with	General: Use new error command
<code>\encodingdefault.</code> 320	names 424
1994-05-19 ltbibl.dtx v1.1a	1994-05-20 ltmiscen.dtx v1.0l
General: Initial version of ltbibl.dtx,	<code>\@writefile</code> : Added correct setting of
split from ltidxbib.dtx 745	<code>\protect.</code> 583

1994-05-20 ltmiscen.dtx v1.0m		\GenericError, \GenericWarning	
General: Use new warning commands	577	and \GenericInfo.	252
1994-05-20 ltoutput.dtx v1.0s		(ASAJ) Replaces \string by	
\@writsetup: Added setting of		\protect in some messages. ...	252
\protect during \shipout.	879	1994-05-22 lterror.dtx v1.2d	
General: Added setting of \protect		\GenericError: (DPC) Alternative	
during \shipout.	852	version added for old TeXs	253
1994-05-20 ltpage.dtx v1.0d		(DPC) New version using long	
\markright: Changed setting for		command name.	253
\protect.	751	1994-05-22 ltfloat.dtx v1.0i	
1994-05-20 ltsect.dtx v1.0c		General: Use new warning commands	719
General: Correct setting of \protect.	716	1994-05-22 ltoutput.dtx v1.0t	
\addcontentsline: Correct setting of		General: Changed warnings and infos	
\protect.	715	to new commands.	852
1994-05-21 ltbibl.dtx v1.1b		1994-05-22 ltpictur.dtx v0.1e	
General: Use new warning commands	745	General: Use new warning cmds ...	670
1994-05-21 lterror.dtx v1.1d		1994-05-23 ltclass.dtx v1.0h	
General: (ASAJ) Made the error		\NeedsTeXFormat: Don't stop	
commands robust.	252	completely when format is wrong	775
1994-05-21 ltfiles.dtx v1.0h		\usepackage: Remove argument if	
General: Use new error commands .	299	possible	775
1994-05-21 ltlists.dtx v1.0f		1994-05-23 ltdirchk.dtx v1.0f	
General: Use new error commands .	613	General: Document \TeXversion ...	1
1994-05-21 ltmiscen.dtx v1.0n		1994-05-23 ltfssstr.dtx v2.3j	
General: Use new error commands .	577	General: Removed def of	
1994-05-21 ltsect.dtx v1.0d		\f@warn@break	442
General: Use new error commands .	706	1994-05-23 ltoutput.dtx v1.0u	
1994-05-21 lttab.dtx v1.0f		\@activechar@info: Added	
General: Use new error commands .	645	\MessageBreak	879
1994-05-21 ltxref.dtx v1.1b		\@writsetup: Changed resetting of	
General: Use new warning commands	572	\protect after shipout to use	
\newlabel: Use new warning		\aftergroup	879
commands	574	General: Added \MessageBreak. ...	852
1994-05-22 ltclass.dtx v1.0f		Changed resetting of \protect after	
General: Use new warning and error		shipout.	852
commands	754	1994-05-24 lterror.dtx v1.2e	
1994-05-22 ltdefs.dtx v1.1f		\@latex@info@no@oline: Macro added	256
General: Use new warning and error		1994-05-24 lterror.dtx v1.2f	
cmds	73	General: (DPC) wrap long lines ...	252
1994-05-22 lterror.dtx v1.1e		1994-05-24 ltfntcmd.dtx v3.3i	
General: (ASAJ) Replaced bgroup by		General: Tidying and typos fixed ..	530
begingroup in error messages, to		1994-05-24 ltmiscen.dtx v1.0q	
stop extra mathords creeping into		\@currentvline: Use \@empty as outer	
math mode.	252	default	588
1994-05-22 lterror.dtx v1.2a		1994-05-25 ltdirchk.dtx v1.0g	
General: (ASAJ) Made		\filename@parse: Mac parser had "	
\GenericError, \GenericWarning		typo for :	12
and \GenericInfo robust.	252	1994-05-25 ltfntcmd.dtx v3.3j	
(ASAJ) Replaced \ and tilde by		General: Insertion of \aftergroups to	
\MessageBreak and \space.	252	implement \nocorr moved to the	
(ASAJ) Replaced		end of the group	530
\@generic@message and		\check@icr: Macros added	534
\@generic@error by			

<code>\check@nocorr@</code> : Insertion of	1994-06-08 ltfinal.dtx v1.0m
<code>\aftergroups</code> moved and defaults	General: Add patch file system 941
set up for efficiency 534	1994-06-09 ltfinal.dtx v1.0n
<code>\DeclareTextFontCommand</code> :	General: For \TeX 2, do not set codes
<code>\expandafter</code> inserted 532	for higher half of character
Insertion of <code>\aftergroups</code> moved 532	table. 932, 939
1994-05-25 ltoutput.dtx v1.0v	1994-06-09 ltfntcmd.dtx v3.3k
General: Extra documentation. 852	General: Tidying and typos fixed in
1994-05-25 ltsect.dtx v1.0e	documentation 530
<code>\@dottedtocline</code> : Put braces around	1994-06-18 ltfntcmd.dtx v3.3l
argument 4 (the actual toc entry)	General: Added check for empty text . 530
to avoid font (and possibly other)	<code>\check@nocorr@</code> : Added check for
changes leaking out to the leaders. 717	empty text 534
1994-05-25 ltthm.dtx v1.0c	1994-06-22 ltfntcmd.dtx v3.3m
General: Modify documentation 702	General: Removed space from
1994-05-25 ltvers.dtx v1.0d	<code>\nfss@text</code> 530
General: Remove PRELIMINARY	Renamed <code>\check@nocorr</code> 530
TEST RELEASE from startup	<code>\check@nocorr@</code> : Renamed
banner (spring is here) 36	<code>\check@nocorr</code> to <code>\text@command</code>
1994-05-25 ltxref.dtx v1.1c	to improve <code>\long</code> error message . 534
General: Modify documentation 572	<code>\DeclareTextFontCommand</code> : Removed
1994-05-26 ltfiles.dtx LaTeX2e	space from <code>\nfss@text</code> 532
<code>\@missingfileerror</code> : Modify message	1994-06-22 ltmath.dtx v1.2t classes
format 316	<code>\mathindent</code> : Set <code>\mathindent</code> at the
1994-05-26 ltlogos.dtx v1.1c	end of the class instead of at begin
General: Remove <code>\SLiTeX</code> logo 298	document 610
1994-05-26 ltmiscen.dtx v1.0r	1994-07-20 ltlogos.dtx v1.1e
General: <code>\literal</code> removed 597	<code>\LaTeX</code> : Save a few tokens 298
1994-05-26 ltplain.dtx v1.1m	<code>\LaTeXe</code> : Save a few tokens 298
<code>\iterate</code> : (CAR) added <code>\long</code> 28	1994-07-20 ltpage.dtx v1.0h
<code>\underbar</code> : (CAR/FMi) changed to	<code>\sloppy</code> : Save a few tokens 752
use box <code>\tw@</code> 30	1994-09-16 ltssbas.dtx v2.1s
1994-05-26 ltplain.dtx v1.1p	<code>\nfss@catcodes</code> : Reset [and] as well,
<code>\underbar</code> : (DPC) changed to use	just in case 391
<code>\sbox</code> 30	1994-10-07 ltoutenc.dtx v1.5l
1994-05-29 ltssdcl.dtx v2.1j	General: Moved the ogonek accent. . 320
General: Use new error commands . . 454	1994-10-11 ltdirchk.dtx v1.0h
1994-05-31 ltfinal.dtx v1.0n	<code>\@TeXversion</code> : Check for TeX3.14 . . 13
General: Renamed <code>lthyphen.*</code> to	General: Modify all of <code>ltxcheck</code> again . 13
<code>lthyphen.*</code> 927	1994-10-12 ltsect.dtx v1.0f
1994-06-01 ltboxes.dtx v1.0i	General: Doc. typos 706
<code>\@frameb@x</code> : Macro added. 636	1994-10-14 fontdef.dtx v2.2a
<code>\@ifframebox</code> : New version, so <code>\width</code>	General: New coding 506
is correct in <code>\framebox</code> 636	1994-10-14 ltssini.dtx v2.2a
<code>\fbox</code> : New version, using	General: New coding for cfg files . . 481
<code>\@frameb@x</code> 635	1994-10-14 ltmiscen.dtx v1.0s
<code>\framebox</code> : New version, so <code>\width</code> is	General: Move math to other file . . 577
correct in <code>\framebox</code> 636	1994-10-14 ltplain.dtx v1.1a
1994-06-01 ltlogos.dtx v1.1d	General: Moved code to other files. . . 14
<code>\LaTeX</code> : Add <code>\m@th</code> to force math size	1994-10-15 ltssbas.dtx v2.1t
calculations 298	<code>\extract@alph@from@version</code> : Warn
1994-06-01 ltoutput.dtx v1.0w	if math alpha is used outside
General: Tidied up typesetting. 852	math 398

- 1994-10-18 ltboxes.dtx v1.0j
`\@frameb@x`: `\leavevmode` added . . . 636
`\@ifframebox`: `\leavevmode` moved to
`\@frameb@x` 636
`\@parboxto`: Macro added to remove
misuse of `\@empty` 638
General: stuff from ltpatch done . . . 630
`\fbox`: `\long` added 635
`\mbox`: `\long` added 631
`\sbox`: `\long` added 634
- 1994-10-18 ltclass.dtx v1.0j
General: Move `\listfiles` to
ltfiles.dtx 754
- 1994-10-18 ltdefs.dtx v1.2a
`\@star@or@long`: macro added 76
General: Add extra test for `\endgraf` 73
Add star-forms for all commands . . 73
`\renew@environment`: reset end
command 80
- 1994-10-18 ltfiles.dtx v1.0i
`\listfiles`: code moved here from
ltclass 318
- 1994-10-18 ltoutenc.dtx v1.5l
General: Added new definitions of
`\patterns` and `\hyphenation`. . . 332
- 1994-10-18 ltoutenc.dtx v1.5m
General: Added new definitions of
`\patterns` and `\hyphenation`. . . 320
- 1994-10-18 ltsect.dtx v1.0g
`\@dottedtocline`: Added
`\normalcolor` for page number . . 717
General: Added `\normalcolor` 706
- 1994-10-19 ltfssbas.dtx v2.1t
`\DeclareFontEncoding`: Add missing
`\relax`. 380
- 1994-10-23 ltfssstrc.dtx v23.k
`\every@math@size`: Renamed to
`\every@math@size` 433
- 1994-10-23 ltmath.dtx v1.0l
`\@eqnnum`: Added `\normalcolor` since
`\eqno` introduces a subgroup of the
displayed math group 607
`\ensuremath`: Remove extra braces:
but see p 168 of Leslie's book . . 609
- 1994-10-24 ltboxes.dtx v1.0k
`\fbox`: Inner braces added (to fix
latex/1061) 635
- 1994-10-25 fontdef.dtx v2.2c
General: Added OMSenc.def 508
- 1994-10-25 ltboxes.dtx v1.0l
`\@isavepicbox`: missing percent
(moved from ltpatch) 634
- 1994-10-25 ltdefs.dtx v1.2b
General: Documentation
improvements 73
- 1994-10-25 ltoutenc.dtx 1.6a
General: Added `\textdollar`,
`\textlbrace`, `\textrbrace`,
`\textsterling`, `\textunderline`. 338
Removed `\textlbrace`,
`\textrbrace`, `\textunderline` to
give them their proper names. . . 338
- 1994-10-25 ltoutenc.dtx v1.6a
General: Added
`\ProvideTextCommand`,
`\UseTextSymbol`, `\UseTextAccent`,
`\DeclareTextSymbolDefault`,
`\DeclareTextAccentDefault`,
`\DeclareTextCommandDefault`,
and
`\ProvideTextCommandDefault`. . . 320
Added the `\Provide` commands,
and the default definitions. . . . 324
Added the defaults. 332
Added the files OTlenc.def,
Tlenc.def and OMSenc.def. . . . 332
Added the OMS encoding. 344
- 1994-10-27 ltoutenc.dtx 1.6b
General: Added `\textasciicircum`
`\textasciitilde` `\textbackslash`
`\textbar` `\textbraceleft`
`\textbraceright`
`\textcompwordmark` `\textemdash`
`\textendash` `\textexclamdown`
`\textgreater` `\texthyphenchar`
`\texthyphen` `\textless`
`\textquestiondown`
`\textquotedblleft`
`\textquotedblright`
`\textquotedbl` `\textquoteleft`
`\textquoteright`
`\textunderscore`
`\textvisiblespace` 338
Added: `\textemdash` `\textendash`
`\textexclamdown`
`\texthyphenchar` `\texthyphen`
`\textquestiondown`
`\textquotedblleft`
`\textquotedblright`
`\textquoteleft`
`\textquoteright` 336
- 1994-10-27 ltoutenc.dtx v1.5d
General: Rewrote
`\DeclareTextSymbol` to define its
argument to use the current

encoding by default, to fit with <code>\DeclareTextCommand</code>	324	Rewrote <code>\copyright</code> to use <code>\textcircled</code>	334
1994-10-27 ltoutenc.dtx v1.6b		1994-10-31 fontdef.dtx v2.2d	
General: Added <code>\textbackslash</code> . . .	344	General: Added OMLenc.def	508
Added more defaults for OT1. . .	332	1994-10-31 fontdef.dtx v2.2e	
Removed the enc.def files	320	General: ... and moved further down	508
Removed the files OT1enc.def, T1enc.def and OMSenc.def.	332	1994-10-31 ltfloat.dtx v1.1a	
Renamed <code>\textlbrace</code> to <code>\textbraceleft</code> and <code>\textrbrace</code> to <code>\textbraceright</code>	344	<code>\dblfloat</code> : Major changes since two-column and one-column cases merged	722
1994-10-29 ltmath.dtx 1.0m		<code>\dblfloatset</code> : Macro added	722
General: ASAJ: Added <code>\DeclareMathOperator</code>	598	Major changes to parameter parsing, setting of local variables, etc; two-column and one-column cases merged; space hacks moved	722
ASAJ: Tidied up documentation.	605	<code>\endfloatbox</code> : (DPC/CAR) Extra box added to remove colour resetting from vmode	728
1994-10-29 ltmath.dtx v1.0m		<code>\floatboxreset</code> : Macro added . . .	726
General: ASAJ: Added <code>\mathellipsis</code> , <code>\mathdollar</code> and <code>\mathsterling</code>	605	<code>\footnotetext</code> : (DPC/CAR) Move colour setting to output routine .	738
ASAJ: Removed <code>\dag</code> , <code>\ddag</code>	605	<code>\savemarbox</code> : (DPC/CAR) Extra box added for colour	732
ASAJ: Renamed <code>\S</code> and <code>\P</code> to <code>\mathsection</code> and <code>\mathparagraph</code> and made them <code>\mathchardefs</code>	605	<code>\setfps</code> : Macro added	723
1994-10-29 ltoutenc.dtx v1.6c		<code>\xdblfloat</code> : Macros removed: <code>\dbflt</code> , <code>\xdblfloat</code>	728
General: Added commands like <code>\dots</code> for use in text and math.	332	<code>\xfloat</code> : (DPC/CAR) Extra box added to remove colour resetting from vmode	724
Renamed <code>\P</code> , <code>\S</code> , <code>\dag</code> and <code>\ddag</code> to <code>\textparagraph</code> , <code>\textsection</code> , <code>\textdagger</code> and <code>\textdaggerdbl</code>	320	Major changes, removing setting of local variables, space hacks etc; two-column and one-column cases merged	723
1994-10-30 ltdefs.dtx v1.2c		Reset hook added	724
<code>\@onelevel@sanitize</code> : Macro added .	99	<code>\xympar</code> : (DPC/CAR) Extra box added since needed for floats . . .	733
General: (CAR) <code>\@onelevel@sanitize</code> added	73	<code>\fps@dbl</code> : Macro added	723
1994-10-30 ltdefs.dtx v1.2f		1994-10-31 ltoutput.dtx v1.1a	
General: (DPC) <code>\newwrite</code> 's moved to ltfiles	73	<code>\makecol</code> : (DPC/CAR) Colour resetting moved to here	876
1994-10-30 ltmath.dtx v1.0n		<code>\topnewpage</code> : (DPC/CAR) Extra box added to remove colour resetting from vmode	868
General: ASAJ: Moved the new commands to ltoutenc.	605	(DPC/CAR) Use <code>\color@begingroup</code> for colour . .	868
1994-10-30 ltoutenc.dtx v1.6d		(DPC/CAR) Use <code>\normalcolor</code> .	868
General: Added <code>\DeclareTextCompositeCommand</code>	320	1994-11-02 ltoutenc.dtx v1.6d	
Added <code>\textcircled</code>	320, 334, 344	General: Wrapped lines longer than 70 characters.	320
Added <code>\t</code>	334	1994-11-03 ltclass.dtx v1.0k	
Added math commands.	320	General: Move <code>\@missingfileerror</code> to ltfiles	758
Added OML encoding.	320, 334		
Added the OML encoding.	345		
Made <code>\textless</code> and <code>\textgreater</code> come from OML.	334		
Moved math commands here from ltmath.	336		
Removed <code>\textregistered</code>	334		

1994-11-03 ltdirchk.dtx v1.0i	1994-11-04 ltsect.dtx 1.0h
General: Generate an error if latex.ltx not used with clean initex 1	\@sect: (ASAJ) Added
1994-11-03 ltfiles.dtx v1.0j	\protected@edef. 710
\@missingfileerror: Move here from ltclass 316	General: (ASAJ) Added
1994-11-04 ltboxes.dtx v1.0m	\protected@xdef to \thanks. . . 706
\@mpfootnotetext: Added	1994-11-04 ltsect.dtx v1.0h
\protected@edef. ASAJ. 641	General: Added \protected@write to \addtocontents. ASAJ. 716
1994-11-04 ltdefns.dtx v1.2e	\addcontentsline: Added
General: Added	\protected@write to \addcontentsline. ASAJ. 715
\set@display@protect to \typeout. ASAJ. 73	1994-11-04 lttab.dtx v1.0h
Added commands for setting and restoring \protect. ASAJ. 85	\@mkpream: (ASAJ) Added
Rewrote protected short commands using \x@protect. ASAJ. 83	\@unexpandable@protect to \@mkpream. 664
1994-11-04 lterror.dtx v1.2g	\multicolumn: (ASAJ) added
General: Added	\set@typeset@protect. 660
\set@display@protect to \Generic* commands. ASAJ. . . 252	1994-11-04 ltxref.dtx v1.1d
1994-11-04 ltfiles.dtx v1.0k	\label: (ASAJ) Added
\@nofiles: Added setting of \protected@write, \makeindex and \makeglossary to \@nofiles. ASAJ. 306	\protected@edef 574
\protected@write: Macro added ASAJ. 306	(ASAJ) Added \protected@write 574
1994-11-04 ltfloat.dtx v1.1b	1994-11-05 ltboxes.dtx v1.0n
\@footnotetext: (ASAJ) Added	\@mpfootnotetext: Color resetting for footnotes moved to endminipage: as for main page. 641
\protected@edef. 738	\color@endbox: macro added for color support 633
\footnotemark: Added	\color@hbox: macro added for color support 633
\protected@xdef to \footnotemark. 739	\endminipage: Color resetting for footnotes moved to here: as for main page. 640
1994-11-04 ltidxglo.dtx v1.1b	1994-11-05 ltboxes.dtx v1.0o
\@wrglossary: Added	\@mpfootnotetext: Color groups restored here. 641
\protected@write to \@wrglossary. 744	1994-11-05 ltfloat.dtx v1.1c
\@wrindex: Added \protected@write to \@wrindex. 743	\@dblflset: Add compatibility with old version of \@xfloat. 722
General: Removed \if@filesw from \makeindex. 742	\@endfloatbox: Use new \color@hbox concept. 728
\makeglossary: Removed \if@filesw from \makeglossary. 743	\@footnotetext: Removed
1994-11-04 ltmiscen.dtx v1.0t	\normalcolor (again) 738
\@writefile: Removed setting of \protect. ASAJ. 583	\@savemarbox: Use new \color@hbox concept. 732
1994-11-04 ltoutenc.dtx v1.6f	\@setfps: Add compatibility with old version of \@xfloat. 723
General: Added _ 335	\@xfloat: Add compatibility with old version of \@xfloat: but the arguments, provided at exorbitant cost, are now completely ignored 723
Added \mathunderscore. 336	Use new \color@hbox concept. . . 724
1994-11-04 ltpage.dtx v1.0e	\@xympar: Use new \color@hbox concept. 733
\markright: Added	
\@unexpandable@protect. ASAJ. 751	

1994-11-05 ltoutenc.dtx v1.6g		1994-11-10 ltbibl.dtx v1.1c	
General: Added setting of		General: Fix <code>\nocite{*}</code>	745
<code>\@typeset@protect</code> to <code>\patterns</code>		<code>\nocite</code> : Fix <code>\nocite{*}</code>	747
and <code>\hyphenation</code>	332	1994-11-10 ltmath.dtx v1.2v classes	
1994-11-05 ltoutput.dtx v1.1b		<code>eqnarray</code> : Added value of <code>\parskip</code> to	
<code>\@topnewpage</code> : Use new <code>\color@hbox</code>		<code>\abovedisplayskip</code> to compensate	
concept.	868	for negative <code>\topsep</code>	612
<code>\@writsetup</code> : Change protect		1994-11-10 ltoutput.dtx v1.1e	
settings for new-style, protect-free		<code>\@writsetup</code> : Modify <code>\protect</code>	
aux-files.	879	setting	879
Use new <code>\color@hbox</code> concept. . .	879	1994-11-10 ltplain.dtx v1.1b	
1994-11-05 ltoutput.dtx v1.1c		General: (CAR) added patch to <code>\loop</code> . .	14
<code>\@begindvi</code> : Added macro	883	<code>\iterate</code> : (CAR) added extra <code>\relax</code> .	28
<code>\@begindvibox</code> : Added macro	864	1994-11-11 ltspace.dtx v1.2a	
<code>\@writsetup</code> : Add new <code>\AtBeginDvi</code>		<code>\:</code> (DPC) Make robust	283
concept	879	1994-11-12 ltfntcmd.dtx v3.3o	
<code>\AtBeginDvi</code> : Added macro	864	<code>\normalsize</code> : Added <code>\MessageBreak</code> .	538
1994-11-06 ltfssbas.dtx v2.1u		1994-11-12 ltlists.dtx v1.2b ltspace	
<code>\cf@encoding</code> : New macro	386	<code>\endtrivlist</code> : Changed order of tests	
<code>\DeclareFixedFont</code> : Renamed		to make <code>\@noitemerror</code> correct:	
<code>\every@size</code> to		end of an era.	622
<code>\every@math@size</code>	378	1994-11-12 ltmiscen.dtx v1.0u	
1994-11-06 ltfssini.dtx v2.2b		<code>center</code> : Changed end macro to <code>\def</code> :	
<code>\@setsize</code> : Use <code>\@typeset@protect</code> .	500	safer and consistent	590
1994-11-06 ltfssstrc.dtx v2.3k		<code>flushleft</code> : Changed end macro to	
<code>\glb@currsz</code> : New implementation .	432	<code>\def</code> : safer and consistent	591
<code>\try@simples</code> : New implementation .	443	<code>flushright</code> : Changed end macro to	
<code>\try@size@substitution</code> : New		<code>\def</code> : safer and consistent	591
implementation	443	1994-11-12 ltplain.dtx v1.1c	
<code>\tryis@simple</code> : New implementation .	443	General: Comment out more encoding	
1994-11-07 fontdef.dtx v2.2f		specific commands	30
General: (DPC) Add		1994-11-12 ltspace.dtx v1.2b	
<code>\DeclareMathSizes</code> declarations .	513	<code>\addpenalty</code> : Corrected error	
(DPC) Updated to use		message	291
<code>\ProvidesFile</code>	508	<code>\addvspace</code> : Corrected error message .	290
1994-11-07 ltfiles.dtx v1.0m		1994-11-13 ltspace.dtx v1.2c	
<code>\document</code> : Renamed <code>\every@size</code> to		<code>\addpenalty</code> : Recorrected error	
<code>\every@math@size</code>	302	message	291
1994-11-07 ltplain.dtx v1.0l		<code>\addvspace</code> : Recorrected error	
<code>\@unused</code> : move here from ltdefs,		message	290
remove duplicate <code>\@mainaux</code>	23	1994-11-14 ltoutput.dtx v1.1f	
1994-11-07 preload.dtx v2.1e		<code>\@begindvi</code> : Use normal box register:	
General: (DPC) Updated to use		why a box?	883
<code>\ProvidesFile</code>	527	<code>\@begindvibox</code> : Use normal box	
1994-11-09 ltboxes.dtx v1.0p		register: why a box?	864
<code>\@finalstrut</code> : Revert <code>\finalstrut</code> to		<code>\@writsetup</code> : Modify new	
2.09 equivalent (from ltpatch) . .	643	<code>\AtBeginDvi</code> concept	879
General: more color changes. . . .	630	General: Removed old definition of	
1994-11-09 ltfssbas.dtx v2.1v		<code>\@testfp</code>	852
<code>\@vpt</code> : (DPC) macros added, from		1994-11-14 ltspace.dtx v1.2d	
<code>setsz</code> .dtx	401	<code>\:</code> (DPC) Macro modified	283
(DPC) reduce save stack usage		1994-11-14 lttab.dtx v1.0i	
<code>latex/1742</code>	401	<code>\tabularnewline</code> : (DPC) Macro	
		added	659

1994-11-16 fontdef.dtx v2.2h		1994-11-18 ltfsstrc.dtx v2.3m	
General: (DPC) Removed <code>\{</code> and <code>\}</code>	508	General: <code>\next</code> to <code>\reserved@f</code>	424
1994-11-17 ltboxes.dtx v1.0q		1994-11-18 ltmath.dtx v1.0p	
General: <code>\@tempa</code> to <code>\reserved@a</code>	630	<code>\phantom</code> : (DPC) colour support	600
1994-11-17 ltclass.dtx v1.0l		(DPC) use <code>\expandafter</code> instead of	
General: <code>\@tempa</code> to <code>\reserved@a</code>	754	<code>\next</code>	600
1994-11-17 ltctrl.dtx v1.0b		<code>\prime@s</code> : (DPC) use <code>\@let@token</code>	
General: <code>\@tempa</code> to <code>\reserved@a</code>	248	instead of <code>\next</code> and	
1994-11-17 ltdefs.dtx v1.0g		<code>\expandafter</code> instead of <code>\nxt</code>	605
General: <code>\@tempa</code> to <code>\reserved@a</code>	73	<code>\smash</code> : (DPC) colour support	601
1994-11-17 ltdirchk.dtx v1.0j		(DPC) use <code>\expandafter</code> instead of	
General: <code>\@tempa</code> to <code>\reserved@a</code>	1	<code>\next</code>	601
1994-11-17 lterror.dtx v1.2h		1994-11-21 ltfloat.dtx v1.1f	
General: <code>\@tempa</code> to <code>\reserved@a</code>	252	<code>\@endfloatbox</code> : Added reset of	
1994-11-17 ltfiles.dtx v1.0n		minipage flag	728
General: <code>\@tempa</code> to <code>\reserved@a</code>	299	Corrected position of	
1994-11-17 ltfinal.dtx v1.0o		<code>\outer@nobreak</code>	728
General: <code>\@tempa</code> to <code>\reserved@a</code>	927	<code>\@marginparreset</code> : Macro added	732
1994-11-17 ltfloat.dtx v1.1e		<code>\@savemarbox</code> : Added <code>\@setminipage</code>	
General: <code>\@tempa</code> to <code>\reserved@a</code>	719	etc	732
1994-11-17 ltfntcmd.dtx v3.3p		Added resetting of size and font	732
General: <code>\@tempa</code> to <code>\reserved@a</code>	530	Changed to <code>\color@vbox</code>	732
1994-11-17 ltfsbas.dtx v2.1w		Use <code>\@setnobreak</code> etc	732
General: <code>\@tempa</code> to <code>\reserved@a</code>	376	<code>\@setminipage</code> : Macro added	726
1994-11-17 ltssdcl.dtx v2.1m		<code>\@setnobreak</code> : Macro added	726
General: <code>\@tempa</code> to <code>\reserved@a</code>	454	<code>\@xfloat</code> : Added <code>\@setminipage</code>	724
1994-11-17 ltfsstrc.dtx v2.3l		Added resetting of size and font	724
General: <code>\@tempa</code> to <code>\reserved@a</code>	424	Changed to <code>\color@vbox</code> so that	
1994-11-17 ltmath.dtx v1.0o		large floats overflow at the	
General: <code>\@tempa</code> to <code>\reserved@a</code>	598	bottom	724
1994-11-17 ltmiscen.dtx v1.0v		Missing percents reinserted after 4,	
General: <code>\@tempa</code> to <code>\reserved@a</code>	577	8: these are not numbers.	723
1994-11-17 ltoutenc.dtx v1.6h		Use <code>\@setnobreak</code>	724
General: (DPC) <code>\@tempa</code> to		<code>\@xympar</code> : Changed to <code>\color@vbox</code>	733
<code>\reserved@a</code>	320	1994-11-21 ltoutput.dtx v1.1i	
1994-11-17 ltoutput.dtx v1.1h		<code>\@addtocurcol</code> : Added <code>\if@nobreak</code>	
General: <code>\@tempa</code> to <code>\reserved@a</code>	852	test before float box	893, 897
1994-11-17 ltpictur.dtx v1.0f		<code>\@specialoutput</code> : Added <code>\if@nobreak</code>	
General: <code>\@tempa</code> to <code>\reserved@a</code>	670	test	872
1994-11-17 ltsect.dtx v1.0i		<code>\@topnewpage</code> : Changed to	
General: <code>\@tempa</code> to <code>\reserved@a</code>	706	<code>\color@vbox</code>	868
1994-11-17 lttab.dtx v1.0j		1994-11-22 ltssdcl.dtx v2.1o	
General: <code>\@tempa</code> to <code>\reserved@a</code>	645	General: wrap long lines	454
1994-11-18 ltboxes.dtx v1.0r		1994-11-22 ltoutenc.dtx v1.6i	
<code>\color@vbox</code> : macro added for color		General: Corrected <code>\dots</code> so that	
support	633	there's no kerning in monowidth	
1994-11-18 ltfinal.dtx v1.0n		fonts.	320
General: re-allow slots 127–255	939	Corrected typo with	
1994-11-18 ltfsbas.dtx v2.1x		<code>\mathunderscore</code> .	320
General: (DPC) use <code>\reserved@f</code> not		Fixed empty accents. Again.	320
<code>\next</code>	376	1994-11-24 ltdefs.dtx v1.2h	
1994-11-18 ltssdcl.dtx v2.1m		<code>\@newenv</code> : Added test for <code>\endgraf</code>	80
<code>\DeclareMathDelimiter</code> : (DPC)			
<code>\expandafter</code> instead of <code>\next</code>	473		

1994-11-25 ltplain.dtx v1.1f	1994-12-01 ltfinal.dtx v1.0p
General: (DPC) Comment out lots of	General: Renamed lthyphen.* to
obsolete code 14	hyphen.*. 927
1994-11-26 ltfloat.dtx v1.1b	1994-12-01 lthyphen.dtx v1.0g
\footnote: (ASAJ) Added	General: Rename lthyphen.ltx/cfg to
\protected@xdef. 738	hyphen.ltx/cfg 925
1994-11-28 ltcntrl.dtx v1.0c	1994-12-01 ltplain.dtx v1.1g
General: Documentation	General: (DPC) More doc changes . . 14
improvements 248	1994-12-02 fontdef.dtx v2.2i
1994-11-30 ltfiles.dtx v1.0o	General: Commented out \ldots.
\@dofilelist: Macro added 319	ASAJ. 506
\listfiles: Use \@dofilelist . . . 318	1994-12-02 ltffssini.dtx v2.2c
\nofiles: There is no	\copyright: \copyright is now in
\@gobblethree. 306	ltoutenc. ASAJ 501
1994-11-30 ltffssbas.dtx v2.1y	1994-12-02 ltlists.dtx v1.0e
\fontshape: Use \@current@cmd in	\@trivlist: RmS: Added check for
\@enc@update. ASAJ. 385	looping 621
1994-11-30 ltmath.dtx 1.0q	1994-12-02 ltoutenc.dtx 1.7b
General: ASAJ:	General: Redefined \a properly. . . . 332
\DeclareMathOperator moved to	1994-12-02 ltoutenc.dtx v1.7b
AMS _{TEX} 598	General: Fixed a bug with \a. 320
1994-11-30 ltmiscen.dtx v1.0w	1994-12-04 lthyphen.dtx v1.0h
\@enddocument@kernel@warnings:	General: Documentation edits for
(DPC) Do warnings even for	/1989 925
\nofiles 579	1994-12-05 ltoutenc.dtx v1.7c
\enddocument: (DPC) Use	General: Added braces to
\@dofilelist 579	\textcircled. 320
1994-11-30 ltoutenc.dtx 1.7a	1994-12-06 ltffssbas.dtx v2.1z
General: Redefined \a for the new	\DeclareFontEncoding: use
scheme. 332	\nfss@catcodes 380
1994-11-30 ltoutenc.dtx v1.6g	\nfss@catcodes: Added tab char as
General: Removed new definitions of	well 390
\patterns and \hyphenation,	1994-12-08 ltoutenc.dtx v1.7d
since encoding-specific commands	General: Added \null and \sh@ft to
now expand in the mouth. 332	\b and \d. 320
1994-11-30 ltoutenc.dtx v1.7a	1994-12-08 lttab.dtx v1.0k
General: Added new code for	\@array: Add \tabularnewline . . . 658
encoding-specific commands.	\@tabularnewline: (DPC) Made it
These now expand in the mouth,	\relax 659
which means that ligaturing and	1994-12-09 ltbibl.dtx v1.1d
kerning can happen. 320	\bibliographystyle: (DPC) Allow
Always load the enc.def file, so that	use in preamble. 747
the default encoding for the	1994-12-10 ltfloat.dtx v1.1g
commands will change. 363	\@dblfloat: Old version reinstated
Redefined \@changed@cmd to expand	temporarily 722
in the mouth. 324	\@dblflset: Macro removed
Removed \@changed@x@mouth since	temporarily 722
\@changed@x now expands in the	Old version reinstated temporarily 722
mouth. 324	\@setfps: Macro removed
Rewrote \@text@composite so it	temporarily 723
allows an empty argument, or an	\@xdblfloat: Macros reinserted
argument containing lots of	temporarily 728
commands. 326	\@xfloat: Old version reinstated
	temporarily 723

Sanitization added temporarily . .	723	<code>\endfilecontents</code> : Close input check stream: latex/1487	786
General: Some temps reinserted temporarily	719	1995-04-21 ltfinal.dtx v1.0q General: Allow initial patch level 0 .	942
<code>\fps@dbl</code> : Macro removed temporarily	723	1995-04-21 ltoutenc.dtx v1.7h General: Added <code>\null \k</code> latex/1274	320
1994-12-10 ltfntcmd.dtx v3.3q <code>\@math@egroup</code> : Don't read arguments	537	1995-04-22 ltfiles.dtx v1.0p <code>\includeonly</code> : Allow blanks in argument	307
<code>\check@nocorr@</code> : Use <code>\space</code> command for comparison	534	1995-04-22 ltmiscen.dtx v1.0x General: Removed extra def of <code>\gobble</code>	577
1994-12-10 ltfsdcl.dtx v2.1p <code>\document@select@group</code> : Surround with braces (add fourth arg) . . .	459	1995-04-23 ltsect.dtx v1.0j <code>\addcontentsline</code> : Use <code>\contentsline</code> internally.	715
<code>\select@group</code> : Surround with braces (add fourth arg)	457	1995-04-24 ltbibl.dtx v1.1e <code>\@citex</code> : Add <code>\mbox</code> to undefined case: latex/1239.	746
1994-12-10 ltoutenc.dtx v1.7e General: Added documentation for the OML encoding.	320	1995-04-24 ltbibl.dtx v1.1f <code>\bibcite</code> : Make <code>\@onlypreamble</code> /1388.	746
Replaced width with <code>\@width</code> and ditto height in vrules.	320	1995-04-24 ltcntrl.dtx v1.0d <code>\@for</code> : Don't expand second argument with <code>\edef</code> : /1317 (DPC)	251
1994-12-14 ltoutenc.dtx v1.7f General: Added braces to <code>\copyright</code> so it works unbraced in subscripts.	320	1995-04-24 ltoutput.dtx v1.1j <code>\fl@tracemessage</code> : Do not add to kernel unless 'trace' specified . .	910
Added check for math mode in <code>\@changed@cmd</code>	320	1995-04-24 ltoutput.dtx v1.1l <code>\@begindivbox</code> : Add <code>\vbox</code> latex/1392	864
Commented out <code>\textasciicircum</code> , <code>\textasciitilde</code> , <code>\textbackslash</code> , <code>\textbar</code> , <code>\textgreater</code> , <code>\textthyphenchar</code> , <code>\textthyphen</code> and <code>\textless</code> to save memory.	320	<code>\@writsetup</code> : Reset <code>\@</code> latex/1451 (DPC)	880
1995-01-12 ltmath.dtx v1.2y classes <code>\@eqnnum</code> : Added <code>\normalcolor</code> . . .	610	1995-04-24 ltpage.dtx v1.0f <code>\fussy</code> : reset <code>\emergencystretch</code> latex/1344	753
1995-03-03 ltoutenc.dtx 1.7g General: Corrected an error in documentation referring to the tabular rather than the tabbing environment.	332	1995-04-24 ltplain.dtx v1.1h <code>\newlanguage</code> : Remove remaining <code>\outer</code> declarations.	17
1995-04-02 ltfntcmd.dtx v3.3r <code>\@math@egroup</code> : Read them again to be able to add <code>\relax</code>	537	1995-04-24 ltxref.dtx v1.1e <code>\newlabel</code> : Make <code>\@onlypreamble</code> for /1388.	574
1995-04-02 ltfsdcl.dtx v2.1q <code>\document@select@group</code> : fix problem for pr/1275	459	1995-04-25 ltdefns.dtx v1.2i <code>\@check@c</code> : Make <code>\long</code> for latex/1346	81
<code>\select@group</code> : fix problem for pr/1275	457	<code>\new@environment</code> : Parse arguments slowly but safely /1507	79
<code>\set@mathdelimiter</code> : fix pr/1329 . .	476	1995-04-25 ltfiles.dtx v1.0q <code>\document</code> : Removed execution of <code>\every@size</code> latex/1407	302
1995-04-02 ltfsini.dtx v2.2d <code>\not@math@alphabet</code> : add <code>\noexpand</code> to second part of message	499	1995-04-25 ltsect.dtx v1.0k <code>\@dottedtocline</code> : Added <code>\hbox</code> around dots.	717
1995-04-21 ltclass.dtx v1.0m <code>\DeclareOption*</code> : Made long /1498	769	1995-04-27 ltboxes.dtx v1.0s <code>\@frameb@x</code> : Move <code>\leavevmode</code> for graphics/1512	636

<code>\@iframebox</code> : Move <code>\leavevmode</code> for graphics/1512	636	1995-05-07 ltsect.dtx v1.0o General: Use <code>\hb@xt@</code>	706
<code>\@iirsbox</code> : Move <code>\leavevmode</code> for graphics/1512	643	1995-05-07 lttab.dtx v1.0l General: Use <code>\hb@xt@</code>	645
<code>\@irsbox</code> : Move <code>\leavevmode</code> for graphics/1512	643	1995-05-08 ltbibl.dtx v1.1g <code>\@citex</code> : Use <code>\@firstofone</code>	746
<code>\fbox</code> : Move <code>\leavevmode</code> for graphics/1512	635	<code>\bibitem</code> : Removed unnecessary braces	746
<code>\raisebox</code> : Move <code>\leavevmode</code> for graphics/1512	642	<code>\nocite</code> : Use <code>\@firstofone</code>	747
1995-04-27 ltfiles.dtx v1.0r <code>\document</code> : Added <code>\global</code> to support groups in hook	303	1995-05-08 ltdefs.dtx v1.2k <code>\typein</code> : Use <code>\@firstofone</code>	75
1995-04-27 ltmiscen.dtx v1.0y <code>\enddocument</code> : <code>\@checkend</code> moved after hook	578	1995-05-08 ltdefs.dtx v1.2l <code>\typein</code> : Remove unnecessary braces Replace <code>\def</code> by <code>\let</code>	75
1995-04-27 ltplain.dtx v1.1i General: Move <code>\hang</code> and <code>\textindent</code> to latex209.def	30	1995-05-08 ltfstrc.dtx v2.3n <code>\ifnot@nil</code> : Use <code>\@firstofone</code>	437
1995-04-29 ltcntrl.dtx v1.0e General: Moved init of <code>\protect</code> to ltdefs.dtx	251	1995-05-11 fontdef.dtx v2.2j General: Updates to some plain macros	506
Removed unused defs for <code>\@setprotect</code> and <code>\@resetprotect</code>	251	1995-05-12 ltclass.dtx v1.0n <code>\DeclareOption*</code> : Use <code>\toks@</code> to remove need to double hash /1557	769
1995-04-29 ltdefs.dtx v1.2j <code>\protect</code> : Init <code>\protect</code> here	85	1995-05-12 ltfloat.dtx v1.1h <code>\@footnotemark</code> : Add <code>\nobreak</code> to allow hyphenation. latex/1605	740
1995-04-29 ltpar.dtx v1.1b General: (TO) Comments clean-up.	262	1995-05-12 ltpictur.dtx v1.0h <code>\pictur@</code> : Macro added for latex/1355	672
1995-05-02 ltsect.dtx v1.0l <code>\@dottedtocline</code> : Don't reset to <code>\rmfamily</code>	717	1995-05-12 ltvers.dtx v1.0e General: Add autoload docstrip guards Check for format older than 1 year	36
1995-05-03 ltsect.dtx v1.0m General: TO: Promoted documentation to doc.sty standard	706	1995-05-13 ltfstrc.dtx v2.3o General: Use single hash mark in <code>\DeclareOption</code>	425
1995-05-06 ltsect.dtx 1.0n <code>\@seccntformat</code> : Use <code>\quad</code> instead of <code>\hspace</code>	712	1995-05-16 ltfloat.dtx v1.1i <code>\@makefnmark</code> : Now use <code>\textsuperscript.</code>	737
<code>\@sect</code> : Added <code>\relax</code> after <code>\@seccntformat</code> just in case	710	<code>\textsuperscript</code> : Command added./pr1503	737
1995-05-07 ltboxes.dtx v1.0t General: Use <code>\hb@xt@</code>	630	<code>\thefootnote</code> : Streamlined parts of code.	736
1995-05-07 ltdefs.dtx v1.2k <code>\hb@xt@</code> : Macro added	74	1995-05-17 ltboxes.dtx v1.0u <code>\@irsbox</code> : Removed surplus braces	643
1995-05-07 ltmath.dtx v1.0r General: Use <code>\hb@xt@</code>	598	1995-05-17 ltdefs.dtx v1.0o <code>\g@addto@macro</code> : Make long for latex/1522	102
1995-05-07 ltoutput.dtx v1.1m General: Use <code>\hb@xt@.</code>	852	1995-05-17 ltlists.dtx v1.0g <code>\@item</code> : Removed surplus braces	625
1995-05-07 ltpictur.dtx v1.0g General: Use <code>\hb@xt@</code>	670	<code>\@nbitem</code> : Removed surplus braces	626
1995-05-07 ltplain.dtx v1.1j General: Use <code>\hb@xt@</code>	14	<code>\enumerate</code> : Use <code>\thr@@</code> and remove surplus braces	627
		<code>\itemize</code> : Use <code>\thr@@</code>	628
		1995-05-18 ltfloat.dtx v1.1j <code>\@makefnmark</code> : Added <code>\normalfont.</code>	737

<code>\thempfootnote</code> : Added <code>\itshape</code>	736	<code>\textsuperscript</code> : Use	
1995-05-19 <code>ltpictur.dtx</code> v1.1a		<code>\@textsuperscript</code>	737
General: Support autoloading feature	670	1995-05-24 <code>ltfssbas.dtx</code> v3.0a	
1995-05-20 <code>ltcounts.dtx</code> v1.1b		General: (DPC) Make file from	
<code>\@definecounter</code> : Streamlined code	368	previous file, <code>fam.dtx</code> 1995/05/20	
<code>\@fnsymbol</code> : Allowing both text and		v2.2d	376
math	371	<code>\mathgroup</code> : (DPC) No need to	
<code>\fnsymbol</code> : Streamlined code	370	redefine <code>\newfam</code> as not outer	376
1995-05-20 <code>ltcounts.dtx</code> v1.1c		1995-05-24 <code>ltfsscmp.dtx</code> v3.0a	
<code>\@definecounter</code> : And do it right	368	General: (DPC) Make file from	
1995-05-20 <code>ltfloat.dtx</code> v1.1k		previous file, <code>fam.dtx</code> 1995/05/20	
<code>\@makefnmark</code> : Moved <code>\normalfont</code>		v2.2d	449
back and use <code>\@textsuperscript</code>	737	1995-05-24 <code>ltfssdcl.dtx</code> v3.0a	
Moved <code>\normalfont</code> to		General: (DPC) Make file from	
<code>\textsuperscript</code>	737	previous file, <code>latint.dtx</code> 1995/05/21	
<code>\textsuperscript</code> : Use <code>\normalfont</code> .	737	v2.1t	454
1995-05-21 <code>ltfssdcl.dtx</code> v2.1t		1995-05-24 <code>ltfssini.dtx</code> v3.0a	
<code>\DeclareMathRadical</code> : Allow for		General: (DPC) Make file from	
undefined cs names	476	previous file, <code>lfont.dtx</code> 1995/05/23	
1995-05-21 <code>ltlists.dtx</code> v1.0f		v2.2e	481
General: Moved to <code>doc.sty</code> standard	613	<code>\cal</code> : (DPC) Remove definition	504
1995-05-21 <code>ltmath.dtx</code> v1.0r		<code>\mit</code> : (DPC) Remove definition	504
<code>\@sqrt</code> : Use <code>\sqrtsign</code>	607	1995-05-24 <code>ltfssstrc.dtx</code> v3.0a	
General: Remove <code>\mathhexbox</code> from		General: (DPC) Make file from	
this file	603	previous file, <code>tracefnt</code> 1995/05/16	
Update some plain macros	598	v2.3o	424
<code>\lefteqn</code> : Use <code>\rlap</code>	609	1995-05-24 <code>ltfssstrc.dtx</code> v3.0b	
<code>\r@t</code> : Use <code>\sqrtsign</code> instead of		General: (DPC) Fix <code>\ProvidesFile</code>	
<code>\sqrt</code>	600	usage	424
1995-05-21 <code>ltoutenc.dtx</code> v1.7h		1995-05-25 <code>ltclass.dtx</code> v1.0p	
<code>\@inmathwarn</code> : Added several		<code>\endfilecontents</code> : Delete	
<code>\@onlypreamble</code>	324	<code>\filecontents</code> after preamble	786
1995-05-21 <code>ltoutenc.dtx</code> v1.7j		1995-05-25 <code>ltfilehook.dtx</code> v1.0t	
General: Updated some plain macros	336	<code>\unqu@tefilef@und</code> : (CAR) added	
1995-05-21 <code>ltplain.dtx</code> v1.1j		<code>\long</code>	813
General: Moved some code to other		1995-05-25 <code>ltfiles.dtx</code> v1.0s	
files	14	<code>\document</code> : Added check for <code>\topskip</code>	
1995-05-22 <code>ltplain.dtx</code> v1.1k		zero	303
General: Definitions of <code>\footins</code> and		1995-05-25 <code>ltfiles.dtx</code> v1.0t	
<code>\footnoterule</code> moved to <code>ltfloat</code>	31	<code>\@iffilenamepath</code> : (CAR) added <code>\long</code>	314
1995-05-22 <code>lttab.dtx</code> v1.1a		<code>\document</code> : Corrected typo	303
General: Support autoloading feature	645	<code>\IfFileExists@</code> : (CAR) added <code>\long</code>	312
1995-05-23 <code>ltfssini.dtx</code> v2.2e		<code>\nofiles</code> : (CAR) added <code>\long</code>	306
<code>\newfont</code> : Font assignment made local		<code>\protected@write</code> : (CAR) added	
again.	500	<code>\long</code>	306
1995-05-24 <code>ltdefns.dtx</code> v1.1l		1995-05-25 <code>ltfloat.dtx</code> v1.1m	
<code>\newif</code> : (DPC) New implementation	80	<code>\@savemarbox</code> : (CAR) Resettings	
1995-05-24 <code>ltdefns.dtx</code> v1.2m		moved to hook	732
<code>\typein</code> : (DPC) New implementation	75	<code>\@xfloat</code> : (CAR) Resettings moved to	
1995-05-24 <code>ltfloat.dtx</code> v1.1l		hook	724
<code>\@textsuperscript</code> : Command		1995-05-25 <code>ltlists.dtx</code> v1.0i	
added.	737	<code>\endtrivlist</code> : Macros moved from	
General: Moved definition of <code>\footins</code>		<code>ltspace.dtx</code>	622
and <code>\footnoterule</code> from <code>ltplain</code>	736		

1995-05-25 ltmath.dtx v1.3c classes		
<code>\@eqnnum</code> : replace		
<code>\reset@font\rmfamily</code> with		
<code>\normalfont</code> (PR 1578)	610	
1995-05-25 ltspace.dtx v1.2f		
<code>\@vbsphack</code> : (CAR) not used so		
‘removed’.	289	
<code>\@vspacer</code> : (CAR) <code>\@restorepar</code>		
added to avoid possible infinite tail		
recursion caused by a typo in the		
argument.	292	
(CAR) macros modified to be more		
efficient	292	
General: Macros moved to ltlists.dtx	279	
1995-05-26 ltdefs.dtx v1.2n		
<code>\@gobblefour</code> : (CAR) Added <code>\longs</code>	82	
1995-05-26 ltmath.dtx v1.0s		
<code>\@eqnnum</code> : Removed <code>\rmfamily</code> (PR		
1578), replaced <code>\reset@font</code> with		
<code>\normalfont</code>	607	
1995-05-26 ltpage.dtx v1.0g		
<code>\ps@plain</code> : removed <code>\rmfamily</code> (PR		
1578)	751	
1995-05-27 ltssbas.dtx v3.0b		
<code>\mathgroup</code> : (FMi) But a need to		
define <code>\new@mathgroup</code>	376	
1995-06-05 fontdef.dtx v2.2k		
General: Moved math commands from		
ltoutenc.dtx.	524	
1995-06-05 ltfinal.dtx v1.0r		
General: Added <code>\MakeUppercase</code> and		
<code>\MakeLowercase</code>	927	
1995-06-05 ltoutenc.dtx v1.7k		
<code>\@inmathwarn</code> : Removed		
<code>\protected@cmd</code> and replaced with		
explicit <code>\noexpand</code>	324	
General: Allowed		
<code>\ProvideTextCommandDefault</code>		
after the preamble.	326	
Commented out <code>\textless</code> and		
<code>\textgreater</code>	334	
Moved math commands to		
fontdef.dtx.	336	
Save some tokens in		
<code>\textvisiblespace</code> and		
<code>\textunderscore</code>	334	
1995-06-06 ltfinal.dtx v1.0s		
General: Made <code>\MakeUppercase</code> and		
<code>\MakeLowercase</code> brace their		
argument.	927	
1995-06-09 ltoutenc.dtx v1.7l		
<code>\DeclareTextComposite</code> : Rewrote		
<code>\DeclareTextComposite</code> to define		
the composite as a no-argument		
command rather than a		
two-argument command.	327	
1995-06-11 ltspace.dtx v1.2g		
<code>\restorecr</code> : (CAR) <code>\relax</code> added to		
stop silent eating of *.	297	
1995-06-13 ltfinal.dtx v1.0t		
General: Add patch level string more		
carefully	942	
Call <code>\errorstopmode</code>	943	
1995-06-13 ltpictur.dtx v1.1b		
General: Use <code>\ProvidesFile</code> in		
autoload	670	
1995-06-14 lttab.dtx v1.1b		
General: Use <code>\ProvidesFile</code> in		
autoload	645	
1995-06-15 ltssbas.dtx v3.0c		
General: (DPC) minor documentation		
changes	376	
1995-06-15 ltsscmp.dtx v3.0b		
General: (DPC) minor documentation		
edits	449	
1995-06-15 ltssdcl.dtx v3.0b		
General: (DPC) minor documentation		
changes	454	
1995-06-19 ltbibl.dtx v1.1h		
<code>\bibcite</code> : Call <code>\@newl@bel</code> so		
repeated keys produce better		
warning.	746	
1995-06-19 ltclass.dtx v1.0q		
<code>\documentclass</code> : Don’t redefine		
<code>\usepackage</code> in compat mode for		
/1634	773	
1995-06-19 ltxref.dtx v1.1e		
<code>\newlabel</code> : Use <code>\@newl@bel</code> to share		
code with <code>\bibcite</code>	574	
1995-06-28 ltssini.dtx v3.0b		
General: (DPC) Fix documentation		
typos	481	
1995-06-28 ltmath.dtx v1.0t		
General: minor doc edits	598	
1995-07-02 ltplain.dtx v1.1n		
General: Removed surplus ‘by’ and ‘=’		
in various places	14	
<code>\offinterlineskip</code> : Replaced 1000 by		
<code>\@m</code>	28	
<code>\showoutput</code> : Use <code>\showoverfull</code> to		
save space	32	
<code>\tracingall</code> : Use <code>\showoutput</code> to		
save space	32	
1995-07-03 ltdefs.dtx v1.2o		
<code>\set@typeset@protect</code> : Use		
<code>\@typeset@protect</code> for init	85	

1995-07-03 ltfnctcmd.dtx v3.3s	1995-07-14 ltbibl.dtx v1.1i
\@st@ic: Use clean interface for jump 536	\bibcite: Remove \@onlypreamble so still defined in new \enddocument 746
1995-07-05 ltfnctcmd.dtx v3.3s	1995-07-14 ltxref.dtx v1.1g
\@st@ic: Renamed from \@test@next 536	\newlabel: Remove \@onlypreamble so still defined in new \enddocument 574
1995-07-05 ltspace.dtx v1.2h	1995-07-19 ltfssini.dtx v3.0d
\@gnewline: Use \@break 285	General: (DPC) TeX2 support 504
\@no@pgbk: Macro replaces \@pgbk and \@nopgbk 283	1995-07-20 ltboxes.dtx v1.0v
\@nopagebreak: Reimplemented both using \@no@pgbk 282	\@isavebox: Use \@sbox 634
1995-07-09 ltcntrl.dtx v1.0f	\@isavepicbox: Use \@sbox 634
\@iforloop: Reimplemented using Kabelschacht method 251	1995-07-21 ltoutput.dtx v1.1o
\@iwhiledim: Reimplemented using Kabelschacht method 249	\@writsetup: Command added ... 879
\@iwhilenum: Reimplemented using Kabelschacht method 249	New, experimental, versions: need in-lining 879
\@iwhiles: Reimplemented using Kabelschacht method 249	1995-08-09 ltmath.dtx v1.0u
\@tfor: Reimplemented using Kabelschacht method 251	General: Added code for class options leqno and fleqn 610
1995-07-09 ltlists.dtx v1.0j	1995-08-11 ltlength.dtx v1.1b
enumerate: Use \@expandafter 627	General: Doc typos fixed for latex/753 374
itemize: Use \@expandafter 628	1995-08-16 ltcntrl.dtx v1.0g
1995-07-12 ltpictur.dtx v1.1d	\@break@tfor: Made long 251
General: allow 2e commands in 209 mode. latex/1737 670	\@forloop: Made defs long 251
1995-07-13 ltdefs.dtx v1.0p	\@fornoop: Made defs long 251
General: Updates to documentation . 73	\@iforloop: Made defs long 251
1995-07-13 ltfiles.dtx v1.0u	\@iwhiledim: Made defs long 249
General: Updates to docu 299	Removed \@whilenoop 249
1995-07-13 ltfssbas.dtx v3.0d	\@iwhilenum: Made defs long 249
\@@defaultsubs: macro added 396	Removed \@whilenoop 249
\@defaultsubs: macro added 396	\@iwhiles: Removed
General: minor documentation changes 376	\@whilesnoop 249
\@wrong@fontshape: Change a macro not a switch to flag default font substitutions 395	\@tfor: Made defs long 251
1995-07-13 ltmiscen.dtx v1.0z	1995-08-16 ltfiles.dtx v1.0v
\@centercr: Use \@nobreak 589	\document: set \@maxdepth 303
\@enddocument@kernel@warnings: Use \@defaultsubs instead of switch 580	set \@do globally 303
\@writefile: Added missing percent and use \@relax in the THEN case 583	set \@topskip globally 303
\@xobeysp: Use \@nobreak 592	1995-08-24 ltfssbas.dtx v3.0f
General: Improve Documentation .. 577	General: Added autoload code 376
\enddocument: Set \@setckpt to \@gobbletwo instead of defining it by hand 578	1995-08-24 ltfsssrc.dtx v3.0c
Shorten redefinition of \bibcite and \newlabel 579	General: Macro \gobble@font@spec removed 438
	\tryis@simple: 445
	1995-08-25 ltoutput.dtx v1.1p
	General: Support autoloading feature (FMi). 852
	1995-09-01 lterror.dtx v1.2i
	General: Add autoload support ... 252
	1995-09-01 ltplain.dtx v1.1m
	\empty: Use \@let to save space 28
	\I: Use \@let to save space 27

1995-09-14 ltplain.dtx v1.1o	1995-10-11 ltoutput.dtx v1.1r
General: Moved <code>\multispan</code> to	<code>\clearpage</code> : Added a check so that it
ltxab.dtx 14	does not lose the argument of
1995-09-14 ltxab.dtx v1.1c	<code>\twocolumn[...]</code> 865
<code>\cline</code> : (DPC) New implementation 668	1995-10-16 ltbibl.dtx v1.1j
1995-09-15 ltfssini.dtx v3.0e	<code>\cite</code> : (DPC) Make robust 746
General: (DPC) Modify TeX2	1995-10-16 ltboxes.dtx v1.0w
message 504	General: Clarify makebox description 630
1995-09-19 ltmiscen.dtx v1.1a	1995-10-16 ltdefns.dtx v1.2u
<code>\verb</code> : Put <code>\@noligs</code> after	<code>\@ifstar</code> : (DPC) New
<code>\verbatim@font</code> where it belongs. 597	implementation, for /1910 98
1995-10-01 ltfiles.dtx LaTeX2e	<code>\new@command</code> : (DPC) Use <code>\@testopt</code>
<code>\@addtofilelist</code> : Macro added . . . 318	/1911 77
1995-10-02 ltdefns.dtx v1.2q	<code>\new@environment</code> : (DPC) Use
<code>\@ifnch</code> : Use <code>\@let@token</code> for	<code>\@testopt</code> /1911 79
internal/924, save <code>\reserved@e</code> . . 98	<code>\typein</code> : (DPC) Use <code>\@testopt</code> /1911 75
<code>\@ifnextchar</code> : Use <code>\@let@token</code> . . . 97	1995-10-16 ltfssini.dtx v3.0f
<code>\@protected@testopt</code> : Macro added . . 78	<code>\reset@font</code> : Added <code>\relax</code> after
<code>\@testopt</code> : Macro added 77	<code>\usefont</code> , as the latter eats up
<code>\@xargdef</code> : New implementation,	spaces. 501
using <code>\@test@opt</code> 77	1995-10-16 ltmath.dtx v1.0y
1995-10-03 fontdef.dtx v2.2l	<code>\@yeqncr</code> : (DPC) Use <code>\@testopt</code>
General: <code>\@@sqrt</code> from patch file for	/1911 608
/1701 506	<code>\sqrt</code> : (DPC) Make robust /1808 . . 607
1995-10-03 ltdefns.dtx v1.2r	1995-10-16 ltspace.dtx v1.2j
<code>\typein</code> : Add missing <code>\@typein</code> for	<code>\nolinebreak</code> : (DPC) Use <code>\@testopt</code>
/1710 (from patch file) 75	/1911 282
1995-10-03 ltpictur.dtx v1.1e	<code>\nopagebreak</code> : (DPC) Use <code>\@testopt</code>
General: New autoloader code 670	/1911 282
1995-10-04 ltfssbas.dtx v3.0g	1995-10-16 ltthm.dtx v1.0g
General: Modify autoloader code 376	General: Revert to previous
1995-10-04 ltfssstrc.dtx v3.0d	<code>\newtheorem</code> behaviour 702
General: (DPC) Modify autoloader	1995-10-17 ltclass.dtx v1.0r
code 424	<code>\@providesfile</code> : Delay definition of
1995-10-04 ltxab.dtx v1.1d	<code>\ProvidesFile</code> till ltfinal 767
General: Modify autoloader support . . 645	<code>\ProcessOptions*</code> : Reset
1995-10-06 ltfiles.dtx v1.0w	<code>\CurrentOption</code> for
<code>\@missingfileerror</code> : Autoloader error 316	graphics/1873 772
1995-10-09 lterror.dtx v1.2j	1995-10-17 ltdirchk.dtx v1.0l
General: Modify autoloader support . . 252	General: Modify initex version of
1995-10-09 ltoutenc.dtx v1.7m	<code>\ProvidesFile</code> 4
<code>\@inmathwarn</code> : Autoloader error 325	1995-10-17 ltfinal.dtx v1.0v
1995-10-10 ltfssbas.dtx v3.0h	<code>\@providesfile</code> : reset macro 943
<code>\showhyphens</code> : Use <code>\normalfont</code> and	<code>\reserved@b</code> : reset here after the
make colour safe, and	<code>\input</code> above 942
autoloadable 399	1995-10-17 ltplain.dtx v1.1s
1995-10-10 ltfssdcl.dtx v3.0c	<code>\eject</code> : Move <code>\supereject</code> to compat
<code>\non@alpherr</code> : (DPC) autoloader error	file 29
message 458	1995-10-17 ltxab.dtx v1.1e
1995-10-10 ltplain.dtx v1.1r	<code>\@cline</code> : (DPC) Use <code>\@multicnt</code> . . 668
General: Autoloader tracing code 14	<code>\@multispan</code> : (DPC) Macro added. 668
1995-10-10 ltthm.dtx v1.0f	1995-10-19 ltfinal.dtx v1.0w
General: Make <code>\newtheorem</code> ‘only	<code>\@filelist</code> : Move after <code>\reserved@a</code>
preamble’ 702	setting:-) 943

1995-10-20 ltbibl.dtx v1.1k		<code>\@setref</code> : Switch for redefined	
<code>\@citex</code> : Removed redefined flag	746	renamed	574
<code>\nocite</code> : Removed redefined flag	747	<code>\if@multiplelabels</code> : Macro removed	574
1995-10-20 ltclass.dtx v1.0s		1995-10-25 ltalloc.dtx v1.1b	
<code>\@begindocumenthook</code> : Make setting conditional, for autoload version	784	General: General doc improvements	246
1995-10-20 ltfsbas.dtx v3.0i		1995-10-25 ltfloat.dtx v1.1n	
General: (DPC) Modify autoload code, change <code>\undefined</code>	376	<code>\@endfloatbox</code> : (CAR) macro added: to unify code for double and single versions	728
1995-10-20 ltfsstrc.dtx v3.0e		<code>\end@dblfloat</code> : (CAR) unify code for double and single versions	727
General: (DPC) Modify autoload code	424	<code>\end@float</code> : (CAR) unify code for double and single versions	726
1995-10-22 ltfsbas.dtx v3.0j		1995-10-25 ltidxglo.dtx v1.1d	
General: (RmS) New size function macro <code>\genb@sfcnt</code> needs to be disabled at <code>\document.</code>	376	General: Doc cleanup	742
1995-10-22 ltfsstrc.dtx v3.0f		1995-10-25 ltsect.dtx v1.0q	
General: Added ‘genb’ and ‘sgenb’ size functions to support new DC font naming scheme.	424	<code>\subparagraphmark</code> : Use <code>\let not</code> <code>\def</code> to save space.	714
1995-10-23 lttab.dtx v1.1f		1995-10-27 ltpictur.dtx v1.1f	
<code>\@settab</code> : (CAR)Ensure that <code>\@hightab</code> increases by at most one	652	General: Move initialization to kernel from autoload file	697
<code>\@startline</code> : (CAR)Ensure that <code>\@nxttabmar</code> is never larger than <code>\@hightab</code>	650	1995-10-31 ltboxes.dtx v1.0x	
<code>\poptabs</code> : (CAR)Ensure that <code>\@curtab</code> is never larger than <code>\@hightab</code>	653	<code>\@finalstrut</code> : Add <code>\nobreak</code> in horizontal/1931	643
<code>\tabbing</code> : (CAR)Make <code>\@hightab</code> consistently a local variable	652	1995-11-01 fontdef.dtx v2.2m	
1995-10-24 ltfiles.dtx v1.1a		General: add <code>\nfss@catcodes</code> for internal/1932	509
<code>\document</code> : Removed multiplelabels switch	302	1995-11-01 ltidirch.dtx v1.0n	
Removed redefined switch	302	General: Initialise <code>\@addtofilelist</code> to <code>\@gobble</code>	4
1995-10-24 ltfsbas.dtx v3.0k		1995-11-01 ltfinal.dtx v1.0x	
<code>\@@defaultsubs</code> : macro removed	396	General: (DPC) Switch meaning of <code>\@addtofilelist</code> for cfg files	933
<code>\wrong@fontshape</code> : Make this code inline since it happens only here	395	1995-11-01 ltfsbas.dtx v3.0m	
1995-10-24 ltmiscen.dtx v1.1b		<code>\DeclareFontShape@</code> : (DPC) Test for <code>\relax not \undefined</code> , internal/1933	377
<code>\@enddocument@kernel@warnings</code> : Changed logic for producing warning messages and removed switch	580	1995-11-01 ltfsini.dtx v3.0g	
Use <code>\@redefined</code> instead of switch	580	General: (DPC) Switch meaning of <code>\@addtofilelist</code> for cfg files	504
1995-10-24 ltxref.dtx v1.1h		1995-11-02 ltfsbas.dtx v3.0n	
<code>\@multiplelabels</code> : Switch for multiplelabels removed	574	<code>\wrong@fontshape</code> : (DPC) Remove extra space with <code>\string</code> for latex/1676	394
<code>\@newl@bel</code> : Switch for multiplelabels replaced by inline code	574	1995-11-02 ltoutenc.dtx v1.7n	
<code>\@redefined</code> : Switch for redefined replaced	573	General: Changed internal name <code>\a</code> to <code>\@tabacckludge</code> to protect against redefinition by malicious users.	332
		1995-11-07 ltlists.dtx v1.0k	
		<code>\@doendpe</code> : Enclosed <code>\setbox0</code> assignment by a group so that it leaves the contents of box 0 intact.	623

- 1995-11-07 ltoutenc.dtx v1.7o
 General: Added `\leavevmode` at start of `\c`, otherwise the output routine might be invoked within the macro. 336
 Changed `\char32` to `\@xxxii` (two tokens less). 337
 Replaced octal number 27 by decimal number 23 to protect against the quote character being active. 337
 Replaced some 0's by `\z@` (faster). 337
- 1995-11-10 ltoutput.dtx v1.1s
`\@shipoutsetup`: Command removed 879
`\@writesetup`: Command removed . 879
 In-lined 879
- 1995-11-14 ltclass.dtx v1.0t
`\@unprocessedoptions`: Allow empty option 785
`\loadwithoptions`: macro added . 774
`\LoadClassWithOptions`: macro added 774
`\RequirePackageWithOptions`: macro added 774
- 1995-11-17 ltffsbas.dtx v3.0m
`\@wrong@font@char`: (DPC) Macro added. latex/1676 396
`\define@newfont`: Redefine `\typeout` latex/1676 389
`\wrong@fontshape`: Support `\@wrong@font@char` latex/1676 . 394
- 1995-11-17 ltoutenc.dtx v1.7p
`\UseTextSymbol`: Support `\@wrong@font@char` latex/1676 . 329
- 1995-11-18 ltoutenc.dtx v1.7q
`\UseTextSymbol`: Modify message slightly 329
- 1995-11-21 fontdef.dtx v2.2n
 General: Incorporate changed figures, as in plain.tex 522
- 1995-11-27 ltffsbas.dtx v3.0n
`\nfss@catcodes`: Reset hash, for definitions in fd files 391
- 1995-11-28 ltfloat.dtx v1.1n
 General: documentation fixes 719
- 1995-11-28 ltfsstrc.dtx v3.0g
 General: documentation fixes 424
- 1995-11-28 ltoutenc.dtx v1.7r
 General: Added math mode checks to text commands. 324
 doc fixes 320
 Renamed `\@changed@x@err` to `\TextSymbolUnavailable`. 324
- 1995-11-29 ltoutenc.dtx v1.7t
 General: Added `\textasciicircum`, `\textasciitilde`, `\textbackslash`, `\textbar`, `\textgreater` and `\textless`. . . 340
 Added `\textasciicircum`, `\textasciitilde`, `\textregistered` and `\texttrademark`. 334
 Added `\textbackslash` and `\textbar`. 333, 344
 Added `\textless` and `\textgreater`. 334, 345
- 1995-12-01 ltoutenc.dtx v1.7u
 General: Made `\SS` a Default, rather than having the default point to the OT1 definition. 334
- 1995-12-04 ltspace.dtx v1.2k
`\nobreakspace`: (Macro added 295
- 1995-12-04 ltspace.dtx v1.2l
`\@xobeysp`: (braces added to definition of tilde 295
- 1995-12-04 preload.dtx v2.4e
 General: Ulrik Vieth. added 12pt OMS and OML preloads /1989 . 529
- 1995-12-05 ltdefs.dtx 1.2w
`\@unexpandable@protect`: Removed `\unexpandable@noexpand` as never used. internal/1733 83
- 1995-12-05 ltfiles.dtx v1.1c
`\document`: `\ignorespaces` added for latex/1933 303
- 1995-12-05 ltfloat.dtx v1.1n
`\@textsuperscript`: Use `\ensuremath` for latex/1984. 737
- 1995-12-05 ltoutenc.dtx v1.7v
`\@inmathwarn`: Changed `\TextSymbolUnavailable` text . . 325
- 1995-12-06 ltffsbas.dtx v3.00
`\nfss@catcodes`: Reset hat, for typeouts etc in fd files 391
- 1995-12-07 ltbibl.dtx v1.1l
`\@citex`: Restored name of `\G@refundefinedtrue` 746
- 1995-12-07 ltfloat.dtx v1.1m
`\@textsuperscript`: Move `\m@th` out of the `\ensuremath` for latex/1984. 737
- 1995-12-07 ltxref.dtx v1.1i
`\@setref`: Switch for `\refundefined` restored 574
`\G@refundefinedtrue`: Renamed (back) from `\G@refundefined` . . 573

1995-12-11 ltoutenc.dtx v1.7w	1996-05-21 ltoutenc.dtx v1.7y
General: Modified <code>\copyright</code> 334	General: Corrected error message (CAR) 363
1995-12-13 ltdefs.dtx 1.2x	1996-05-21 ltsect.dtx v1.0s
<code>\-</code> : Documentation changed. 100	<code>\sect</code> : (DPC) Added extra braces for internal/2148 710
1996-01-10 ltfiles.dtx v1.1d	(DPC) Moved brace to allow commands like <code>\MakeUppercase</code> in 6th argument. Changed <code>\par</code> to <code>\endgraf</code> to allow non-long commands. internal/2148 710
<code>\iffileonpath</code> : Change argument handling to not require doubled hash. latex/2024 314	<code>\esect</code> : (DPC) Added extra braces for internal/2148 713
1996-01-20 ltidxglo.dtx v1.1e	(DPC) Moved brace to allow commands like <code>\MakeUppercase</code> in 4th argument. Changed <code>\par</code> to <code>\endgraf</code> to allow non-long commands. internal/2148 713
<code>\makeglossary</code> : Make no-op after use pr/2048 743	1996-05-23 ltoutenc.dtx v1.7z
<code>\makeindex</code> : Make no-op after use pr/2048 743	<code>\@strip@args</code> : <code>\expandafter</code> added to match other changes for latex/2133 329
1996-01-20 ltspace.dtx v1.2m	<code>\add@accent</code> : macro added. latex/2133 326
<code>\vspace</code> : Made robust 292	<code>\DeclareTextAccent</code> : Reimplemented using <code>\add@accent</code> to save space latex/2133 326
1996-03-25 ltmath.dtx v1.1a	<code>\DeclareTextCompositeCommand</code> : Modified to cope with new <code>\add@accent</code> command: required removal of check for one argument-command 327
<code>\@ensuredmath</code> : Macro added for amslatex/2104 610	1996-05-24 ltoutput.dtx v1.1t
<code>\ensuremath</code> : Reimplement for amslatex/2104 609	<code>\@specialoutput</code> : Check that <code>\@colroom</code> is less than <code>\vsize</code> , indicating that a float has been added 870
1996-04-18 ltpage.dtx v1.0i	Cut-off point changed to <code>1.5\baselineskip</code> 870
General: Improve documentation . . 750	<code>\@topnewpage</code> : Cut-off point changed to <code>2.5\baselineskip</code> 869
1996-04-22 ltmiscen.dtx v1.1c	1996-05-25 ltoutput.dtx v1.1u
General: Improve Documentation . . 577	<code>\@specialoutput</code> : Correct the above check 870
1996-04-22 ltspace.dtx v1.2n	1996-06-03 ltmiscen.dtx v1.1d
General: Documentation Improvements 279	<code>\@verbatim</code> : Exchanged the following two code lines so that <code>\dospecials</code> cannot reset the category code of characters handled by <code>\@noligs</code> . 593
1996-04-22 lttab.dtx v1.1g	General: Move setting of verbatim font and <code>\@noligs</code> 577
<code>\@tabclassz</code> : (DPC) Extra <code>\hskip</code> keeps <code>tabcolsep</code> in empty columns internal/2122 665	<code>\verb</code> : Put setting of verbatim font after <code>\dospecials</code> so that <code>\dospecials</code> cannot reset the
1996-04-23 ltcounts.dtx v1.1d	
General: Documentation improvements 366	
1996-04-24 ltfiles.dtx v1.1e	
<code>\document</code> : (DPC) Reset <code>\AtBeginDocument</code> eg for latex/1297 302	
1996-05-08 ltfstsrc.dtx v3.0h	
<code>\math@egroup</code> : Use <code>\bgroup</code> instead of <code>\begingroup</code> to match a kernel change made in 1994!! 436	
1996-05-09 ltfntcmd.dtx v3.3t	
<code>\check@icr</code> : Default definitions added 534	
1996-05-17 fontdef.dtx v2.2o	
General: <code>\@@sqrt</code> removed, at last 506, 522	
1996-05-17 ltfiles.dtx v1.1f	
<code>\nofiles</code> : added <code>\write</code> to <code>\protected@write</code> for latex/2146 306	
1996-05-18 ltoutenc.dtx v1.7x	
General: Produce error if encoding not found. pr/2054 363	

category code of characters handled by <code>\@noligs</code>	597	<code>\nfss@catcodes</code> : omit <code>\relax</code> as not needed	390
1996-06-10 ltboxes.dtx v1.0y <code>\@parboxto</code> : (DPC) Changed <code>\endgraf</code> to <code>\@@par</code>	638	1996-07-26 ltssdcl.dtx v3.0e <code>\init@restore@version</code> : Removed <code>\ifrestore@version</code> switch and replaced by <code>\init@restore@version</code>	458
1996-06-10 ltsect.dtx v1.0t <code>\@sect</code> : (DPC) Changed <code>\endgraf</code> to <code>\@@par</code>	710	1996-07-26 ltfsstrc.dtx v3.0i <code>\init@restore@glb@settings</code> : macro added replacing <code>\if@inmath</code> switch	435
<code>\@ssect</code> : (DPC) Changed <code>\endgraf</code> to <code>\@@par</code>	713	1996-07-26 ltlists.dtx v1.0l <code>\@item</code> : Remove unnecessary <code>\global</code> before <code>\@minipage</code>	624
1996-06-13 ltdirchk.dtx v1.0r General: documentation improvements mainly from internal/2174	1	Remove unnecessary <code>\global</code> before <code>\@nobreak</code>	625
1996-06-14 lttab.dtx v1.1h <code>\@tabclassz</code> : (DPC) Change both <code>\z@skip</code> to 1sp for latex/2160	665	1996-07-26 ltmath.dtx v1.1b General: Removed <code>\global</code> before <code>\@ignoretrue</code> in various places. . .	598
1996-06-22 ltspace.dtx v1.2o General: Documentation of problems added	279	1996-07-26 ltmiscen.dtx v1.1e <code>\@ignorefalse</code> : put <code>\global</code> into definition	578
1996-07-10 ltfinal.dtx v1.0y <code>\toks</code> : Free up memory from scratch registers /2213	942	<code>\begin</code> : remove <code>\global</code> before <code>\@ignore</code>	584
1996-07-19 ltoutenc.dtx v1.8a <code>\@strip@args</code> : Use char 0 not @ as carrier for <code>\lowercase</code> /2197 . . .	329	<code>\end</code> : remove <code>\global</code> before <code>\@ignore</code>	587
1996-07-26 ltboxes.dtx v1.0z <code>\if@minipage</code> : put <code>\global</code> into definition	639	<code>\ignorespacesafterend</code> : user level macro added	578
1996-07-26 ltclass.dtx v1.0u <code>\@classoptionslist</code> : made only preamble	758	1996-07-26 ltoutput.dtx v1.1v <code>\@testfp</code> : remove <code>\global</code> before <code>\@test</code>	913
<code>\@unusedoptionlist</code> : made only preamble	759	<code>\@xtryfc</code> : remove <code>\global</code> before <code>\@test</code>	887
1996-07-26 ltdefs.dtx v1.2y <code>\@reargdef</code> : third arg picked up by <code>\@yargdef</code>	78	<code>\@ztryfc</code> : remove <code>\global</code> before <code>\@test</code>	888
<code>\@renew@command</code> : use <code>\noexpand</code> instead of <code>\string</code>	79	General: put <code>\global</code> into definition remove <code>\global</code> before <code>\@test</code>	862
use <code>\relax</code> in place of empty arg . .	79	<code>\clearpage</code> : add number of missing percents	865
<code>\@renew@environment</code> : use <code>\relax</code> in place of empty arg	80	1996-07-26 ltplain.dtx v1.1t <code>\sh@ft</code> : replace <code>\dimen\z@</code> by <code>\dimen@</code>	31
1996-07-26 ltfloat.dtx v1.1n <code>\@endfloatbox</code> : remove unnecessary <code>\global</code> before <code>\@minipage</code>	728	1996-07-26 ltsect.dtx v1.0u <code>\@starttoc</code> : removed <code>\global</code> before <code>\@nobreak</code>	715
<code>\@savemarbox</code> : remove unnecessary <code>\global</code> before <code>\@minipage</code>	732	<code>\@xsect</code> : Removed <code>\global</code> before <code>\@nobreak</code>	711
<code>\@setminipage</code> : remove unnecessary <code>\global</code> before <code>\@minipage</code>	726	1996-07-26 ltspace.dtx v1.2p <code>\if@nobreak</code> : put <code>\global</code> inside definition	285
<code>\@setnobreak</code> : remove unnecessary <code>\global</code> before <code>\@nobreak</code>	726	1996-07-27 ltssbas.dtx v3.0q General: <code>\if@inmath</code> switch removed	388
1996-07-26 ltssbas.dtx v3.0p <code>\@DeclareMathSizes</code> : use faster <code>\if</code> test	383	1996-07-27 ltspace.dtx v1.2q General: Further documentation of problems	279

1996-07-27 ltspace.dtx v1.2r		1996-10-21 lttab.dtx v1.1i	
General: Correct documentation of		<code>\@array</code> : Use <code>\set@typeset@protect</code>	658
problems	279	General: Moved the code associated	
1996-08-02 ltfloat.dtx v1.1o		with <code>\@mkpream</code> into the group	
<code>\@xympar</code> : Remove <code>\global</code> before		provided by the box, for	
<code>\@ignore</code>	733	robustness (latex/2183)	657
1996-08-02 ltsect.dtx v1.0v		<code>\multicolumn</code> : Make <code>\multicolumn</code>	
<code>\@afterheading</code> : Removed <code>\global</code>		long (latex/2180)	660
before <code>\@nobreak</code>	713	<code>\tabbing</code> : Moved the <code>\indent</code> so that	
1996-08-02 ltspace.dtx v1.2s		the <code>\everypar</code> can remove it when	
<code>\@Esphack</code> : Remove <code>\global</code> before		necessary; this is needed because	
<code>\@ignore</code>	288	the code for items in lists has	
1996-08-25 ltfssbas.dtx v3.0r		changed (see pr/22111)	652
<code>\nfss@catcodes</code> : Reset the acute,		1996-10-23 ltlists.dtx v1.0m	
grave and double quote chars as		<code>\@item</code> : <code>\@nobreak</code> moved into the	
well	391	<code>\everypar</code> and not executed	
1996-09-21 ltoutput.dtx v1.1w		unconditionally, see above	625
<code>\@writesetup</code> : Added		<code>\kern</code> changed to <code>\setbox</code>	625
<code>\@parboxrestore</code> and made		Added setting of <code>\clubpenalty</code> and	
consequent deletions: wait for the		set <code>\@nobreakfalse</code> only when	
howls of protest	879	necessary	625
1996-09-25 ltdirchk.dtx v1.0t		1996-10-23 ltsect.dtx v1.0x	
General: Move ltxcheck to separate file	13	<code>\@xsect</code> : Replaced <code>\hskip</code> with	
1996-09-28 ltmiscen.dtx v1.1f		<code>\setbox</code> as used in	
<code>\@xobeysp</code> : Moved to ltspace.dtx	592	<code>\@afterheading</code>	711
1996-09-28 ltspace.dtx v1.2t		1996-10-24 ltboxes.dtx v1.1a	
<code>\@xobeysp</code> : Moved from ltmiscen.dtx		<code>\@arrayparboxrestore</code> : Added local	
and redefined to use		settings of flags: dangerous!	638
<code>\nobreakspace</code>	295	<code>\@iiiminipage</code> : Use it or lose it	
1996-09-29 ltfiles.dtx v1.1g		(<code>@setminpage</code>): Frank will want to	
<code>\document</code> : Added disabling of		lose it	640
<code>\@nodocument</code>	303	1996-10-24 ltfloat.dtx v1.1p	
1996-09-29 ltoutput.dtx v1.1x		<code>\@floatboxreset</code> : Added local	
<code>\newpage</code> : Checks for noskipsec and		settings of flags: dangerous!	726
inlabel added	866	<code>\@marginparreset</code> : Added local	
1996-09-29 ltsect.dtx 1.0w		settings of flags: dangerous!	732
<code>\@noskipsectrue</code> : Added		<code>\@xfloat</code> : Added <code>\@nodocument</code> to	
documentation	707	trap floats in the preamble	723
1996-09-30 ltoutput.dtx v1.1y		1996-10-24 ltoutput.dtx v1.1z	
<code>\newpage</code> : Checks for noskipsec and		<code>\@addtocurcol</code> : Added <code>\nobreak</code> , etc	
inlabel removed pending further		as appropriate	893, 897
tests	866	<code>\@specialoutput</code> : Added <code>\nobreak</code> as	
1996-10-04 ltclass.dtx v1.0v		appropriate	872
<code>\RequirePackageWithOptions</code> : Reset		<code>\@topnewpage</code> : Added <code>\@nodocument</code>	
<code>\@unprocessedoptions</code> for /2269	774	to trap <code>\twocolumn</code> in the	
1996-10-05 ltfiles.dtx v1.1h		preamble	868
<code>\@clubpenalty</code> : Added setting its		<code>\newpage</code> : Better checks for noskipsec	
value	301	and inlabel added, plus nobreak	866
1996-10-08 ltfntcmd.dtx v3.3u		1996-10-25 ltlists.dtx v1.0n	
<code>\DeclareTextFontCommand</code> : Removed		<code>\endtrivlist</code> : Change <code>\indent</code> to	
<code>\check@icr</code> when in vmode since		<code>\leavevmode</code>	622
it causes various errors (see		Reset flags explicitly	622
pr/2157)	532	1996-10-25 ltoutput.dtx v1.2a	
		<code>\newpage</code> : Reset all flags explicitly	866

1996-10-26 ltlists.dtx v1.0o	1996-11-18 ltoutenc.dtx v1.8d
<code>\endtrivlist</code> : Correct typo 622	General: (DPC) lowercase external file
1996-10-27 ltoutenc.dtx v1.8c	names. internal/1044 363
<code>\@strip@args</code> : Removed macro . . . 327	1996-11-20 fontdef.dtx v2.2p
General: Added <code>\r A</code> 337	General: lowercase fd and enc.def file
Added	names /1044 506
<code>\textasteriskcentered</code> . . 333, 344	1996-11-20 ltvers.dtx v1.0f
Corrected syntax descriptions . . . 321	General: Check for old format
Removed <code>\aa</code> and <code>\AA</code> . . 333, 337, 340	modified /2319 36
1996-10-28 ltplain.dtx v1.1u	1996-11-23 ltoutenc.dtx v1.8e
General: (CAR) More doc changes . . 14	General: Corrected description 321
<code>\dotfill</code> : Removed math mode 31	Extended description 322
1996-10-29 ltplain.dtx v1.1v	1996-11-28 ltvers.dtx v1.0g
<code>\dotfill</code> : Got arithmetic correct	General: Check for old format
(CAR) 31	modified /2319 36
1996-10-29 ltspace.dtx v1.2u	1996-12-06 ltdirchk.dtx v1.0u
<code>\@gnewline</code> : Added macro 285	<code>\IfFileExists</code> : *** removed from
<code>\@no@lnbk</code> : Macro replaces <code>\@lnbk</code> and	various messages for GNU Make.
<code>\@no@lnbk</code> 283	internal/2338 10
<code>\:</code> : Corrected and rationalised code 283	1996-12-06 ltfloat.dtx v1.1r
<code>\nolinebreak</code> : Reimplemented both	<code>\@caption</code> : Call <code>\@setminpage</code> if
using <code>\@no@lnbk</code> 282	needed. latex/2318 722
1996-10-31 ltfinal.dtx v1.0z	1996-12-06 ltssini.dtx v3.0h
General: Added extra <code>\lcode</code> , hoping	General: (DPC) Remove *** from
it does no harm in T1	messages internal/2338 504
(pr/1969) 933, 940	1996-12-17 ltdefs.dtx v1.0w
1996-10-31 ltlists.dtx v1.0p	<code>\g@addto@macro</code> : Use <code>\begingroup</code> to
<code>\@trivlist</code> : Added check for missing	save making a mathord 102
item in outer list 621	1996-12-20 ltsect.dtx v1.0z
1996-10-31 ltsect.dtx v1.0y	<code>\@dottedtocline</code> : Added <code>\nobreak</code> for
General: Corrected and tidied	latex/2343 717
documentation; removed long	1997-01-08 fontdef.dtx v2.2q
lines 706	General: Use <code>\DeclareMathDelimiter</code>
1996-11-03 ltplain.dtx v1.1w	to set delimiter codes 515
<code>\dotfill</code> : Saved tokens by using	<code>\mathparagraph</code> : Define using
<code>\hb@xt@</code> 31	<code>\DeclareMathSymbol</code> 524
1996-11-04 ltterror.dtx v1.2m	1997-01-08 ltfiles.dtx v1.1j
<code>\@nodocument</code> : Always define	<code>\@include</code> : reset <code>\deadcycles</code>
<code>\@nodocument</code> in kernel, so that it	latex/2365 309
can be cleared by <code>\document</code> 258	1997-01-08 ltmath.dtx v1.1d
1996-11-04 ltlists.dtx v1.0q	<code>\root</code> : (DPC) Remove spurious space
<code>\@trivlist</code> : Moved check for missing	tokens from plain T _E X definition
item: only checked when not	/2359 600
inlabel flag is false 621	1997-02-05 ltdefs.dtx v1.0x
1996-11-05 ltfiles.dtx v1.1i	<code>\g@addto@macro</code> : missing percent
<code>\nofiles</code> : Standard <code>\if@nobreak</code> test	/2402 102
added 306	1997-02-21 ltlists.dtx v1.0r
1996-11-09 ltmath.dtx v1.1c	<code>\@item</code> : <code>\ifvoid</code> check added for
<code>\@ensuredmath</code> : Made long, as it was	<code>\noindent</code> . latex/2414 625
before. /2104 610	1997-03-21 ltcounts.dtx v1.1e
1996-11-18 ltssbas.dtx v3.0s	<code>\fnsymbol</code> : Use <code>\mathsection</code> and
<code>\define@newfont</code> : (DPC) lowercase fd	<code>\mathparagraph</code> . latex/2445 . . . 370
file names. internal/1044 390	

1997-04-14 ltfiles.dtx v1.1k	1997-08-29 ltoutenc.dtx v1.9f
<code>\document</code> : Set the document space	General: Added OT4 encoding,
factor defaults. latex/2404 302	provided by Marcin Woliński. . . 320
<code>\normalsfcodes</code> : Macro added (from	1997-09-09 ltdefns.dtx v1.2z
patch file) latex/2404 306	<code>\providecommand</code> : Use <code>\begingroup</code>
1997-04-14 ltoutput.dtx v1.2b	to avoid generating math ords if
<code>\@writsetup</code> : Call <code>\normalsfcodes</code>	used in math mode. pr/2573 81
(from patch file) latex/2404 881	1997-09-15 ltpictur.dtx v1.1g
Move <code>\label</code> and <code>\index</code> (from	<code>\@getcirc</code> : Warn if lines become
patch file) 881	invisible pr/2524 691
1997-04-24 ltbibl.dtx v1.1m	<code>\@picture@warn</code> : Macro added
<code>\@citex</code> : <code>\@empty</code> to avoid primitive	pr/2524 691
error on empty cite keys.	<code>\@sline</code> : Warn if lines become
latex/2432 746	invisible pr/2524 680
1997-04-30 ltoutenc.dtx v1.9a	1997-10-06 ltcounts.dtx v1.1f
General: Changed <code>\textsc</code> to	<code>\@Roman</code> : Change <code>\@Roman</code> to be fully
<code>\scshape</code> 334	expandable, so that the result is
Introduced <code>\textcopyright</code> and	written properly to files. 371
modified <code>\copyright</code> 334	<code>\@slowromancap</code> : Macro added. . . . 371
Introduced <code>\textcopyright</code> and	1997-10-08 ltlogos.dtx v1.1h
modify <code>\copyright</code> 335	<code>\LaTeX</code> : Simplify macro (force loading
Modified <code>\textunderscore</code> ,	of suitable math fonts once). . . . 298
removing <code>\mathunderscore</code> 334	1997-10-10 ltclass.dtx v1.0y
Modified <code>\underscore</code> , removing	<code>\endfilecontents</code> : <code>\@currenvir</code> in
<code>\mathunderscore</code> 335	banner 788
1997-04-30 ltoutenc.dtx v1.9b	<code>\reserved@e</code> not <code>\verbatim@out</code> to
General: Added <code>\leavevmode</code> to	save a csname 787
<code>\textunderscore</code> 334	Check for text before or after <code>\end</code>
1997-05-04 ltoutenc.dtx v1.9c	environment. latex/2636 788
General: Added ‘hex index tabs’ . . . 341	Use <code>\@gobbletwo</code> 787
Added TS1 encoding v2.2.beta . . . 347	1997-10-17 ltfntcmd.dtx v3.3w
1997-05-07 ltoutenc.dtx v1.9d	<code>\check@nocorr@</code> : Check for vertical
General: Added <code>\leavevmode</code> to	mode moved here, from
<code>\textcompwordmark</code> 334	<code>\DeclareTextFontCommand</code> (see
1997-05-07 ltspace.dtx v1.2v	PR/2646). 534
<code>\newline</code> : Made completely robust. . 284	<code>\DeclareTextFontCommand</code> :
1997-05-29 ltfsssrc.dtx v3.0j	Reinstalled <code>\check@icr</code> as check is
General: Replaced <code>\@</code> by	now done in <code>\check@nocorr@</code> (see
<code>\MessageBreak</code> , as suggested by	PR/2646). 532
Donald Arseneau. 426	1997-10-20 ltfinal.dtx v1.1a
1997-05-29 ltlogos.dtx v1.1f	<code>\@uclclist</code> : Removed <code>\aa</code> and <code>\AA</code>
<code>\LaTeXe</code> : Added <code>\m@th</code> so that the	from <code>\@uclclist</code> as these are
$\text{\LaTeX} 2_{\epsilon}$ logo works with non-zero	macros. 940
values of <code>\mathsurround</code> 298	1997-10-21 ltdefns.dtx v1.2z1
1997-06-16 ltdirchk.dtx v1.0v	<code>\renewcommand</code> : Use
General: documentation improvements	<code>\begingroup</code> / <code>\endgroup</code> rather
mainly from internal/2520 1	than braces for grouping, to avoid
1997-06-16 ltfloat.dtx v1.1s	generating empty math atom. . . . 79
General: documentation fixes 719	1997-10-21 ltfssbas.dtx v3.0t
1997-06-16 ltfntcmd.dtx v3.3v	<code>\define@newfont</code> : Move
General: Fix typo in documentation. . 530	<code>\makeatletter</code> to
1997-08-05 ltoutenc.dtx v1.9e	<code>\nfss@catcodes</code> 390
General: Corrected order of arguments	
in <code>\UseTextSymbol</code> example. . . . 321	

<code>\nfss@catcodes</code> : Moved	Removed default settings, see next section.	347
<code>\makeatletter</code> from		
<code>\try@load@font@shape</code>	1997-12-19 ltoutenc.dtx v1.9i	
1997-11-09 ltoutput.dtx v1.2c	General: Documentation corrections.	320
<code>\@specialoutput</code> : Remove incorrect code: only one <code>\@emptycol</code> is needed here	1997-12-20 fontdef.dtx v2.2s	
	General: Added documentation . . .	508
<code>\@topnewpage</code> : Documentation of <code>vsize</code> check enhanced	1997-12-31 ltoutenc.dtx v1.9k	
	General: Further correction	321
1997-11-13 ltfssdcl.dtx v3.0f	1998-01-12 ltoutenc.dtx v1.9k	
<code>\DeclareSymbolFont</code> : (DPC) Really update <code>\group@list</code> don't leave new version in <code>\toks@. latex/2661</code>	General: Added <code>\ProvidesPackage</code> for <code>textcomp.sty</code>	320
	Adding missing braces and <code>\ushape</code>	349
<code>\stepcounter</code> : (DPC) Remove as never used. (Re)defined in <code>ltxcounts</code>	1998-01-16 ltoutenc.dtx v1.9m	
	General: fixed decimal codes. <code>latex/2734</code>	345
1997-11-19 ltfloat.dtx v1.1t	1998-03-04 ltdefns.dtx v1.2z2	
<code>\@footnotetext</code> : Missing percent, again	<code>\@xargdef</code> : Unnecessary <code>\expandafter</code> removed: <code>pr/2758</code> .	77
1997-11-19 ltoutput.dtx v1.2d	1998-03-05 ltoutenc.dtx v1.9n	
<code>\@vtryfc</code> : Reindent code, to be understandable(DPC).	General: Added masc/fem ords as in <code>pr/2579</code>	334
1997-11-20 ltfssdcl.dtx v3.0g	1998-03-20 ltdefns.dtx v1.2z3	
<code>\document@select@group</code> : (DPC) inline use of <code>\stepcounter</code> (faster, and saves a <code>cname</code> per math version as no reset list)	<code>\@thirdofthree</code> : Macro added	82
	1998-03-20 ltoutenc.dtx v1.9o	
<code>\select@group</code> : (DPC) inline use of <code>\stepcounter</code> (faster, and saves a <code>cname</code> per math version as no reset list)	General: Documentation added about order of decls	323
	Documentation added for <code>pr/2783</code>	322
1997-11-23 ltoutenc.dtx v1.9g	<code>\UndeclareTextCommand</code> : Macro added for <code>pr/2783</code>	331
General: Use <code>\textperthousand</code> , <code>\textpertenthousand</code> and <code>\textfractionsolidus</code> not <code>\textpermill</code> , <code>\textpertenmill</code> and <code>\textfraction</code> . /2673	1998-03-20 lttextcomp.dtx v1.9o	
	General: Added various <code>\UndeclareTextCommand</code> declarations for <code>pr/2783</code>	568
1997-12-17 ltoutenc.dtx v1.9h	Load decls after defaults for speed.	568
General: Added <code>\textperthousand</code> and <code>\textpertenthousand</code>	1998-03-21 ltclass.dtx v1.0z	
Added code for <code>textcomp.sty</code>	General: Added to documentation of <code>filecontents</code>	754
Added section.	1998-03-21 ltclass.dtx v1.1a	
Added <code>textcomp.sty</code>	<code>\@providesfile</code> : Allow <code>&</code> . <code>Internal/2702</code>	767
As in OT1, Added <code>\leavevmode</code> at start of <code>\c</code> , otherwise the output routine might be invoked within the macro.	General: Correct to new <code>onlypreamble</code> command list	802
	1998-03-25 ltfssbas.dtx v3.0u	
Changed to decimal codes in <code>\oalign</code>	<code>\showhyphens</code> : Suppress unnecessary error when used in preamble . . .	399
Changed to decimal codes.	1998-04-11 fontdef.dtx v2.2t	
Documentation changes and additions.	General: Added <code>\mathring</code> accent (<code>pr2785</code>)	522
Example corrected, braces removed.	1998-04-15 fontdef.dtx v2.2u	
	General: Use new syntax for <code>\DeclareMathDelimiter</code>	515

1998-04-15 ltffssdcl.dtx v3.0h	1998-08-17 ltfnctcmd.dtx v3.3x
<code>\@xxDeclareMathDelimiter</code> : Macro	General: (RmS) Minor documentation
added (pr/2662) 473	fixes. 530
1998-04-17 fontdef.dtx v2.2v	1998-08-17 ltffssbas.dtx v3.0v
General: Reinsert symbol defs for <	General: (RmS) Documentation fixes. 376
and > chars. 516	1998-08-17 ltffssdcl.dtx v3.0i
1998-04-18 fontdef.dtx v2.2w	General: (RmS) Corrected minor
General: Reinsert symbol def for /	glitches in changes entries. 454
char. 516	1998-08-17 ltffssini.dtx v3.0i
1998-05-07 ltclass.dtx v1.1b	General: (RmS) Minor documentation
<code>\load@onefilewithoptions</code> : Modify	fixes. 481
help message for latex/2805 779	1998-08-17 ltlogos.dtx v1.1i
1998-05-18 lttab.dtx v1.1j	General: (RmS) Minor documentation
<code>\@endpbox</code> : Use <code>\setlength</code> to set	fixes. 298
<code>\hsize</code> , so that the changes in the	1998-08-17 ltmath.dtx v1.1c
calc package apply here. 668	General: (RmS) Minor documentation
<code>\tabular*</code> : Use <code>\setlength</code> , so that	fixes. 598
calc extensions apply. 657	1998-08-17 ltmiscen.dtx v1.1g
1998-05-20 ltfinal.dtx v1.1b	General: (RmS) Minor documentation
General: Set up lccodes before loading	fixes. 577
hyphenation files: pr/2639 932	1998-08-17 ltspc.dtx v1.2w
Set up uc/lccodes after loading	General: Documentation fixes. 279
hyphenation files: pr/2639 939	1998-08-17 preload.dtx v2.1g
1998-05-28 lterror.dtx v1.2n	General: (RmS) Minor documentation
<code>\@notdefinable</code> : Added message re	fixes. 527
‘end...’ pr/1555 258	1998-09-19 ltoutenc.dtx v1.9r
1998-06-04 ltboxes.dtx v1.1c	<code>\a</code> : Added <code>\string</code> (pr/2878) 332
<code>\@rule</code> : Support calc-expressions . . 642	1998-11-13 lttab.dtx v1.1m
1998-06-12 ltoutenc.dtx v1.9p	<code>\@array</code> : Check for hmode to see if
General: Corrected 130 and 131, see	something went wrong during
pr/2834 350	parsing (pr/2884) 658
Renamed <code>\textmacron</code> pr/2840 . . 351	1999-01-05 fontdef.dtx v2.2x
1998-06-12 ltoutenc.dtx v1.9q	General: Need special protection for
<code>\add@accent</code> : Explicitly set	character > in <code>\changes</code> entry. . . 506
<code>\spacefactor</code> after <code>\accent</code>	1999-01-06 ltffssbas.dtx v3.0w
(pr/2877) 327	<code>\DeclareFontEncoding</code> : Added
1998-06-12 lttextcomp.dtx v1.9p	<code>\LastDeclaredEncoding</code> to
General: Renamed <code>\textmacron</code>	support cyrillic integration
pr/2840 564	(pr/2988) 380
1998-06-18 lttab.dtx v1.1k	<code>\LastDeclaredEncoding</code> : Added
General: Small addition to	<code>\LastDeclaredEncoding</code> to
documentation 645	support cyrillic integration
1998-07-06 lttab.dtx v1.1l	(pr/2988) 380
General: Small correction to	1999-01-06 ltoutenc.dtx v1.9r
documentation 645	<code>\@strip@args</code> : New impl for
1998-08-17 ltboxes.dtx v1.1e	latex/2930 329
General: (RmS) Minor Documentation	General: Minor documentation fix. . 349
fixes. 629	1999-01-06 ltoutput.dtx v1.2e
1998-08-17 ltclass.dtx v1.1c	<code>\@makecol</code> : Added negative vskip, as
General: (RmS) Minor documentation	when processing outputbox below:
fixes. 754	suggested by Fred Bartlett
1998-08-17 ltdirchk.dtx v1.0w	pr/2892 876
General: (RmS) Documentation	1999-01-07 ltdefns.dtx v1.3a
improvements. 1	<code>\@ifnextchar</code> : made long 97

<code>\@newenvb</code> : made long and brace optional arg. latex/2896	80	1999-06-12 ltoutenc.dtx v1.9v	General: Extend <code>\@uclclist</code> only once	364
<code>\@testopt</code> : made long and brace optional arg. latex/2896	77	1999-10-09 ltmath.dtx v1.1e	<code>\active@math@prime</code> : Macro added, see PR 3104.	604
1999-01-07 ltdefns.dtx v1.3b		<code>\prime@s</code> : Introduce <code>\active@math@prime</code>	605	
<code>\@ifnextchar</code> : extra <code>\long</code> . latex/2902	97	1999-10-09 ltoutput.dtx 1.2f	<code>\@activechar@info</code> : Reset definition of active prime character (used in math mode)	879
1999-01-07 ltoutenc.dtx v1.9r		1999-10-28 ltoutenc.dtx v1.9w	<code>\add@accent</code> : Give <code>\accent@spacefactor</code> a default definition (pr/3084)	327
General: Hackery to allow using fontenc several times	365	1999-12-08 ltoutenc.dtx v1.9x	General: Changed <code>\CYRRHOOK</code> and <code>\cyrrhook</code> to <code>\CYRRHK</code> and <code>\cyrrhk</code> as name changed in the cyrillic bundle for naming consistency with other “hook” glyphs.	363
Hackery to temp support cyrillic uc/lc	363	2000-01-07 ltmiscen.dtx v1.1h	<code>\@verbatim</code> : Disable hyphenation even if the font allows it.	593
1999-01-13 ltoutenc.dtx v1.9s		2000-01-15 ltpictur.dtx v1.1i	<code>\@upvector</code> : Removed space at end-of-line, CAR	683
<code>\@strip@args</code> : Simplified solution for latex/2930	329	2000-01-30 ltfntcmd.dtx v3.3y	<code>\DeclareTextFontCommand</code> : Use <code>\hmode@bgroup</code> now (pr/3160)	532
1999-01-18 ltdefns.dtx v1.3c		2000-01-30 ltoutenc.dtx v1.9y	General: Use <code>\hmode@bgroup</code> where applicable (pr/3160)	336–339, 344–347, 349
<code>\@yargd@f</code> : New implementation DPC /2942	78	<code>\add@accent</code> : Use <code>\hmode@bgroup</code> where applicable (pr/3160)	326	
1999-02-09 ltdefns.dtx v1.3d		<code>\hmode@bgroup</code> : Macro added	327	
<code>\@yargd@f</code> : catch bad argument forms by re-inserting #3	78	2000-01-30 ltoutenc.dtx v1.9z	<code>\@use@text@encoding</code> : Macro reimplemented (pr/3160)	329, 330
1999-02-12 lttextcomp.dtx v3.0j		<code>\add@accent</code> : Macro reimplemented (pr/3160)	326	
<code>\legacyoldstylenums</code> : Use <code>\rmdefault</code> instead of <code>cmm</code> (pr/2954)	539	<code>\hmode@start@before@group</code> : Macro added (pr/3160)	330	
1999-02-24 ltoutenc.dtx v1.9t		2000-05-19 ltmiscen.dtx v1.1i	<code>\enddocument</code> : Reset <code>\AtEndDocument</code> for latex/3060	578
General: Corrected hackery cyrillic uc/lc list	363	2000-05-26 ltpage.dtx v1.0j	<code>\@markright</code> : Reimplementation to fix expansion error (pr/3203).	752
1999-03-01 ltdefns.dtx v1.3e		<code>\leftmark</code> : Use <code>\@empty</code> instead of brace group (pr/3203).	752	
<code>\@ifnextchar</code> : remove extra <code>\long</code> . internal/2967	97			
1999-04-15 ltpictur.dtx v1.1h				
<code>\@getlarrow</code> : Replaced octal number, CAR	682			
<code>\@upvector</code> : Replaced octal number, CAR	683			
General: Replaced octal number, CAR	682, 683			
Replaced octal numbers, CAR	670			
1999-04-19 ltfloat.dtx v1.1u				
<code>\caption</code> : Made caption an error outside a float: latex/2815	722			
1999-04-27 ltboxes.dtx v1.1f				
<code>\@parboxto</code> : (CAR) Changed <code>\@empty</code> to <code>\relax</code> as flag for natural width: pr/2975	638			
1999-04-29 ltdefns.dtx v1.3f				
<code>\@yargd@f</code> : Full expansion and conversion needed for digit in new version, see pr/3013	78			
New macro added	78			
1999-06-10 ltoutenc.dtx v1.9u				
General: Ensure that we also forget old options (pr/2888)	365			

<code>\markright</code> : Reimplementation to fix expansion error (pr/3203).	751	(pr/3334)	767
<code>\rightmark</code> : Use <code>\@empty</code> instead of brace group (pr/3203).	752	2001-05-25 <code>ltdirchk.dtx</code> v1.0x	
2000-06-02 <code>ltpage.dtx</code> v1.0k		General: Explicitly set catcode of <code>\endlinechar</code> to 10 (pr/3334) . . .	4
<code>\@markright</code> : Small adjustment to give slightly less expansion, CAR	752	2001-05-28 <code>ltoutenc.dtx</code> v1.93	
<code>\markright</code> : Small adjustment to give slightly less expansion, CAR . . .	751	General: Added composites for compatibility with T1, pr/3295 .	338
Tidied 1.0j reimplementation, CAR	751	Changed the effect of <code>\.i</code> , pr/3295	341
2000-07-11 <code>ltmiscen.dtx</code> v1.1j		2001-06-02 <code>fontdef.dtx</code> v2.2y	
<code>\@enddocument@kernel@warnings</code> : Fix typo in warning	579	General: Provide default cfg files (pr/3264)	525
2000-07-12 <code>ltoutput.dtx</code> 1.2g		2001-06-04 <code>fontdef.dtx</code> v2.2z	
General: Ensure that rule is in <code>\normalcolor</code>	919	General: Guard against math active equal and pipe sign in <code>\models</code> (pr/3333)	521
2000-07-12 <code>ltoutput.dtx</code> 1.2i		Guard against math active equal sign in <code>\Relbar</code> (pr/3333)	521
<code>\@makecol</code> : Removed negative vskip, as it gives unacceptable results when the depth is large: pr/3189	876	2001-06-04 <code>ltclass.dtx</code> v1.1e	
2000-07-19 <code>ltoutput.dtx</code> v1.2h		<code>\@providesfile</code> : But only if it is a char (pr/3334)	767
<code>\@writeseup</code> : Reset and restore <code>\@if@newlist</code> for internal/3231 .	880	2001-06-04 <code>ltdirchk.dtx</code> v1.0y	
2000-08-23 <code>ltfinal.dtx</code> v1.1c		General: But only if it is a char (pr/3334)	4
General: Fix typo in warning	934	2001-06-04 <code>ltpictur.dtx</code> v1.1j	
2000-08-30 <code>ltoutenc.dtx</code> v1.91		<code>\@sline</code> : Don't warn for exactly zero pr/3318	680
<code>\@use@text@encoding</code> : Rearranged but no change to final code, CAR (pr/3160)	329	2001-06-04 <code>ltvers.dtx</code> v1.0i	
<code>\add@accent</code> : Rearranged but no change to final code, CAR (pr/3160)	326	General: Check for old format disabled	36
2000-09-01 <code>ltfinal.dtx</code> v1.1d		2001-06-05 <code>ltoutenc.dtx</code> v1.94	
<code>\errhelp</code> : Set error help empty at very end (pr/449 done correctly).	942	General: Text composite Commands need kludges for ‘,’ – see <code>tlb1903.lvt</code>	338
2000-09-24 <code>ltfloat.dtx</code> v1.2b		2001-08-26 <code>ltclass.dtx</code> v1.1f	
<code>\end@dblfloat</code> : FMI: use output routine to defer float	727	<code>\@providesfile</code> : Readdded setting of space char (pr/3353)	767
2000-09-24 <code>ltoutput.dtx</code> v1.2b		2002-02-24 <code>ltplain.dtx</code> v1.1x	
<code>\doclearpage</code> : FMI: ensure <code>\doclearpage</code> is called again until all floats are output.	874	<code>\loggingall</code> : Macro added	32
2000-09-24 <code>ltoutput.dtx</code> v1.2n		<code>\loggingoutput</code> : Macro added	32
<code>\addtocurcol</code> : FMI: test for wide float was in wrong place	892	<code>\showoutput</code> : Use newly added <code>\loggingoutput</code>	32
2001-01-07 <code>ltoutput.dtx</code> v1.2j		<code>\tracingall</code> : Use newly added <code>\loggingoutput</code>	32
<code>\@writeseup</code> : And do it in the right macro (pr/3286)	880	2002-06-16 <code>ltoutenc.dtx</code> v1.95	
2001-02-16 <code>ltxref.dtx</code> v1.1k		General: Added <code>\textbardbl</code> (pr/3400)	344
<code>\@newl@bel</code> : Added an extra grouplevel (PR3250), jlb	574	Added default for <code>\textbardbl</code> (pr/3400)	333
2001-05-25 <code>ltclass.dtx</code> v1.1d		2002-06-17 <code>ltoutenc.dtx</code> v1.95	
<code>\@providesfile</code> : Explicitly set catcode of <code>\endlinechar</code> to 10		General: Corrected <code>\c</code> for T1 (pr/3442)	339
		Definition of <code>\textexclamdown</code> changed (pr/3368)	337

Definition of <code>\textquestiondown</code> changed (pr/3368)	337	2004-01-04 ltbibl.dtx v1.1p <code>\nocite</code> : Changed error message . .	748
2002-06-18 ltoutenc.dtx v1.95 General: Changed def for <code>\textregistered</code> to avoid small caps (pr/3420)	334	2004-01-04 ltoutenc.dtx v1.99c General: More adjustments for ogonek (pr/3532)	339
2002-10-01 ltfloat.dtx v1.1v <code>\thempfootnote</code> : Use braces around <code>\itshape</code> to keep font change local (pr/3460).	736	2004-01-23 ltdefns.dtx v1.1g <code>\@newenva</code> : Use kernel version of <code>\@ifnextchar</code> (pr/3501)	80
2002-10-02 ltffsbas.dtx v3.0x <code>\DeclareFontSubstitution</code> : Adding <code>\LastDeclaredEncoding</code> introduced a bug as on some occasions that macro name was stored in the internal lists instead of the actual encoding. (pr/3459)	380	<code>\@testopt</code> : Use kernel version of <code>\@ifnextchar</code> (pr/3501)	77
2002-10-28 ltlists.dtx v1.0s <code>\endtrivlist</code> : Check for math mode (pr/3437)	622	<code>\@xargdef</code> : Use kernel version of <code>\@ifnextchar</code> (pr/3501)	77
2002-10-28 ltoutenc.dtx v1.96 General: coding change, to follow bug fix by DEK in plain.tex (pr/3469)	337, 346	<code>\@xdblarg</code> : Use kernel version of <code>\@ifnextchar</code> (pr/3501)	99
2002-12-13 ltbibl.dtx v1.1n <code>\@citex</code> : Added <code>\leavevmode</code> in case citation is at start of paragraph (pr/3486)	746	2004-01-23 ltdefns.dtx v1.3g <code>\kernel@ifnextchar</code> : Added macro (pr/3501)	98
2003-01-01 ltfntcmd.dtx v3.3z General: Code checked and documentation extended by Chris	532	2004-01-28 ltclass.dtx v1.1g <code>\@providesfile</code> : Use kernel version of <code>\@ifnextchar</code> (pr/3501)	768
2003-05-18 ltbibl.dtx v1.1o <code>\nocite</code> : Check if we are after <code>\document</code>	748	2004-01-28 ltvers.dtx v1.0k General: Check for old format made 5 years (pr/3601)	36
2003-08-27 ltpictur.dtx v1.1k <code>\@bezier</code> : added missing displacement pr/3566	699	2004-02-02 fontdef.dtx v2.3 General: Many things from here on made robust	520
<code>\@sline</code> : check for <code>\@linechar</code> being empty pr/3570	680	2004-02-02 ltoutenc.dtx v1.99 General: Added <code>\textbigcircle</code> . .	344
2003-10-13 ltfinal.dtx v1.1e General: Added extra <code>\lccode</code> for <code>\-</code> and <code>\textcompwordmark</code>	933	2004-02-04 fontdef.dtx v2.3a General: Added bigtriangle synonyms for stmaryrd	518
2003-12-16 ltoutput.dtx v1.2k <code>\@makecol</code> : Ensure that <code>\@elt</code> has a defined state (pr/3586)	876	2004-02-04 ltspc.dtx v1.3 <code>\nobreakdashes</code> : (Macro added . . .	294
2003-12-30 ltpictur.dtx v1.1j <code>\@getcirc</code> : issue warning if circle size can't be met pr/3473	691	2004-02-06 ltoutenc.dtx v1.99d <code>\@inmathwarn</code> : New command added to fix severe bug: pr/3563	324
2004-01-03 ltoutenc.dtx v1.99b General: Added <code>\textogonekcentered</code> (pr/3532)	339	2004-02-07 ltoutput.dtx v1.2l <code>\@doclearpage</code> : Empty kludgeins box if necessary, pr/3528	874
Added composites for <code>\k</code> (pr/3532)	344	2004-02-13 ltoutenc.dtx v1.99e General: Documentation fixes: typos	320
Use <code>\ooalign</code> for <code>\k</code> (pr/3532)	339	2004-02-15 ltbibl.dtx v1.1q <code>\@cite@ofmt</code> : Added hook with default value <code>\hbox</code>	749
		<code>\@citex</code> : Changed to use a hook with default value <code>\hbox</code>	747
		2004-02-15 ltspc.dtx v1.3a <code>\nobreakdashes</code> : (Added spacefactor setting	294
		2004-10-20 ltoutput.dtx v1.2m <code>\@makecol</code> : Removed dead code . . .	876
		2005-07-27 ltssdcl.dtx v3.0j <code>\DeclareMathAlphabet</code> : (MH) Make document commands robust . . .	465

<code>\DeclareSymbolFontAlphabet:</code> (MH) Make document commands robust	478	<code>\t@st@ic:</code> Use switch <code>\ifmaybe@ic</code> instead of <code>\if@tempswa</code>	536
<code>\new@mathalphabet:</code> (MH) Make document commands robust	466	2010-08-17 ltmiscen.dtx v1.1k <code>\enddocument:</code> Use braces around <code>\input arg (pr/4124)</code>	579
<code>\non@alpherr:</code> (MH) Change because command is now properly robust	458	2010-08-17 ltmiscen.dtx v1.1l <code>\enddocument:</code> Change of plan: use <code>\@@input</code> instead (pr/4124)	579
<code>\SetMathAlphabet:</code> (MH) Make document commands robust	467	2011-05-08 ltfssdcl.dtx v3.0n <code>\in@:</code> Simplified thanks to Bruno.	454
2005-09-27 ltoutenc.dtx v1.99g General: Replace <code>\sh@ft</code> by <code>\ltx@sh@ft</code>	336, 339, 346	2011-08-19 ltclass.dtx v1.1i <code>\@ifclasswith:</code> Re-jig definition after more stringent <code>\in@</code> test.	765
2005-09-27 ltplain.dtx v1.1y <code>\ltx@sh@ft:</code> New macro	31	2011-09-03 ltfssdcl.dtx v3.0o <code>\new@mathversion:</code> (Will) Remove <code>\global</code> before <code>\newcount</code> (unnecessary and caused etex bug).	462
<code>\sh@ft:</code> Macro no longer used but left for compatibility	31	2012-01-20 ltplain.dtx v2.0b <code>\loggingall:</code> etex tracing if available	32
2005-11-08 ltoutenc.dtx v1.99h General: Added <code>\ij</code> and <code>\IJ</code> from babel. (pr/3771)	333, 338, 340	2013-07-07 ltclass.dtx v1.1i General: Correctly describe how the date in <code>\@ifpackagelater</code> is used	757
2005-11-10 ltmath.dtx v1.1g <code>\l:</code> (MH) Fixed potential problem in <code>\l</code> (pr/3399).	605	2014-04-18 ltoutput.dtx v1.1o General: Handle infinite glue from <code>\enlargethispage</code> (pr/4023)	920
General: (MH) Minor documentation fixes.	598	2014-04-24 ltoutput.dtx v1.2n <code>\fl@tracemessage:</code> Renamed internal trace commands; provide as package	910
2006-05-18 ltboxes.dtx v1.1g <code>\@parboxto:</code> Ensure <code>\@parboxto</code> holds the value of <code>\@tempdimb</code> not the register itself (pr/3867)	638	2014-04-27 ltfloating.dtx v1.2b <code>\end@dblfloat:</code> Inline the code to allow some coexistence with packages that hook into <code>\end@float</code> and do not know about the algorithm change	727
2006-09-13 ltoutput.dtx v1.1m General: Ensure that rule is in <code>\normalcolor</code>	920	2014-06-10 ltfloating.dtx v1.2b <code>\end@dblfloat:</code> missing <code>\fi</code> added	727
2007-08-05 ltclass.dtx v1.1h <code>\@fileswithoptions:</code> Prevent loss of brackets PR/3965	776	2014-12-30 ltfinal.dtx v2.0a <code>\newmarks:</code> macro added	927
2007-08-06 ltctrl.dtx v1.0h <code>\@fornoop:</code> Really make defs long	251	<code>\newXeTeXintercharclass:</code> macro added	927
2007-08-31 ltfssdcl.dtx v3.0l <code>\SetSymbolFont@:</code> Font warning changed to info for encoding change (pr/3975)	464	2014-12-30 ltfloating.dtx v1.2a <code>\@textsubscript:</code> Command added (latexrelease)	738
2009-09-24 ltvers.dtx v1.0l General: Stop checking for old format	36	<code>\textsubscript:</code> Command added (latexrelease)	737
2009-10-20 ltfssdcl.dtx v3.0m <code>\in@:</code> More robust thanks to Heiko.	454	2014-12-30 ltfssbas.dtx v3.0y <code>\mathgroup:</code> move allocation to ltplain.	376
2009-10-28 lttextcomp.dtx v1.99k General: Added Latin Modern and TeX Gyre subsets	569	2014-12-30 ltoutput.dtx v1.2m General: Command updated (latexrelease)	919
2009-11-04 lttextcomp.dtx v1.99l General: Added more Latin Modern and TeX Gyre subsets	569		
2009-12-14 ltfntcmd.dtx v3.4a <code>\ifmaybe@ic:</code> Macro added	535		
<code>\maybe@ic@:</code> Use switch <code>\ifmaybe@ic</code> instead of <code>\if@tempswa</code>	535		

2014-12-30 ltplain.dtx v2.0a			
\@alloc: macro added	19	\@stpelt: Reset all within counters in one go (latexrelease)	368
\@alloc@chardef: macro added	18	2015-01-11 ltcounts.dtx v1.1h	
\@alloc@top: macro added	18	\TextOrMath: Add command to solve robustness issues (pr/3752)	
\@ch@ck: macro added	19	(latexrelease)	372
\extrafloats: macro added	20	2015-01-11 ltfloat.dtx v1.2b	
\newlanguage: New engine-specific allocation scheme (latexrelease)	17	\@dblfloatplacement: float order in 2-column (latexrelease)	729
2014-12-30 ltspace.dtx v1.3b		\@xfloat: Check for valid option (latexrelease)	723
\@: \@ discards spaces when moving (pr3039)(latexrelease)	295	\end@dblfloat: float order in 2-column (latexrelease)	727
2015-01-03 ltdefs.dtx v1.4a		2015-01-11 ltfsbas.dtx v3.0y	
\typein: use modified definition in luatex	75	\@DeclareMathSizes: Allow arbitrary units (latexrelease)	383
2015-01-03 ltdirchk.dtx v1.1		2015-01-11 ltspace.dtx v1.3d	
General: Enable extra primitives when LuaTeX is used	3	\@Esphack: Allow hyphenation (Donald Arseneau pr/3498)	
2015-01-03 ltfinal.dtx v2.0a		(latexrelease)	288
General: Skip resetting codes with Unicode engines	939	\@esphack: Allow hyphenation (Donald Arseneau pr/3498)	
Unicode data loading added	930	(latexrelease)	287
2015-01-07 ltvers.dtx v1.0n		2015-01-14 ltoutput.dtx v1.2n	
\@check@IncludeInRelease: macro added	38	\@addtocurcol: float order in 2-column (latexrelease)	891
2015-01-08 ltboxes.dtx v1.1h		\@addtodblcol: float order in 2-column (latexrelease)	902
\framebox: Make Robust (latexrelease)	636	\@addtonextcol: float order in 2-column (latexrelease)	898
\makebox: Make Robust (latexrelease)	630	\@doclearpage: Empty kludgeins box if necessary, pr/3528	873
\parbox: Make Robust (latexrelease)	637	float order in 2-column (latexrelease)	873
\raisebox: Make Robust (latexrelease)	642	\@startdblcolumn: float order in 2-column (latexrelease)	885
\rule: Make Robust (latexrelease)	642	\@xtryfc: float order in 2-column (latexrelease)	887
\savebox: Make Robust (latexrelease)	633	\@ztryfc: float order in 2-column (latexrelease)	888
2015-01-08 ltdefs.dtx v1.4a		2015-01-14 ltspace.dtx v1.3e	
\MakeRobust: Added macro	85	\addpenalty: Avoid adding redundant skips (DPC)	291
2015-01-08 ltlength.dtx v1.1c		2015-01-17 ltvers.dtx v1.0m	
\setlength: to ensure first length argument is terminated. (latexrelease)	374	\@check@IncludeInRelease: modified with \@currname	38
2015-01-08 ltmath.dtx v1.1h		2015-01-19 ltvers.dtx v1.0o	
\): Make Robust (latexrelease)	605	\@check@IncludeInRelease: Optional argument	38
\]: Make Robust (latexrelease)	605	2015-01-20 ltoutput.dtx v1.2m	
2015-01-09 ltfsini.dtx v3.1a		\fl@tracemessage: Reset \IncludeInRelease flags	911
\em: Allow \emph to produce small caps (latexrelease)	498		
\eminershape: macro added (latexrelease)	498		
2015-01-09 ltspace.dtx v1.1h			
\addpenalty: Donald Arseneau's fix from PR/377703 (latexrelease)	291		
2015-01-10 ltcounts.dtx v1.1h			
\@fnsymbol: Unse \TextOrMath (latexrelease)	371		

2015-01-22 ltvers.dtx v1.0p	2015-02-21 ltplain.dtx v2.0e
General: Preserve any <code>\everyjob</code>	General: Removed autoload code . . . 14
material inserted by a loader (<code>.ini</code>	2015-02-21 lttab.dtx v1.1n
file) 37	General: Removed autoload code . . . 645
2015-01-23 ltfinal.dtx v2.0b	2015-02-21 ltvers.dtx v1.0r
<code>\newmarks</code> : use reserved count 256 . . 927	General: Removed autoload code . . . 36
<code>\newXeTeXintercharclass</code> : use	2015-02-21 ltvers.dtx v1.0w
reserved count 257 927	<code>\@check@IncludeInRelease</code> : set
2015-01-23 ltplain.dtx v2.0c	<code>\@currname</code> empty here (in case
<code>\extrafloats</code> : reserve counts 256–265 20	<code>\IncludeInRelease</code> input early) . 37
2015-01-24 ltfinal.dtx v2.0c	2015-02-22 ltfsscnp.dtx v3.0e
General: Skip T1-code entirely with	General: Moved all code into
Unicode engines 930	<code>latexrelease</code> - obsolete commands
2015-02-03 ltfinal.dtx v2.0d	are no longer automatically part of
General: Set <code>\lccode</code> for - with	the kernel 449
Unicode engines 932	2015-03-02 ltplain.dtx v2.0f
2015-02-16 ltoutenc.dtx v1.99m	<code>\e@mathgroup@top</code> : macro added . . . 19
General: Added <code>\textcommabelow</code>	<code>\newlanguage</code> : allow 255 math groups
<code>latex/4414</code> 335	in Unicode engines 17
2015-02-16 ltoutenc.dtx v1.99n	2015-03-10 ltplain.dtx v2.0g
General: Added <code>\textcommaabove</code> . . 336	<code>\hideoutput</code> : macro added 34
Added composites for <code>ç</code> 344	<code>\loggingall</code> : Reorganize to be less
Added composites for <code>\c</code> 338	noisy 32
2015-02-16 lttextcomp.dtx v1.99m	<code>\tracingnone</code> : macro added 33
General: Added <code>lmtt</code> (Heiko Oberdiek)	2015-03-12 ltoutput.dtx v1.2m
<code>latex/4415</code> 569	General: initialise <code>\@dbldeferlist</code>
2015-02-19 ltvers.dtx v1.0q	again 863
<code>\@check@IncludeInRelease</code> : Swap	2015-03-18 ltfssdcl.dtx v3.0q
argument order 38	<code>\DeclareSymbolFont</code> : Restrict Symbol
2015-02-20 ltplain.dtx v2.0d	fonts to 0–15 463
<code>\loggingall</code> : Spell commands	<code>\document@select@group</code> : Introduce
correctly :-)) 32	<code>\e@mathgroup@top</code> 459
2015-02-21 ltdefs.dtx v1.4b	<code>\select@group</code> : Introduce
General: Removed autoload support . 73	<code>\e@mathgroup@top</code> 457
2015-02-21 ltterror.dtx v1.2o	2015-03-26 ltfinal.dtx v2.0d
General: Removed autoload support 252	General: Use renamed
2015-02-21 ltfiles.dtx v1.1m	<code>unicode-letters.def</code> 930
General: Removed autoload support 299	2015-04-07 ltfssbas.dtx v3.1a
2015-02-21 ltfssbas.dtx v3.0z	<code>\wrong@fontshape</code> : Try loading fd file
General: Removed autoload code . . 376	if family has changed 394
2015-02-21 ltfsscnp.dtx v3.0d	2015-04-28 ltfinal.dtx v2.0f
General: Removed autoload code . . 449	<code>\newXeTeXintercharclass</code> : define
2015-02-21 ltfssdcl.dtx v3.0p	<code>\xe@alloc@intercharclass</code> for
General: Removed autoload code . . 454	compatibility with older xelatex
2015-02-21 ltfssstrc.dtx v3.0k	initialization 927
General: Removed autoload code . . 424	2015-05-10 ltlists.dtx v1.0t
2015-02-21 ltoutenc.dtx v1.99m	<code>\@doendpe</code> : Explicitly reset
General: Removed autoload code . . 320	<code>\clubpenalty</code> before clearing
2015-02-21 ltoutput.dtx v1.2n	<code>\everypar</code> ; see also pr/0462 and
General: Removed autoload code . . 852	pr/4065 623
<code>\f@depth</code> : macro added (<code>latexrelease</code>) 872	2015-06-19 ltfinal.dtx v2.0g
2015-02-21 ltpictur.dtx v1.1k	<code>\e@alloc@intercharclass@top</code> : Use
General: Removed autoload code . . 670	–1 for first range to get contiguous
	allocation 928

<code>\newmarks</code> : Use <code>-1</code> for first range to get contiguous allocation	927	<code>module_warning</code> : Function added . . .	54
2015-06-19 <code>ltpain.dtx</code> v2.0h		<code>modules</code> : Function modified	52
General: delete spurious old definition of <code>\newtoks</code>	23	<code>create_callback</code> : Function added . .	63
<code>\e@alloc</code> : extra braces in case arguments not single token	19	<code>provides_module</code> : Function added . .	53
<code>\newlanguage</code> : Use <code>-1</code> for first range to get contiguous allocation	17	<code>luatexbase</code> : Table added	52
2015-06-23 <code>ltfinal.dtx</code> v2.0h		2015-10-02 <code>ltdirchk.dtx</code> v1.2a	
General: set <code>\patch@level</code> in <code>ltvers</code> rather than in <code>ltfinal/ltpatch</code> . . .	941	General: Allow backing out of unprefixed names	3
2015-06-23 <code>ltvers.dtx</code> v1.0t		2015-10-02 <code>ltluatex.dtx</code> v1.0b	
General: set <code>\patch@level</code> in <code>ltvers</code> rather than in <code>ltfinal/ltpatch</code> . . .	36	General: Fix backing out of \TeX code	51
2015-08-06 <code>ltpain.dtx</code> v2.0i		2015-10-02 <code>ltluatex.dtx</code> v1.0c	
<code>\extrafloats</code> : Add <code>\string</code> in case argument is not an unexpandable primitive	20	General: Allow backing out of Lua code	51
2015-08-23 <code>ltdirchk.dtx</code> v1.2		2015-10-02 <code>ltluatex.dtx</code> v1.0e	
General: Do not use <code>luatex</code> prefix . . .	3	<code>uninstall</code> : Function added	66
2015-08-23 <code>ltvers.dtx</code> v1.0v		2015-10-03 <code>ltluatex.dtx</code> v1.0f	
General: Allow negative <code>patchlevel</code> for pre-release	37	<code>provides_module</code> : use <code>luatexbase__log</code>	53
2015-08-30 <code>ltpain.dtx</code> v2.1a		2015-10-27 <code>ltpain.dtx</code> v2.1b	
<code>\newinsert</code> : new <code>\newinsert</code> implementation	22	<code>\extrafloats</code> : Use global assignment when switching to extended range	20
2015-09-205 <code>ltoutput.dtx</code> v1.3a		2015-11-07 <code>ltspace.dtx</code> v1.3f	
General: extended <code>\@freelist</code>	862	<code>\@esphack</code> : Only space if there is no space at the end of the hlist <code>latex/4443</code>	287
2015-09-24 <code>ltluatex.dtx</code> v1.0a		2015-11-14 <code>ltluatex.dtx</code> v1.0g	
<code>call_callback</code> : Function added	63	General: Track Lua \TeX changes for (<code>new</code>) <code>token.create</code>	54
<code>callback.register</code> : Function modified	60	2015-11-18 <code>ltpain.dtx</code> v2.2a	
<code>callback_descriptions</code> : Function added	66	<code>\newlanguage</code> : Extended stream allocation in <code>luatex</code> (0.85)	17
<code>\catcodetable@atletter</code> : Macro added	48	2015-11-19 <code>ltpain.dtx</code> v2.2b	
<code>\catcodetable@initex</code> : Macro added	48	<code>\newlanguage</code> : Only extend allocation of write streams (see <code>luatex</code> list) .	17
<code>\catcodetable@latex</code> : Macro added .	48	2015-11-27 <code>ltluatex.dtx</code> v1.0h	
<code>\catcodetable@string</code> : Macro added	48	<code>callback_descriptions</code> : Match test in in-callback <code>latex/4445</code>	66
<code>add_to_callback</code> : Function added . .	64	<code>in_callback</code> : Guard against undefined list <code>latex/4445</code>	66
<code>remove_from_callback</code> : Function added	65	2015-11-29 <code>ltluatex.dtx</code> v1.0i	
<code>new_attribute</code> : Function added	55	General: Declare this as local before used in the module error definitions (PHG)	52
<code>disable_callback</code> : Function added .	66	<code>call_callback</code> : Check name is not nil in error message (PHG)	63
<code>in_callback</code> : Function added	66	<code>create_callback</code> : Check name is not nil in error message (PHG)	63
<code>\newattribute</code> : Macro added	47	2015-12-02 <code>ltluatex.dtx</code> v1.0j	
<code>\newcatcodetable</code> : Macro added . . .	47	General: Adjust <code>hashtokens</code> to store the result of <code>tex.hashtokens()</code> , not the function (PHG)	54
<code>\newluabytecode</code> : Macro added	50	Assorted typos fixed (PHG)	45
<code>\newluachunkname</code> : Macro added . . .	50	Declaration/use of <code>first_head</code> fixed (PHG)	53
<code>\newluafunction</code> : Macro added	49		
<code>\newwhatsit</code> : Macro added	50		
<code>module_error</code> : Function added	54		
<code>module_info</code> : Function added	54		

Remove nonlocal iteration variables (PHG)	45	2016-02-18 ltffssdcl.dtx v3.0r	
Remove unreachable code after calls to error() (PHG)	45	<code>\@DeclareMathDelimiter</code> : Check for delimiter not <code>\delimiter</code>	474
2015-12-02 ltluatex.dtx v1.0k		<code>\DeclareMathAccent</code> : Check for mathaccent not <code>\mathacemt</code>	469
General: resolve name and i.description (PHG)	61	<code>\DeclareMathRadical</code> : Check for radical not <code>\radical</code>	476
<code>call_callback</code> : Give more specific error messages (PHG)	63	<code>\DeclareMathSymbol</code> : Check for mathchar not <code>\mathchar</code>	471
<code>add_to_callback</code> : Give more specific error messages (PHG)	64	2016-03-13 ltluatex.dtx v1.0n	
<code>remove_from_callback</code> : adjust initialization of cb local (PHG)	65	General: contribute_ filter added	59
Give more specific error messages (PHG)	65	insert_ local_ par added	59
<code>create_callback</code> : Give more specific error messages (PHG)	63	2016-03-29 ltpictur.dtx v1.1l	
2015-12-10 ltfinal.dtx v2.0i		<code>\@oval</code> : add setting of line tests	692, 693
General: Use new common Unicode data loaders	930	initialise tests	692
2015-12-18 ltluatex.dtx v1.0l		<code>\@ovhorz</code> : use glue not leaders if horizontal line not required	695
General: Load Unicode data from source	48	<code>\@ovvert</code> : use glue not leaders if vertical line not required	694
2016-01-04 ltfinal.dtx v2.0j		<code>\if@ovhline</code> : macro added (latex/4452)	692
General: Do not set up inter character classes for XeTeX	930	<code>\if@ovvline</code> : macro added (latex/4452)	692
<code>\e@alloc@intercharclass@top</code> : Start allocation at one not three	927	2016-04-22 ltfinal.dtx v2.0q	
2016-01-05 ltfinal.dtx v2.0k		<code>\e@alloc@intercharclass@top</code> : XeTeX 0.99996 has 4096 char classes not 256	928
<code>\e@alloc@intercharclass@top</code> : Remove duplicated code	927	2016-06-19 ltoutenc.dtx v1.99m	
2016-01-05 ltfinal.dtx v2.0l		General: OT1 definition (was duplicate T1 definition)	338
General: Correct latexrelease guards	930	2016-06-20 ltclass.dtx v1.1j	
Ensure old definitions for inter-character class toks are available using latexrelease	930	General: don't declare as <code>\@onlypreamble</code>	764
Missing brace	930	2016-07-29 ltplain.dtx v2.2c	
2016-01-05 ltfinal.dtx v2.0m		<code>\extrafloats</code> : use <code>\global \chardef</code>	20
General: Undefine XeTeX classes when using patching an older kernel	930	<code>\newinsert</code> : fix for tlb-newinsert-001	22
2016-01-05 ltfinal.dtx v2.0p		2016-10-02 ltclass.dtx v1.2a	
General: Only apply XeTeX change if XeTeX is in use	930	<code>\@ifclasswith</code> : Ignore spaces while checking for option clash	765
2016-02-11 ltluatex.dtx v1.0m		<code>\ExecuteOptions</code> : Ignore spaces in argument	773
General:		2016-10-15 ltdirchk.dtx v1.2b	
pdf_stream_filter_callback removed	60	General: Require eTeX	4
process_rule, [hv]pack_quality append_to_vlist_filter added	59	2016-10-15 lterror.dtx v1.2p	
read_cidmap_file added	59	General: Require eTeX	252
show_warning_message added	59	2016-10-15 ltfinal.dtx v2.0r	
token_filter removed	59	General: Require eTeX	927
		2016-10-15 ltfinal.dtx v2.0s	
		General: Tidy up status of char 127	927
		2016-10-15 ltffssini.dtx v3.1b	
		General: Require eTeX	481
		2016-10-15 ltplain.dtx v2.2d	
		General: Require eTeX	14

2016-10-16 ltplain.dtx v2.3a	2017-02-19 ltoutenc.dtx v2.0f
<code>\newlanguage</code> : Allow languages up to 16383 in luatex	General: add <code>\@empty</code> to guard against 3rd argument being empty
2016-10-19 ltcounts.dtx v1.1j	declare composites with empty base for hat and tilde, use same slots for <code>\textasciicircum</code> ans <code>\textasciitilde</code>
<code>\TextOrMath</code> : Test directly for <code>\protected</code>	declare straight quotes using new <code>\remove@tlig</code> command
2016-11-06 ltplain.dtx v2.3b	2017-02-22 ltoutenc.dtx v2.0g
General: Drop <code>\outer</code> entirely	General: Fix typo introduced at 2.0f
2016-11-09 ltclass.dtx v2.1b	2017-02-24 ltoutenc.dtx v2.0h
<code>\@fileswithoptions</code> : Improve <code>\ifx</code> tests PR/4497	General: introduce <code>\DeclareUnicodeAccent</code>
2016-11-17 ltluatex.dtx v1.0p	<code>\DeclareTextCompositeCommand</code> : add check whether the accent command is defined for this encoding
2016-12-03 fontdef.dtx v3.0a	2017-03-08 ltclass.dtx v1.2c
General: (DPC) Default to TU encoding for Unicode TeX engines	General: add <code>\@parse@version@dash</code> to support yyyy-mm-dd as well as yyyy/mm/dd
<code>\shapedefault</code> : (DPC) Default to TU encoding for Unicode TeX engines	2017-03-09 ltfinal.dtx v2.0t
2016-12-04 ltoutenc.dtx v2.0a	<code>\l@nohyphenation</code> : ensure <code>\l@nohyphenation</code> is defined.
General: Added TU encoding	2017-03-09 ltmiscen.dtx v1.1m
2017-01-01 ltoutput.dtx v1.3b	<code>\@verbatim</code> : Use <code>\language</code> not <code>\hyphenchar</code>
General: make <code>fpmin</code> negative so ignored even if float height is negative	<code>\verb</code> : Use <code>\language</code> to stop hyphenation
2017-01-10 ltfsbas.dtx v3.2a	2017-03-10 ltfiles.dtx v1.1n
<code>\showhyphens</code> : Add version of <code>\showhyphens</code> that works with XeTeX.	<code>\document</code> : Save language default
2017-01-23 ltoutenc.dtx v2.0b	2017-03-10 ltoutput.dtx v1.3c
General: Added TU specific commands in ASCII range pr/4500	<code>\@writsetup</code> : Reset <code>\language</code>
2017-01-24 ltoutenc.dtx v2.0c	2017-03-13 ltdefns.dtx v1.5a
General: Declare TU composites for i and j	<code>\-</code> : Define <code>\-</code> in terms of <code>\hyphenchar</code>
Make <code>\textasteriskcentered</code> U+2217 not U+204E	2017-03-27 ltdefns.dtx v1.5b
TeX ligature syntax for xetex and luatex reversed	<code>\@dischyph</code> : Define <code>\@dischyph</code> after <code>\-</code>
2017-01-24 ltoutenc.dtx v2.0d	2017-03-28 ltluatex.dtx v1.1e
General: Declare macron composites for YyGg	General: <code>glyph_</code> <code>stream_</code> provider added
2017-02-12 ltoutenc.dtx v2.0e	2017-03-29 ltboxes.dtx v1.3a
General: Declare fallback code for <code>\textasteriskcentered</code>	<code>\@arrayparboxrestore</code> : Reset <code>\lineskiplimit</code>
2017-02-18 ltluatex.dtx v1.1c	2017-04-05 ltoutenc.dtx v2.0i
<code>new_attribute</code> : Parameterize count used in tracking	<code>\DeclareTextCompositeCommand</code> : Declare accent command if not already declared when declaring a composite.
<code>new_bytecode</code> : Parameterize count used in tracking	2017-04-10 ltplain.dtx v2.3c
<code>new_chunkname</code> : Parameterize count used in tracking	<code>\newlanguage</code> : Correction to code to skip write18 in luatex
<code>new_whatit</code> : Parameterize count used in tracking	

2017-04-11 ltoutput.dtx v2.4a	2018-05-08 ltclass.dtx v1.2i
\newpage: account for the depth of the last row of the page 866	\pkgcls@parse@date@arg: Make suspicious rollback a warning not error: github issue 43 797
2017-12-17 ltoutput.dtx v1.4b	2018-05-11 ltfinal.dtx v2.18
\@addtonextcol: fix doc guards . . . 898	General: Make invalid UTF-8 also safe, for legacy filesystem encodings . . 937
2018-01-06 ltdefs.dtx 1.5c	2018-05-29 ltclass.dtx v1.2j
\@ifundefined: Avoid defining undefined commands to \relax . . 96	\endfilecontents: use \curname not \@undefined 789
2018-02-18 ltclass.dtx v1.2d	2018-08-11 ltoutenc.dtx v2.0j
\@ifl@ter: Added 0 up front to make bad data come out as 0. 764	General: Provide \guillemetleft and \guillemetright 340, 346, 355
General: Introduce rollback concept 794	2018-08-18 ltluatex.dtx v1.1h
2018-03-08 ltcounts.dtx v1.1k	General: append_ to_ vlist_ filter is exclusive 59
\@ifbothcounters: Interface added 369	2018-08-24 ltfinal.dtx v1.1f
\@removefromreset: Interface added 368	\document@default@language: Add to latexrelease (github/68) 934
\counterwithin: Interface added . . 369	2018-09-02 ltsect.dtx v1.1b
\counterwithout: Interface added . . 369	\@dottedtocline: Prevent protrusion (https://tex.stackexchange.com/q/172785/10109) 717
2018-03-24 ltclass.dtx v1.2e	2018-09-24 fontdef.dtx v3.0b
\pkgcls@use@this@release: Use full file name for old release 799	General: Start LR-mode if necessary (git/49) 524
2018-03-25 ltfinal.dtx v2.1a	2018-09-24 ltmath.dtx v1.2b
General: default to UTF-8 935	\smash: Start LR-mode if necessary (git/49) 601
\UseRawInputEncoding: Macro added 936	\vphantom: Start LR-mode if necessary (git/49) 601
2018-03-27 ltclass.dtx v1.2f	2018-09-24 ltspace.dtx v1.3h
\endfilecontents: Use full file name for old release 788	\enspace: Start LR-mode if necessary (git/49) 296
2018-04-06 ltfinal.dtx v2.1b	\leavevmode@ifvmode: Macro added (git/49) 296
\UseRawInputEncoding: Undo changes to \DeclareFontEncoding@ and definition of \DeclareUnicodeCharacter 936	2018-09-26 ltdefs.dtx v1.5e
2018-04-07 ltfinal.dtx v2.1c	\renew@command: Always explicitly generate a space after the curname and not rely on \noexpand to save tokens (git/41) 79
\UseRawInputEncoding: Undefine \inputencodingname 936	2018-09-26 ltmiscen.dtx v1.1n
2018-04-08 ltclass.dtx v1.2g	\@writefile: Sometimes mask the endline char when writing to files (github/73) 583
\@ifl@ter: Strip leading spaces from dates. 764	\add@percent@to@temptokena: Sometimes mask the endline char when writing to files (github/73) 582
2018-04-08 ltclass.dtx v1.2h	\protected@file@percent: Sometimes mask the endline char when writing to files (github/73) 582
\@onefilewithoptions: Pass expanded date 795	2018-09-26 ltsect.dtx v1.1c
2018-04-08 ltfinal.dtx v2.1d	\addcontentsline: Sometimes mask the endline char when writing to
General: Delay full UTF-8 handling to \everyjob 937	
2018-04-11 ltcounts.dtx v1.1l	
\counterwithin: Correct default (issue/38) 369	
2018-05-02 ltluatex.dtx v1.1g	
General: find_ sfd_ file removed . . . 58	
finish_ synctex_ callback added . . 59	
glyph_ not_ found added 60	
read_ sfd_ file removed 58	

files (github/73)	715	2019-02-07 ltfsbas.dtx v3.2b	
2018-10-10 ltspace.dtx v1.3i		<code>\define@newfont</code> : Changed wording of warning (github/107)	390
<code>\@esphack</code> : Don't introduce breakpoints if <code>@nobreak</code> is true and after sections	287	2019-06-18 ltluatex.dtx v1.1j	
2018-10-11 ltmiscen.dtx v1.1o		General: finish_ synctex_ callback renamed finish_ synctex_	59
<code>\@sverb</code> : Provide visible space in <code>\verb*</code> also for XeTeX and LuaTeX (github/69)	595	font_ descriptor_ objnum_ - provider added	60
<code>\@setupverbvisiblespace</code> : Provide visible space in <code>\verb*</code> also for XeTeX and LuaTeX (github/69)	594	make_ extensible added	60
<code>\@verbvisiblespacebox</code> : Provide visible space in <code>\verb*</code> also for XeTeX and LuaTeX (github/69)	594	new_ graf added	59
<code>\asciispace</code> : Provide visible space in <code>\verb*</code> also for XeTeX and LuaTeX (github/69)	594	page_ objnum_ provider added	59
<code>verbatim*</code> : Provide visible space in <code>\verb*</code> also for XeTeX and LuaTeX (github/69)	595	process_ pdf_ image_ content added	59
<code>\verbvisiblespace</code> : Provide <code>\verbvisiblespace</code> such that it is usable in normal text (github/70)	594	wrapup_ run added	59
Provide visible space in <code>\verb*</code> also for XeTeX and LuaTeX (github/69)	594	2019-07-01 ltclass.dtx v1.3a	
2018-10-21 ltluatex.dtx v1.1i		<code>\endfilecontents</code> : Support UTF8 and spaces in filecontents environment file name	786
<code>new_luafunction</code> : Function added	57	2019-07-01 ltfiles.dtx v1.2a	
2018-11-09 ltbibl.dtx LaTeX2e		<code>\IfFileExists</code> : Support UTF-8	312
<code>\bibliography</code> : Zap spaces in the argument as BibTeX doesn't support them (github/88)	747	<code>\includeonly</code> : Support UTF-8	307
2018-11-18 ltoutenc.dtx v2.0k		<code>\set@curr@file</code> : Support UTF-8	311
General: Provide <code>\Hwithstroke</code> and <code>\hwithstroke</code>	357	2019-07-09 ltfsbas.dtx v3.2c	
2018-11-19 ltoutenc.dtx v2.0k		<code>\DeclareErrorFont</code> : Don't set any <code>\f@...</code> macros	393
General: Added <code>\Hwithstroke</code> and <code>\hwithstroke</code>	339	2019-07-09 ltfsini.dtx v3.1c	
2018-11-28 ltoutput.dtx v1.4d		General: Explicitly set some defaults	503
<code>\@combinedblfloats</code> : Unbox <code>\@outputbox</code> to preserve boxing level (github/94)	885	2019-08-22 ltxref.dtx v1.1l	
2018-12-30 lttab.dtx v1.1p		<code>\labelformat</code> : Commanded moved from <code>varioref.sty</code>	575
<code>\@tabclassz</code> : Add extra <code>\hskip</code> to guard against an <code>\unskip</code> at the start of a c-column cell (gh/102)	665	<code>\Ref</code> : Commanded moved from <code>varioref.sty</code>	575
2019-02-07 ltfilehook.dtx v1.1o		<code>\refstepcounter</code> : Allow <code>\p@...</code> to have an argument	575
<code>\unqu@tefilef@und</code> : Expand <code>\@filef@und</code> before executing second argument (github/109)	813	2019-08-27 fontdef.dtx v3.0c	
2019-02-07 ltfiles.dtx v1.1o		General: Various commands made robust throughout the file	517
<code>\swaptwoargs</code> : Helper macro added	315	2019-08-27 ltboxes.dtx v1.3b	
		General: Various commands made robust	629
		2019-08-27 ltclass.dtx v1.3b	
		<code>\endfilecontents</code> : Make various commands robust	786
		2019-08-27 ltdefs.dtx v1.5f	
		General: Make various commands robust	101
		<code>\MakeRobust</code> : Make the assignments global as we may need to apply them inside a group	85
		2019-08-27 ltfilehook.dtx v1.2b	
		<code>\unqu@tefilef@und</code> : Make command robust	813

2019-08-27 ltfiles.dtx v1.2b	2019-08-30 lterror.dtx v1.2q
\IfFileExists: Make command robust 312	\conditionally@traceoff: Macro added 261
2019-08-27 ltfssbas.dtx v3.2d	\conditionally@traceon: Macro added 261
General: Make various commands robust 376	2019-09-09 ltfssdcl.dtx v3.0s
2019-08-27 ltfssdcl.dtx v3.0s	\DeclareMathSymbol: Allow definition if the math symbol was a command already robust 471
\DeclareMathAccent: Make math accents robust 469	2019-09-11 ltclass.dtx v1.3c
\set@mathdelimiter: Make math delimiters robust 476	\endfilecontents: Support optional argument for filecontents 786
2019-08-27 ltfssini.dtx v3.1d	2019-09-14 ltfinal.dtx v2.1h
General: Make various commands robust 481	\euclclist: Expand UTF8 chars when case changing (github/177) 940
2019-08-27 ltidxglo.dtx v1.1f	2019-09-16 ltxref.dtx v1.1m
General: Make \index and \glossary robust 742	General: Correctly revert the \p@... change 575
2019-08-27 ltlength.dtx v1.1d	2019-09-21 fontdef.dtx v3.0d
General: Make various command robust 374	General: Distangle alias (gh/184) 519, 523
2019-08-27 ltlogos.dtx v1.1j	2019-10-02 ltexpl.dtx v0.0
\TeX: Make \TeX command robust 298	General: Initial version 68
2019-08-27 ltmath.dtx v1.2c	2019-10-02 ltfinal.dtx v2.2
General: Make various commands robust 598	General: Load ltexpl 942
2019-08-27 ltmiscen.dtx v1.1p	2019-10-02 ltluatex.dtx v1.1k
General: Make various commands robust 577	General: linebreak__filter is exclusive 59
\begin: Make command robust 584	mlist__to__hlist is exclusive 59
\end: Make command robust 587	process__rule is exclusive 59
2019-08-27 ltoutput.dtx v1.4e	2019-10-07 lttab.dtx v1.1q
\@begindvibox: Make \AtBeginDvi robust 864	\extracolsep: This needs to expand 657
2019-08-27 ltpage.dtx v1.0l	2019-10-11 ltfiles.dtx v1.2c
General: Make various commands robust 750	\set@curr@file: Remove one brace group 311
2019-08-27 ltpictur.dtx v1.1m	2019-10-11 ltfsstrc.dtx v3.0l
General: Make various commands robust 670	\@font@aliasinfo: Added 'alias' size function 447
Remove several unnecessary \gdef definitions 670	2019-10-18 ltclass.dtx v1.3d
2019-08-27 ltsect.dtx v1.1d	\load@onefilewithoptions: Initialize \...-h@@k only when loading the package or class (gh/198) 778
General: Make various commands robust 706	2019-10-22 ltluatex.dtx v1.1j
2019-08-27 ltspace.dtx v1.3j	General: page__objnum__provider and process__pdf__image__content classified data 59
General: Make various commands robust 279	2019-10-25 ltmiscen.dtx v1.1q
2019-08-27 lttab.dtx v1.1q	\add@percent@to@temptokena: Allow unbalanced conditionals in #1 (gh/202) 582
General: Make various commands robust 645	2019-10-26 ltfiles.dtx v1.2d
Remove several unnecessary \gdef definitions 645	\@iffilenamepath: quote on openin 314
	\IfFileExists: don't quote name 312
	\IfFileExists@: quote on openin 312
	\set@curr@file: remove quotes 311

2019-11-01 ltdirchk.dtx v1.3a		<code>\ttfamily</code> : Streamlined implementation with hook	492
<code>\filename@parse</code> : take last . not first	12		
2019-11-02 ltmiscen.dtx v1.1s		2020-01-20 ltfsdcl.dtx v3.0t	
<code>\@centercr</code> : Make <code>\@centercr</code> robust		<code>\set@mathdelimiter</code> : fix for gh/251	476
(gh/203)	589	2020-01-20 ltoutenc.dtx v2.0n	
2019-11-02 ltspace.dtx v1.3k		General: fix for gh/251	335
<code>\@normalcr</code> : Make also <code>\@normalcr</code>		2020-01-22 lttextcomp.dtx v1.0b	
robust	283	<code>\tc@subst</code> : The overall default is	
2019-11-09 ltfiles.dtx v1.2e		<code>\textcompsubstdefault</code> not	
<code>\set@curr@file</code> : expand and <code>\string</code>		<code>\substdefault</code>	542
before removing quotes	311	2020-01-25 fontdef.dtx v3.0f	
2019-11-10 ltmiscen.dtx v1.1r		General: Load <code>tlenc.def</code> last (gh/255)	508
<code>\add@percent@to@temptokena</code> : fix to		2020-01-25 ltoutenc.dtx v2.0m	
special comment catcodes		General: Load each encoding file only	
(gh/202)	582	once (gh/255)	363
2019-11-11 ltfiles.dtx v1.2f		2020-01-27 ltclass.dtx v1.3g	
<code>\@iffileonpath</code> : make <code>\@filef@und</code>		<code>\endfilecontents</code> : Fix typo in error	
match quoting used on <code>\openin</code> .	314	message	788
2019-11-22 ltoutenc.dtx v2.0l		2020-01-28 ltclass.dtx v1.3h	
General: Avoid spurious if fontenc		<code>\endfilecontents</code> : Allow spaces in	
selects LY1 as default encoding		option string and display only	
(gh/199)	364	unknown options not the whole	
2019-11-29 ltclass.dtx v1.3e		option list (gh/256)	786
<code>\@pr@videpackage</code> : Protect package		2020-01-31 ltvers.dtx v1.1e	
info text (gh/52)	766	General: Allow for upcoming format as	
2019-12-17 fontdef.dtx v3.0e		pre-release 0	37
<code>\mddefault</code> : Set <code>\bfdefault</code> to “b”	511	2020-02-02 ltluatex.dtx v1.1l	
<code>\shapedefault</code> : Set <code>\shapedefault</code>		General: Add reverselist callback type	61
explicitly to “n”	512	glyph_ info added	60
<code>\updefault</code> : Set <code>\updefault</code> to “up”	511	page_ order_ index added	59
2019-12-17 ltfntcmd.dtx v3.4c		post_ linebreak_ filter is	
<code>textssc</code> : Macro added	533	<code>reverselist</code>	59
2019-12-17 ltfsbas.dtx v3.2e		<code>create_callback</code> : Provide proper	
<code>\usefont</code> : Don’t call <code>\fontseries</code> or		fallbacks for user-defined callbacks	
<code>\fontshape</code>	385	without user-provided default	
2019-12-17 ltfsini.dtx v3.1e		handler	63
General: Provide custom series		2020-02-05 ltfsini.dtx v3.1g	
settings a la mweights	482	<code>\DeclareFontSeriesDefault</code> :	
<code>\DeclareEmphSequence</code> : Provide <code>\emph</code>		Clarified error text	483
sequences	497	Corrected misspelled csname	
2019-12-18 ltoutenc.dtx v2.0m		(gh/264)	483
General: Don’t fake		2020-02-05 lttextcomp.dtx v2.0n	
<code>\textcompwordmark</code> ; take default		General: Changed the package default	
from T1 instead	334	to info (gh/262)	559
<code>\add@accent</code> : Avoid code that breaks		Ensure we are on a new format	
<code>\accent</code>	327	(gh/260)	559
2019-12-21 fontdef.dtx v3.0e		2020-02-07 ltfsini.dtx v3.1h	
General: Distangle alias (gh/184)	517–520	<code>\symbol</code> : XeTeX-specific version to	
2020-01-05 ltclass.dtx v1.3f		avoid bug in maths mode.	500
<code>\endfilecontents</code> : Support more		2020-02-10 ltfsaxes.dtx v1.0c	
write streams in LuaTeX gh/238	786	<code>\fontseries</code> : Switch	
2020-01-11 ltfsini.dtx v3.1f		<code>\if@forced@series</code> added	412
<code>\rmfamily</code> : Streamlined		<code>\fontseriesforce</code> : Switch	
implementation with hook	492	<code>\if@forced@series</code> added	412

<code>\if@forced@series</code> : Switch	2020-02-25 ltfssini.dtx v3.1j
<code>\if@forced@series</code> added 412	<code>\bf@default</code> : Drop surplus “m” from
2020-02-10 ltfssini.dtx v3.1h	<code>\bfdefault</code> and <code>\mddefault</code>
<code>\@defaultfamilyhook</code> : Add	(gh/291) 488
<code>\@defaultfamilyhook</code> to	<code>\prepare@family@series@update</code> :
<code>\normalfont</code> (gh/269) 492	Drop surplus “m” from
<code>\reset@font</code> : Add	<code>\target@series@value</code> (gh/291) 485
<code>\@defaultfamilyhook</code> to	<code>\update@series@target@value</code> : Drop
<code>\normalfont</code> (gh/269) 501	surplus “m” from <code>\reserved@d</code>
2020-02-10 lttextcomp.dtx v1.0c	(gh/291) 486
General: Use <code>\@tabacckludge</code> for	2020-02-27 ltdefs.dtx v1.5g
tabbing where necessary (gh/271) 545	<code>\gobblethree</code> : Macro added 82
2020-02-11 fontdef.dtx v3.0g	2020-02-27 ltfssaxes.dtx v1.0d
General: Provide value for	<code>\series@maybe@drop@one@m</code> : Drop
<code>\@fontenc@load@list</code> (gh/273) . 508	“m” in certain values from a fixed
2020-02-11 ltfssini.dtx v3.1h	list (gh/293) 416
General: Provide default value for	<code>\set@target@series</code> : Drop “m” only
<code>\@fontenc@load@list</code> (gh/273) . 503	in a specific set of values (gh/293) 415
2020-02-11 ltoutenc.dtx v2.0o	2020-02-27 ltfssbas.dtx v3.2g
General: Update	<code>\DeclareFontShape@</code> : Only “m” if the
<code>\@fontenc@load@list</code> with option	series value is a member of a fixed
list (gh/273) 365	list and issue warning if doing it
2020-02-14 ltpictur.dtx v1.1n	(gh/293) 377
<code>\linethickness</code> : Suppress spaces	2020-03-02 ltexpl.dtx v1.0a
following the declaration (gh/274) 675	General: Don’t load expl3 if already in
2020-02-18 ltfssini.dtx v3.1i	the format (gh/295) 68
<code>\bfseries</code> : Make the <code>\ifx</code> selection	2020-03-05 ltexpl.dtx v1.1
outside of <code>\fontseries</code> argument	General: Load xparse.ltx if
so that it is not done several times 489	<code>\NewDocumentCommand</code> is not
<code>\mdseries</code> : Make the <code>\ifx</code> selection	defined by expl3.ltx 68
outside of <code>\fontseries</code> argument	2020-03-06 ltboxes.dtx v1.3c
so that it is not done several times 490	<code>\clap</code> : Macro <code>\clap</code> added 644
<code>\prepare@family@series@update</code> : No	2020-03-07 ltluatex.dtx v1.1m
series auto-update when forced	<code>remove_from_callback</code> : Do not call
(gh/277) 484	<code>callback.register</code> for user-defined
Recognize current family if it is not	callbacks 65
a “meta” family and auto-update	2020-03-07 ltmath.dtx v1.2e
series using <code>\bfdefault</code> (gh/277) 485	<code>\negthickspace</code> : Add <code>amsmath</code>
2020-02-18 ltmath.dtx v1.2d	<code>math/text</code> spacing commands to
<code>\mathindent</code> : Make <code>\mathindent</code> a	the kernel (gh/303) 603
skip register to match <code>amsmath</code>	2020-03-07 ltspc.dtx v1.3l
(gh/252) 610	General: Moved <code>\thinspace</code> ,
<code>equation</code> : Separate formula and eqn	<code>\negthinspace</code> and <code>\,</code> to
number by at least a space in <code>fleqn</code>	ltmath.dtx (gh/303) 279
option 611	2020-03-19 fontdef.dtx v3.0h
2020-02-20 ltclass.dtx v1.3j	General: Support legacy use of
<code>\endfilecontents</code> : Fix missing quotes	<code>\bfdefault</code> and <code>\mddefault</code>
around file name (gh/284) 788	(gh/306) 511
2020-02-24 ltfssbas.dtx v3.2f	2020-03-19 ltfssdcl.dtx v3.0u
<code>\DeclareFontShape@</code> : Drop surplus	<code>\document@select@group</code> : fix for
“m” in series when defining	(gnats/3357) 459
<code>fontshape</code> (gh/289) 377	2020-03-19 ltfssini.dtx v3.1k
	<code>\bfseries</code> : Support legacy use of
	<code>\bfdefault</code> and <code>\mddefault</code>

(gh/306)	489	<code>\addvspace</code> : Support calc syntax (gh/152)	290
<code>\DeclareFontSeriesDefault</code> : Support legacy use of <code>\bfdefault</code> and <code>\mddefault</code> (gh/306)	483	2020-04-21 <code>lttab.dtx</code> v1.1r <code>\@itabcr</code> : Support calc syntax (gh/152)	651
<code>\mdseries</code> : Support legacy use of <code>\bfdefault</code> and <code>\mddefault</code> (gh/306)	490	<code>\@yargarraycr</code> : Support calc syntax (gh/152)	659
2020-04-06 <code>ltfssini.dtx</code> v3.1m <code>\bf@def@ult</code> : Hook added (gh/306)	489	2020-04-22 <code>ltmiscen.dtx</code> v1.1u <code>\@sverb</code> : Drop spaces before <code>\verb</code> delimiter (gh/327)	595
<code>\bfseries</code> : Hook added (gh/306)	489	2020-04-22 <code>ltoutenc.dtx</code> v2.0p General: y unicode value in <code>tuenc.def</code>	320
<code>\mdseries</code> : Hook added (gh/306)	490	2020-04-29 <code>lttextcomp.dtx</code> v1.0d General: Make all capital accents text commands for <code>hyperref</code> (gh/332)	545
2020-04-07 <code>ltclass.dtx</code> v1.3k <code>\IfFormatAtLeastTF</code> : Macro added; also in rollback (gh/168)	763	2020-05-02 <code>ltfiles.dtx</code> v1.2g <code>\@include</code> : Support spaces in filenames by enclosing the names of <code>.aux</code> -files in quotes (gh/217)	308
<code>\load@onefile@withoptions</code> : Use different method to ignore unprocessed options (gh/22)	781, 782	<code>\includeonly</code> : Get rid of leading and trailing spaces from the filename (gh/217)	307
<code>\ProcessOptions*</code> : Use different method to ignore unprocessed options (gh/22)	772	Improved support for spaces in filenames (gh/217)	307
<code>\RequirePackageWithOptions</code> : Use different method to ignore unprocessed options (gh/22)	774	Pass the filename to <code>\@include</code> by value instead of by reference (gh/217)	307
2020-04-09 <code>ltfloat.dtx</code> v1.2d <code>\@textsubscript</code> : Set non-zero baseline (gh/249)	738	2020-05-05 <code>ltxref.dtx</code> v1.1n <code>\refstepcounter</code> : record the counter name in <code>\@currentcounter</code>	575
<code>\textsubscript</code> : Set non-zero baseline (gh/249)	737	2020-05-06 <code>ltspace.dtx</code> v1.3n General: Made <code>softhyphen</code> active in TU engines	297
2020-04-13 <code>ltfssdel.dtx</code> v3.0v <code>\process@table</code> : Small update for speed.	461	2020-05-09 <code>ltdefns.dtx</code> v1.5j <code>\@if@DeclareRobustCommand</code> : Added <code>\DeclareCommandCopy</code> (gh/239)	93
2020-04-13 <code>ltfssini.dtx</code> v3.1n <code>\init@series@setup</code> : Handling <code>\seriesdefault</code> changes (gh/315)	487	<code>\DeclareCommandCopy</code> : Added <code>\DeclareCommandCopy</code> (gh/239)	91
<code>\seriesdefault@kernel</code> : Handling <code>\seriesdefault</code> changes (gh/315)	504	2020-05-11 <code>ltdefns.dtx</code> v1.5j <code>\@dischph</code> : Do not overwrite <code>\-</code> under <code>LuaTeX</code>	100
2020-04-21 <code>ltmath.dtx</code> v1.2f <code>\@yeqncr</code> : Support calc syntax (gh/152)	608	2020-05-15 <code>ltdefns.dtx</code> v1.5g <code>\typeout</code> : Allow <code>\par</code> in the argument (gh/335)	73
2020-04-21 <code>ltmiscen.dtx</code> v1.1t <code>\@icentercr</code> : Support calc syntax (gh/152)	590	2020-05-19 <code>ltfssaxes.dtx</code> v1.0e <code>\series@maybe@drop@one@m</code> : Need to use <code>\edef</code> (gh/336)	416
2020-04-21 <code>ltpictur.dtx</code> v1.1o <code>\@istackcr</code> : Support calc syntax (gh/152)	675	2020-05-19 <code>ltfssini.dtx</code> v3.2a <code>\IfFontSeriesContextTF</code> : Macros added (gh/335)	495
2020-04-21 <code>ltspace.dtx</code> v1.3m <code>\@hspace</code> : Support calc syntax (gh/152)	296	2020-05-31 <code>ltmiscen.dtx</code> v1.1u <code>\centering</code> : Added <code>\finalhyphendemerits</code> setting (gh/247)	590
<code>\@newline</code> : Support calc syntax (gh/152)	284		
<code>\@vspace@calcify</code> : Support calc syntax (gh/152)	284		
<code>\@vspacer</code> : Support calc syntax (gh/152)	293		

<code>\raggedleft</code> : Added	<code>\date</code> : Don't make the command
<code>\finalhyphendemerits</code> setting	<code>\long</code> (gh/354) 706
(gh/247) 591	2020-08-01 ltluatex.dtx v1.1p
<code>\raggedright</code> : Added	General: new_ graf is exclusive . . . 59
<code>\finalhyphendemerits</code> setting	2020-08-02 ltluatex.dtx v1.1q
(gh/247) 590	<code>\newattribute</code> : Move reset to 0 inside
2020-06-04 ltexpl.dtx v1.2c	conditional 47
General: Define a local version of some	<code>\newluabytecode</code> : Move reset to 0
\LaTeX basic macros to support	inside conditional 50
package loading 68	<code>\newluachunkname</code> : Move reset to 0
2020-06-04 ltfinal.dtx v2.2a	inside conditional 50
General: Load ltexpl in ltdefns . . . 942	<code>\newluafunction</code> : Move reset to 0
2020-06-05 ltclass.dtx v1.3l	inside conditional 49
<code>\@currnamestack</code> : Added	<code>\newwhatsit</code> : Move reset to 0 inside
<code>\@expl@pop@filename@@</code> 761	conditional 50
Added <code>\@expl@push@filename@@</code>	2020-08-08 ltclass.dtx v1.3m
and	<code>\endfilecontents</code> : define
<code>\@expl@push@filename@aux@@</code> . . 760	<code>\q@curr@file</code> directly as the
2020-06-05 ltfiles.dtx v1.2h	quotes have already been removed
<code>\document</code> : Added hook to load	(gh/220) 787
\LaTeX backend code 301	2020-08-10 ltluatex.dtx v1.1r
2020-06-10 ltluatex.dtx v1.1n	General: Load ltluatex Lua module
General: Define	during format building 50
<code>\@gobble/\@firstofone</code> even for	2020-08-15 ltpictur.dtx v1.2a
\LaTeX to allow early loading. . . . 45	<code>\@defaultunitsset</code> : Macro added . . 672
2020-07-04 ltoutenc.dtx v2.0q	2020-08-19 ltdefns.dtx v1.5k
General: Implement <code>\remove@tlig</code> in	<code>\@carcube</code> : Made <code>\long</code> for
\LaTeX without font reloading . . 352	<code>\NewCommandCopy</code> 76
2020-07-08 ltexpl.dtx v1.2d	<code>\robust@command@act</code> : Made
General: Add a last-minute hook for	<code>\robust@command@act</code> (was
<code>expl3</code> 69	<code>\declare@command@copy</code>) more
2020-07-08 ltfinal.dtx v2.2b	generic 89
General: Add a last-minute hook for	<code>\ShowCommand</code> : Added <code>\ShowCommand</code>
<code>expl3</code> 933	(gh/373) 92
2020-07-27 ltmath.dtx v1.2g	2020-08-19 ltexpl.dtx v1.2e
<code>\cases</code> : Don't make the command	General: Add
<code>\long</code> (gh/354) 602	<code>\@expl@cs@<thing>@spec@N</code> for
<code>\matrix</code> : Don't make the command	<code>\ShowCommand</code> (gh/373) 71
<code>\long</code> (gh/354) 602	Add <code>\@expl@cs@to@str@N</code> and
<code>\pmatrix</code> : Don't make the command	<code>\@expl@str@if@eq@nnTF</code> for
<code>\long</code> (gh/354) 602	<code>\NewCommandCopy</code> (gh/239) 71
2020-07-27 ltoutenc.dtx v2.0r	2020-08-20 ltplain.dtx v2.3d
<code>\@use@text@encoding</code> : Don't make the	<code>\alloc@</code> : Define <code>\alloc@</code> in terms of
command <code>\long</code> (gh/354) . . 329, 330	<code>\e@alloc</code> 21
2020-07-27 ltpage.dtx v1.0m	2020-08-21 ltclass.dtx v1.3o
<code>\markright</code> : Don't make the command	General: Integration of new hook
<code>\long</code> (gh/354) 751	management interface 754
2020-07-27 ltpictur.dtx v1.1p	2020-08-21 ltdefns.dtx v1.5l
<code>\linethickness</code> : Don't make the	General: Integration of new hook
command <code>\long</code> (gh/354) 675	management interface 73
2020-07-27 ltsect.dtx v1.1e	2020-08-21 ltdefns.dtx v1.5m
<code>\author</code> : Don't make the command	<code>\MakeRobust</code> : Make <code>\MakeRobust</code>
<code>\long</code> (gh/354) 706	produce the same command

structure as	2020-09-30 ltffssini.dtx v3.2d
<code>\DeclareRobustCommand</code> 85	<code>\bfseries</code> : <code>\bfdefault@previous</code> not
2020-08-21 ltexpl.dtx v1.2d	<code>\bfseries@previous</code> (gh/395) .. 489
General: Dropped unused command . 68	<code>\mdseries</code> : <code>\mddefault@previous</code> not
2020-08-21 ltfiles.dtx v1.2i	<code>\mdseries@previous</code> (gh/395) .. 490
General: Integration of new hook	2020-10-01 ltclass.dtx v1.3r
management interface 299	<code>\providepackage</code> : Allow for package
2020-08-21 ltfinal.dtx v2.2i	substitution 766
General: Integration of new hook	2020-10-01 ltsect.dtx v1.1e
management interface 927	<code>\addcontentsline</code> : add a fourth
2020-08-21 ltffssaxes.dtx v1.0g	argument for better hyperref
General: Integration of new hook	compatibility 715
management interface 423	2020-10-04 ltfiles.dtx v1.2j
2020-08-21 ltffssini.dtx v3.2b	<code>\include</code> : Quotes around the aux file
<code>\bf@default</code> : Integration of new hook	name removed, they are not
management interface 489	needed and upset BibTeX
<code>\bfseries</code> : Integration of new hook	(gh/400) 309
management interface 489	2020-10-04 lthooks.dtx v1.0d
<code>mdseries/defaults</code> : Integration of	General: Definition <code>\AddToHookNext</code>
new hook management interface 492	was supposed to be for <code>\AddToHook</code>
<code>\mdseries</code> : Integration of new hook	vice versa (gh/401) 230
management interface 490	2020-10-08 ltclass.dtx v1.3s
<code>\reset@font</code> : Integration of new hook	<code>\@currnamestack</code> : Added missing
management interface 501	2020/02/02 <code>\IncludeInRelease</code> . 760
<code>\rmfamily</code> : Integration of new hook	2020-10-11 ltclass.dtx v1.3t
management interface 492	<code>\load@onefilewithoptions</code> : Restore
<code>\ttfamily</code> : Integration of new hook	<code>\@currpkg@reqd</code> after finished
management interface 492	loading a package file (gh/408). . 780
2020-08-21 ltmiscen.dtx v1.1v	2020-10-18 ltclass.dtx v1.3t
General: Integration of new hook	<code>\PassOptionsToClass</code> : Drop path
management interface 577	from <code>\input@path</code> (gh/414). . . 768
2020-08-21 ltoutput.dtx v1.4f	2020-10-23 ltmiscen.dtx v1.1w
<code>\@begindvibox</code> : Integration of new	<code>\enddocument</code> : Make
hook management interface 864	<code>\enddocument/afteraux</code> one-time 579
2020-08-23 ltxref.dtx v1.1o	2020-10-26 ltmiscen.dtx v1.1x
<code>\refstepcounter</code> : add default	<code>\@kernel@before@enddocument</code> :
definition of <code>\@currentcounter</code> . 575	<code>\enddocument</code> should always start
2020-09-06 ltclass.dtx v1.3q	out in vmode (gh/385) 581
<code>\load@onefilewithoptions</code> : Save	2020-11-09 ltclass.dtx v1.3u
<code>\@currpkg@reqd</code> so that we don't	<code>\pkgcls@rollbackdate@error</code> :
lose track of package	Change help text because the
substitutions. 780	package may have existed then —
2020-09-06 ltdefns.dtx v1.5n	there is just no rollback data
<code>\char@if@alph</code> : Macro added 99	(gh/423). 800
2020-09-06 ltexpl.dtx v1.2f	2020-11-09 ltmath.dtx v1.2h
General: Add	<code>\exp@one@str@map@function@CNN</code> and <code>\exp@one@str@map@function@CNN</code>
<code>\exp@one@str@map@function@CNN</code> and <code>\exp@one@str@map@function@CNN</code>	for <code>\string@makeletter</code> (gh/386) 72
2020-09-09 ltshipout.dtx v1.0b	<code>\negthickspace</code> and <code>\negthickspace</code> have been only in
<code>__shipout_picture_overlay:n</code> :	amsmath, so we need to undefine
Prevent overfull box warnings	for rollback (gh/423) 604
(gh/387) 844	2020-11-20 ltclass.dtx v1.3u
2020-09-26 ltfinal.dtx v2.2j	<code>\@currpath</code> : Macro added 759
General: Load first aid file if existing 943	<code>\@kernel@currpathstack</code> : Macro
	added 762

<code>\load@onefile@withoptions</code> : Copy option list to the requested package.	781	<code>\fontseries</code> : Distangle series and shape update (gh/444)	412
<code>\PassOptionsToClass</code> : Copy option list to the requested package. . . .	768	<code>\fontshape</code> : Distangle series and shape update (gh/444)	420
<code>\ProvidesPackage</code> : Use string comparison instead of <code>\ifx</code>	765	<code>\fontshapeforce</code> : Distangle series and shape update (gh/444)	420
2020-11-20 ltcmd.dtx v1.0a		2020-12-04 ltfssini.dtx v3.2f	
General: Initial version derived from <code>xparse.dtx</code>	103	General: Adjust start values for series and shape (gh/444)	503
2020-11-20 ltfilehook.dtx v1.0d		2020-12-10 ltbibl.dtx v1.1s	
<code>\unqu@tefilef@und</code> : Move loading to <code>\@input@file@exists@with@hooks</code> and expand <code>\@filef@und</code> to avoid getting the wrong file name in the case of a substitution.	812	<code>\nocite</code> : Delay any <code>\nocite</code> in the preamble instead of raising an error	748
2020-11-23 ltshipout.dtx v1.0d		2020-12-10 ltfssbas.dtx v3.2h	
<code>__shipout_execute_cont::</code> Check for both kernel and user hook (gh/431)	834	<code>\usefont</code> : Drop “m” if the series value is a member of a fixed list and issue warning if doing it (gh/453)	385
<code>__shipout_execute_main_cont:Nnnn</code> : Check for both kernel and user hook (gh/431)	836	2020-12-14 ltclass.dtx v1.3v	
2020-11-24 ltexpl.dtx v1.2g		<code>\@currnamestack</code> : Removed <code>\@expl@@hook@curr@name@push@@n</code>	760
General: Support for roll forward (gh/434)	70–72	2020-12-18 ltexpl.dtx v1.2h	
2020-11-24 ltfilehook.dtx v1.0d		<code>\@kernel@after@enddocument@afterlastpage</code> : Define kernel <code>\enddocument</code> hooks early	68
General: Support for roll forward (gh/434)	810	2020-12-22 ltfssaxes.dtx v1.0h	
2020-11-24 lthooks.dtx v1.0f		<code>\delayed@merge@font@series</code> : Distangle series and shape update (gh/444)	414, 415
<code>__hook_end_document_label_check::</code> Support for roll forward (gh/434)	227	<code>\delayed@merge@font@shape</code> : Distangle series and shape update (gh/444)	421
2020-11-24 ltshipout.dtx v1.0d		2020-12-22 ltfsstrc.dtx v3.0n	
General: Support for roll forward (gh/434)	848	<code>\selectfont</code> : Execute delayed series and shape updates (gh/444)	428
<code>\AtBeginDvi</code> : Support for roll forward (gh/434)	847	2021-01-07 ltfilehook.dtx v1.0e	
2020-11-25 ltdefns.dtx v1.5o		General: Added rollback for this case to avoid spurious errors (part of gh/463)	822
<code>\@carcube</code> : Added missing <code>latexrelease</code> entry	76	<code>\unqu@tefilef@und</code> : Restore <code>\CurrentFile(Path)(Used)</code> after the input (gh/464)	812
2020-12-02 ltluatex.dtx v1.1s		2021-01-07 lthooks.dtx v1.0h	
General: Fix return value of list callbacks	61	<code>__hook_strip_double_slash:w</code> : Assume hook name has at least three nonempty parts (gh/464)	202
2020-12-03 ltfsstrc.dtx v3.0m		<code>__hook_tl_set:cx</code> : Manually define some <code>l3tl</code> commands to work around <code>expl3</code> changes	188
<code>\selectfont</code> : Install a hook in <code>\selectfont</code> (gh/444)	429	2021-01-08 ltshipout.dtx v1.0f	
2020-12-04 ltfilehook.dtx v1.0d		<code>__shipout_execute_cont::</code> Added another kernel hook for more flexibility (cf	
<code>\undeclare@file@substitution</code> : Don’t drop file substitution commands on rollback	815		
2020-12-04 ltfssaxes.dtx v1.0h			
General: Reorganized the rollback data	403		

https://github.com/pgf-tikz/pgf/issues/1900	2021-01-10 ltfloat.dtx v3.0w	834
2021-01-10 ltshipout.dtx v1.0g	<code>\document@select@group</code> : fix for (gh/501)	459
<code>\@kernel@after@shipout@background</code> : Internal hook	2021-02-16 ltfloat.dtx v1.2f	
<code>\@kernel@after@shipout@background</code> added	<code>\footref</code> : <code>\footref</code> added	740
<code>\RawShipout</code> : Macro added	2021-02-17 ltoutenc.dtx v2.0t	
2021-01-19 ltshipout.dtx v1.0h	General: Adjust values for <code>\textasteriskcentered</code> To match TS1 definition (gh/502)	357
<code>__shipout_run_firstpage_hook::</code> : Handling of firstpage hook altered	Special definition for <code>\textasteriskcentered</code> when missing in TS1 (gh/502)	348
2021-01-21 ltclass.dtx v1.3w	2021-02-18 ltclass.dtx v1.3x	
<code>\@kernel@currpathstack</code> : Add empty entry for latexrelease	<code>\@fileswithoptions</code> : save raw class option list (gh/85)	776
2021-01-21 ltexpl.dtx v1.3a	<code>\@remove@eq@value</code> : macro added (gh/85)	769
General: Move xparse rollback code to <code>ltcmd.dtx</code>	<code>\@use@option</code> : value from unused option list (gh/85)	772
2021-01-21 ltfinal.dtx v2.2l	<code>\OptionNotUsed</code> : value from unused option list (gh/85)	769
General: Load glyphtounicode.tex for pdfTeX	<code>\PassOptionsToClass</code> : save raw option lists (gh/85)	768
2021-01-22 ltshipout.dtx v1.0i	2021-02-19 ltoutenc.dtx v2.0u	
<code>__shipout_finalize_box::</code> : Add <code>pre_shipout_filter</code> Lua callback	General: Add <code>\textnonbreakinghyphen</code> , <code>\textfiguredash</code> and <code>\texthorizontalbar</code> (gh/404)	337, 341, 356
2021-01-24 ltexpl.dtx v1.3a	2021-02-25 ltfinal.dtx v2.2m	
General: Define <code>expl3</code> hooks conditionally	General: Improve speed of <code>ToUnicode</code> everyjob loading code	934
2021-01-31 ltfilehook.dtx v1.0f	2021-03-03 ltclass.dtx v1.3y	
<code>\@curr@file@reqd</code> : set <code>\protect</code> to <code>\string</code> gh/481	<code>\endfilecontents</code> : Fix overwrite check for files with UTF-8 (gh/415)	787
2021-02-03 ltfloat.dtx v1.2e	2021-03-05 ltclass.dtx v1.3z	
<code>\@savemarbox</code> : Explicitly end with <code>\par</code> (gh/489)	<code>\ProcessOptions*</code> : modify so braces to not give errors (gh/513)	770
2021-02-04 ltboxes.dtx v1.4b	2021-03-12 ltfiles.dtx v1.2k	
<code>\color@endbox</code> : Always add the color groups (gh/488)	<code>\IfFileExists@</code> : Allow unbalanced conditionals (gh/530)	312
2021-02-08 ltfilehook.dtx v1.0g	2021-03-18 ltcmd.dtx v1.0b	
<code>\unqu@tefilef@und</code> : Undo the internal for robust <code>\InputIfFileExists</code> in rollback (gh/494)	General: Use <code>\NewModuleRelease</code>	103
2021-02-08 ltmiscen.dtx v1.1y	2021-03-18 ltfilehook.dtx v1.0h	
<code>\end</code> : Undo the internals for robust <code>\begin</code> and <code>\end</code> in rollback (gh/494)	<code>__filehook_file_pop_assign:nnnn</code> : Define <code>\g_@@_input_file_seq</code> to avoid losing data when rolling back.	809
2021-02-10 ltboxes.dtx v1.4b	2021-03-18 ltfsaxes.dtx v1.0i	
<code>\@mpfootnotetext</code> : Explicitly run <code>\par</code> in support for paragraph tagging	General: Fix rollforward definition.	413
2021-02-10 ltfloat.dtx v1.2e	2021-03-18 ltfsini.dtx v3.2g	
<code>\@footnotetext</code> : Explicitly run <code>\par</code> at the end of footnote text in preparation for paragraph hooks	General: Add legacy hook definitions for rollback.	493

2021-03-18 lthooks.dtx v1.0i	2021-04-20 ltexpl.dtx v1.3c
<code>__hook_end_document_label_check::</code>	<code>\@kernel@after@enddocument@afterlastpage:</code>
Only add <code>top-level</code> if not already	Don't empty kernel hooks on
there. 226	rollback 68
Remove the (empty) "top-level"	2021-04-20 ltfilehook.dtx v1.0i
from <code>\@currnamestack</code> 227	<code>\@curr@file@reqd:</code> Make expand to
General: Use <code>\NewModuleRelease</code> . . . 186	a string (tracks change in
2021-03-18 ltvers.dtx v1.1f	l3kernel) 815
<code>\@check@IncludeInRelease:</code> Add	2021-04-26 ltfsbas.dtx v3.2i
support for usage in	<code>\usefont:</code> Unconditionally switch to
<code>\NewModuleRelease</code> 38	the requested font face (gh/444) 385
<code>\new@moduledate:</code> Added	2021-04-26 ltfsini.dtx v3.2h
<code>\NewModuleRelease</code> 39	<code>\reset@font:</code> Unconditionally switch
2021-03-19 lttextcomp.dtx v1.0e	to the requested font face
General: Use <code>\NewModuleRelease</code> . . 539	(gh/444) 501
2021-03-26 ltplain.dtx v2.3e	2021-04-26 ltfsstrc.dtx v3.0o
<code>\@unused:</code> Allocate <code>\@inputcheck</code> and	<code>\selectfont:</code> Unset the forced series
<code>\@unused</code> early so that they are	boolean when reaching
before expl3 allocates more	<code>\selectfont</code> (gh/444) 429
streams (gh/538) 23	2021-04-29 lthooks.dtx v1.0m
2021-03-27 ltclass.dtx v1.4a	<code>\ProvideMirroredHookPair:</code> Add
<code>\@currnamestack:</code> Do not completely	<code>\ProvideHook</code> etc. 225
roll back if expl3 is loaded. . . . 761	2021-04-29 ltoutenc.dtx v2.0v
2021-04-16 ltvers.dtx v1.1g	General: Add composites for
<code>\new@moduledate:</code> <code>\NewModuleRelease</code>	<code>\ae/\AE/\ae/\AE</code> (gh/552) 361
with the same arguments as	2021-05-18 ltclass.dtx v1.4b
<code>\IncludeInRelease</code> 39	<code>\@raw@classoptionslist:</code> Initialise to
2021-04-17 ltfiles.dtx v1.2m	<code>\relax</code> to match
<code>\@kernel@after@begindocument:</code>	<code>\@classoptionslist</code> 758
Move	2021-05-24 ltcmd.dtx v1.0e
<code>\@kernel@before@begindocument</code>	General: Use <code>\msg_...</code> instead of
and	<code>__kernel_msg...</code> 103
<code>\@kernel@after@begindocument</code>	2021-05-24 ltcmdhooks.dtx v1.0b
init earlier so that other modules	General: Use <code>\msg_...</code> instead of
can write to the hooks 304	<code>__kernel_msg...</code> 234
2021-04-18 ltluatex.dtx v1.1t	2021-05-24 ltfilehook.dtx v1.0k
General: <code>input_level_string</code> added . 60	General: Use <code>\msg_...</code> instead of
2021-04-18 ltplain.dtx v2.3f	<code>__kernel_msg...</code> 808
<code>\loggingall:</code> 3 32	2021-05-24 lthooks.dtx v1.0n
Drop pre- ϵ -TeX support 32	General: Use <code>\msg_...</code> instead of
<code>\tracingnone:</code> 3 33	<code>__kernel_msg...</code> 186
Drop pre- ϵ -TeX support 33	2021-05-24 ltpara.dtx v1.0g
2021-04-19 ltcmd.dtx v1.0d	General: Use <code>\msg_...</code> instead of
<code>__kernel_cmd_if_xparse:NTF:</code>	<code>__kernel_msg...</code> 272
Renamed	2021-05-26 ltdefs.dtx v1.5p
<code>__cmd_cmd_if_xparse:NTF</code> to	<code>\MakeRobust:</code> Normalize error message
<code>__kernel_cmd_if_xparse:NTF</code> for	in <code>\MakeRobust</code> 85
cross-module usage 154	

Index

The italic numbers denote the pages where the corresponding entry is described, numbers underlined point to the definition, all others indicate the places where it is used.

Symbols	
<code>\!</code>	b387 , b389 , H201 , X509
<code>\"</code>	r226 , r378 , r419 , r457 , r468 , r579 , r611 , r638 , r646 , r652 , r656 , r662 , r666 , r672 , r678 , r685 , r686 , r692 , r696 , r746 , r792 , r1236 , r1254 , r1260 , r1264 , r1270 , r1274 , r1280 , r1286 , r1293 , r1294 , r1300 , r1304 , r1306 , r1413 , u416 , D171 , D198 , X510
<code>\#</code>	a62 , a75 , b6 , b14 , b454 , f696 , i238 , u403 , X493
<code>\\$</code>	a74 , b4 , b13 , f695 , r307 , r444 , r451 , r565 , r804 , r811 , D631 , D638 , X494
<code>\%</code>	a75 , a105 , a107 , a127 , b14 , b452 , f696 , r491 , r493 , u405 , D640 , D760 , G135 , S1403 , S1404 , X495
<code>\&</code>	a74 , b5 , b13 , b453 , f695 , S364 , X496
<code>\'</code>	b474 , r227 , r379 , r421 , r455 , r465 , r581 , r591 , r597 , r599 , r602 , r604 , r612 , r618 , r624 , r626 , r629 , r631 , r639 , r643 , r650 , r654 , r659 , r664 , r667 , r669 , r676 , r681 , r682 , r689 , r694 , r697 , r747 , r794 , r813 , r815 , r816 , r817 , r820 , r822 , r823 , r824 , r826 , r827 , r1230 , r1251 , r1258 , r1262 , r1267 , r1272 , r1275 , r1277 , r1284 , r1289 , r1290 , r1297 , r1302 , r1305 , r1313 , r1314 , r1361 , r1362 , r1367 , r1368 , r1379 , r1380 , r1385 , r1386 , r1414 , r1415 , r1429 , r1430 , r1431 , r1432 , r1445 , r1446 , u415 , z628 , A253 , D161 , G576 , H241 , J290 , J311 , K72 , V584 , X511
<code>\(</code>	H258 , H332
<code>\)</code>	b474 , H258 , H333
<code>*</code>	u408 , H238 , S1084 , S1215 , S1329 , S1405
<code>\+</code>	K72
<code>\,</code>	b388 , b390 , A506 , G576 , H7 , H8 , H40 , H154 , H156 , H159 , H184 , H207 , H208 , H224
<code>\-</code>	100 , 297 , b356 , f24 , f730 , o416 , o478 , r416 , r417 , r574 , r788 , r789 , u410 , G576 , J289 , J310 , K72 , X210 , X250
<code>\.</code>	b387 , b389 , q45 , q114 , q172 , r228 , r380 , r452 , r453 , r474 , r587 , r588 , r614 , r615 , r641 , r748 , r818 , r825 , r1235 , r1317 , r1318 , r1327 , r1328 , r1337 , r1338 , r1355 , r1416 , r1417 , r1453 , r1454 , u409 , D173 , D199
<code>\..default</code>	404
<code>\/</code>	a97 , f25 , i152 , u357 , u411 , S363
<code>\:</code>	b388 , b390 , f690 , f691 , H201 , H239
<code>\;</code>	603 , b388 , b390 , A500 , H185 , H201
<code>\<</code>	r575 , r739 , u406 , G576 , K71 , K109
<code>\=</code>	r229 , r381 , r473 , r749 , r1233 , r1307 , r1308 , r1323 , r1324 , r1346 , r1347 , r1348 , r1373 , r1374 , r1399 , r1400 , r1433 , r1434 , r1435 , r1436 , r1451 , r1452 , r1459 , r1460 , z628 , D179 , J290 , J311 , K71
<code>\></code>	r572 , r740 , u407 , G576 , H213 , H228 , H229 , H239 , K71
<code>\?</code>	b387 , b389 , X511
@@ commands:	
<code>\@@_set_curr_file:nNN</code>	T351
<code>\@@_set_curr_file_assign:nnnNN</code>	T351
<code>\@@\textvisiblespace_\meta_{hook}</code>	189
<code>\@@_next\textvisiblespace_\meta_{hook}</code>	189
<code>\@@par</code>	262
<code>\@TeXversion</code>	1 , 6
@botlist commands:	
<code>\@botlist:</code>	V1029 , V1123 , V1282
@botnum commands:	
<code>\@botnum:</code>	V1003
<code>\@car</code>	75
<code>\@cdr</code>	75
@citeb commands:	
<code>\@citeb:</code>	Q16 , Q45 , Q62
@colht commands:	
<code>\@colht:</code>	V541 , V555
@colnum commands:	
<code>\@colnum:</code>	V1188 , V1356
<code>\@cons</code>	75
<code>\@currdir</code>	1 , 6
@currname commands:	
<code>\@currname:</code>	q655
<code>\@dblarg</code>	74
@dbldeferlist commands:	
<code>\@dbldeferlist:</code>	V1755 , V1797
@dbltopnum commands:	
<code>\@dbltopnum:</code>	V1654 , V1781
@deferlist commands:	
<code>\@deferlist:</code>	V1114 , V1211 , V1272 , V1380 , V1424 , V1453 , V1511 , V1543 , V1629 , V1669

File Key: a=ltldirchk.dtx, b=ltplain.dtx, c=ltvers.dtx, d=ltluatex.dtx, e=ltexpl.dtx, f=ltdefns.dtx, g=ltcmd.dtx, h=lt hooks.dtx, i=ltcmdhooks.dtx, j=ltalloc.dtx, k=ltcntrl.dtx, l=lterror.dtx, m=ltpar.dtx, n=lt para.dtx, o=lt space.dtx, p=ltlogos.dtx, q=ltfiles.dtx, r=ltoutenc.dtx, s=ltcounts.dtx, t=ltlength.dtx, u=ltfssbas.dtx, v=ltfssaxes.dtx, w=ltfssstrc.dtx, x=ltfsscmp.dtx, y=ltfssdcl.dtx, z=ltfssini.dtx, A=fontdef.dtx, B=preload.dtx, C=ltfntcmd.dtx, D=lttextcomp.dtx, E=ltpageno.dtx, F=ltxref.dtx, G=ltmiscen.dtx, H=ltmath.dtx, I=ltlists.dtx, J=ltboxes.dtx, K=lttab.dtx, L=lt pictur.dtx, M=ltthm.dtx, N=ltsect.dtx, O=ltfloat.dtx, P=ltidxglo.dtx, Q=ltbibl.dtx, R=lt page.dtx, S=ltclass.dtx, T=ltfilehook.dtx, U=ltshipout.dtx, V=ltoutput.dtx, W=lthyphen.dtx, X=ltfinal.dtx

<code>\@ifclasslater</code>	757	r1292, r1298, r1303, r1315, r1316,
<code>\@ifclassloaded</code>	757	r1333, r1334, r1341, r1342, r1356,
<code>\@ifclasswith</code>	757	r1357, r1358, r1387, r1388, r1409,
<code>\@ifdefinable</code>	74	r1410, r1411, r1412, u398, u399,
<code>\@ifnextchar</code>	74	u404, D169, D197, G127, G136,
<code>\@ifpackagelater</code>	757	S1085, S1086, S1087, S1168, S1171,
<code>\@ifpackageloaded</code>	757	S1174, S1216, S1217, S1218, S1300,
<code>\@ifpackagewith</code>	757	S1303, S1306, S1330, S1331, S1332,
<code>\@ifstar</code>	74	S1390, S1393, S1396, X241, X242,
<code>\@ifundefined</code>	74	X243, X244, X245, X246, X247,
<code>\@missingfileerror</code>	758	X248, X249, X498, X504, X505,
<code>\@namedef</code>	74	X506, X507, X541, X542, X543,
<code>\@nameuse</code>	74	X544, X545, X546, X547, X548, X549
<code>\@restorepar</code>	262	<code>\</code>
<code>\@setpar</code>	262	f696, r314, A254, H256, H257, X499
<code>\@topnum</code> commands:		<code>\</code>
<code>\@topnum</code>	V1049	r232, r383, r420, r454, r464, r580,
<code>\[</code>	u412, z40, <u>H276</u> , H334, <u>H431</u> , X512	r642, r649, r653, r658, r663, r668,
<code>\]</code>	279	r675, r679, r680, r688, r693, r751,
<code>\[</code>	a45, a46, a74, a247, a248, a249, a250,	r793, r1229, r1250, r1257, r1261,
	a253, a260, a261, a262, a263, a266,	r1266, r1271, r1276, r1283, r1287,
	a273, a274, a275, a276, a279, a286,	r1288, r1296, r1301, u414, z628,
	a292, a293, a297, a299, a300, a304,	D175, G576, J290, J311, K72, X514
	a309, a310, a313, a319, b13, d264,	<code>\ </code>
	d416, f219, f276, f437, f525, f543,	r561, s135, s146, A573, A574, X515
	f695, g396, g460, g614, g626, g657,	<code>\~</code>
	g1874, g1904, g1918, g1949, g2042,	a75, b10, b14, f696, l20,
	g2081, g2090, g2123, h728, h739,	o422, r239, r287, r384, r467, r559,
	h1060, h1066, h1076, h1096, h1107,	r645, r657, r661, r671, r687, r691,
	h1108, h1113, h1118, h1123, i235,	r752, r1232, r1249, r1253, r1265,
	i289, l227, n107, n114, n121, n122,	r1269, r1279, r1295, r1299, r1343,
	n123, n124, o52, o480, q648, q663,	r1344, r1345, r1397, r1398, D187,
	r560, u400, A251, G358, G363,	D207, G507, G522, G537, G580, X502
	G368, G378, G382, G386, G410,	<code>\sqcup</code>
	H356, H491, J322, J455, J457,	a74, a91,
	K73, K170, K180, K194, L139, X497	b13, b393, b411, f695, l19, l20, l21,
<code>\{</code>	a3, a7, a74, b2, b13,	l22, l25, o421, u397, u647, u683,
	g545, g1326, i236, l22, r308, r562,	u708, A252, G406, G407, G512,
	u401, A249, G409, H59, H154, X500	G527, M36, M38, Q17, S357, X492
<code>\}</code>	a8, a74, b3, b13, i237, l21, r309,	A
	r563, u402, A250, G409, H59, X501	<code>\A</code>
<code>\<addto-cmd></code>	167	X238, X517, X538
<code>\<filename></code>	817	<code>\a</code>
<code>\<function></code>	142	r223, <u>K1</u> , X229, X518, X529
<code>\]</code>	b474, u413, <u>H276</u> , H335, <u>H455</u> , X513	<code>\AA</code>
<code>\^</code>	a63, a72, a75, a119, a330, b7, b9,	b399, r240, r428, r526
	b11, b14, b393, b394, b408, b409,	<code>\aa</code>
	f20, f696, g6, g7, g1362, g1622,	b399, r245, r422, r536
	g1627, g2270, o480, o482, o484,	<code>\abovedisplayskip</code>
	r231, r286, r382, r456, r466, r558,	b374, H499
	r644, r651, r655, r660, r665, r670,	<code>\abovedisplayskip</code>
	r677, r683, r684, r690, r695, r750,	b373,
	r1231, r1248, r1252, r1259, r1263,	H492, H494, H496, H497, H498, H499
	r1268, r1273, r1278, r1285, r1291,	<code>\accent</code>
		416, r73, r393, r423, r479, r763
		<code>\active</code>
		a64,
		a119, a330, b10, b11, b408, b409,
		b411, G406, G407, G506, G512,
		G521, G527, G536, G578, H241,
		H256, S1085, S1086, S1087, S1168,
		S1171, S1174, S1216, S1217, S1218,
		S1300, S1303, S1306, S1330, S1331,
		S1332, S1390, S1393, S1396, V584

File Key: a=ltldirchk.dtx, b=ltplain.dtx, c=ltvers.dtx, d=ltluatex.dtx, e=ltexpl.dtx, f=ltdefns.dtx, g=ltcmd.dtx, h=lt hooks.dtx, i=ltcmdhooks.dtx, j=ltalloc.dtx, k=ltcntrl.dtx, l=lterror.dtx, m=ltpar.dtx, n=lt para.dtx, o=lt space.dtx, p=lt logos.dtx, q=lt files.dtx, r=ltoutenc.dtx, s=ltcounts.dtx, t=ltlength.dtx, u=ltfssbas.dtx, v=ltfssaxes.dtx, w=ltfssstrc.dtx, x=ltfsscmp.dtx, y=ltfssdcl.dtx, z=ltfssini.dtx, A=fontdef.dtx, B=preload.dtx, C=ltfntcmd.dtx, D=lttextcomp.dtx, E=ltpageno.dtx, F=ltxref.dtx, G=ltmiscen.dtx, H=ltmath.dtx, I=ltlists.dtx, J=ltboxes.dtx, K=lttab.dtx, L=lt pictur.dtx, M=ltthm.dtx, N=ltsect.dtx, O=ltfloat.dtx, P=ltidxglo.dtx, Q=ltbibl.dtx, R=lt page.dtx, S=ltclass.dtx, T=ltfilehook.dtx, U=ltshipout.dtx, V=ltoutput.dtx, W=lthyphen.dtx, X=ltfinal.dtx

- \acute A516
- add commands:
 - add_to_callback d705
- \add_to_callback 44
- \addcontentsline 582, N70, N80, N159, O16
- \AddEverypageHook 832
- \addpenalty 291, o270, N50, V338, V1153, V1319, I124, I170, I175
- \AddThispageHook 832
- \addtocontents 582, N164, N171, N177, N180
- \addtocounter 366
- \addtocounter s6, s18
- \AddToHook 165, 167, 168, 173, 176, 179–181, 190, 191, 232, 236, 806, 830–832, h1160, h1276, w152, G38, G39, G303, G304, G305, G306, Q52, S990, S991, S992, T522, T524, T534, T535, T536, T537, U190, U464, U487
- \AddToHookNext 168, 173, 176, 218, 823, 828, 831, 832, h1162, h1274, T523, U488
- \addtolength 374
- \addtolength t16, H494, H496
- \addtoversion x20, x139
- \advspace o233, G336, N50, I124, I171, I172, I176, I224
- \adjdemerits b332
- \AE r241, r398, r527, r768, r1119, r1429, r1433, X570
- \ae r246, r401, r537, r772, r1125, r1431, r1435, X570
- \afterassignment 86, b424, b427, f256, f262, f305, r212, r220, u328, H186
- \AfterEndEnvironment 182, G299
- \aftergroup ... u91, u342, w202, w268, y114, y121, y129, C64, G510, G525, G540, J148, V604, V605, V662, V663
- \AfterLastShipout T534
- \afterpreamble 303
- \aleph A308
- \allocationnumber 274, b37, b57, b69, b71, b143, b144, b145, b195, b196, b237, b238, b239, b252, b253, b254, b271, b277, b283, b284, b297, b298, b299, d52, d53, d54, d91, d205, n39, K4, K9, X44, X45, X46
- \allowbreak b431, f772, f773, f792, f794, H40
- \Alph 366
- \Alph 829, s106
- \alph 366
- \alph s105
- \alpha A268
- \amalg A379
- \AmSfont 413
- \and 706
- \and N14, N27
- \angle A337
- \approx A422
- \arabic 366
- \arabic 829, s102, M33, U350
- \arccos H13
- \arcsin H10
- \arctan H16
- \arg H26
- \ArgumentSpecification 151, g1710, g1717, g1733, g1740
- array (environment) K168
- \array K168
- \arraycolsep H359, H360, H504, H505, K258, K338
- \arrayrulewidth .. K324, K338, K346, K347, K359, K363, K366, K376, K378
- \arraystretch K186, K187, K342
- \Arrowvert A569
- \arrowvert A567
- \asciispace G466, G468, G471, G474, G475, G494
- \ast A232, A395
- \asympt A449
- \AtBeginDocument 167, 170, 180, 181, 183, 422, 748, q125, q180, v737, Q34, S981
- \atbegindocumenthook 303
- \AtBeginDvi 181, 828, 831, 839, 840, 847, U403, U439, V86
- \AtBeginEnvironment 182, G299
- \AtBeginShipout 831, U444, U487
- \AtBeginShipoutAddToBox 830, U492
- \AtBeginShipoutAddToBoxForeground .. 830, U492
- \AtBeginShipoutBox 830, U485
- \AtBeginShipoutDiscard 831, U491
- \AtBeginShipoutFirst ... 831, U447, U489
- \AtBeginShipoutInit 831, U486
- \AtBeginShipoutNext 831, U445, U487
- \AtBeginShipoutOriginalShipout 831, U500
- \AtBeginShipoutUpperLeft 830, U492
- \AtBeginShipoutUpperLeftForeground . 830, U492
- \AtEndAfterFileList T536
- \AtEndDocument 181, 184, G69, S981, T540
- \AtEndDvi 828, 831, U454, U459
- \AtEndEnvironment 182, G299
- \AtEndOfClass 805, H430, S981
- \AtEndOfPackage 805, S527, S546, S628, S639, S981
- \AtEndPreamble 183

File Key: a=ltldirchk.dtx, b=ltplain.dtx, c=ltvers.dtx, d=ltluatex.dtx, e=ltexpl.dtx, f=ltdefs.dtx, g=ltcmd.dtx, h=lthooks.dtx, i=ltcmdhooks.dtx, j=ltalloc.dtx, k=ltcntrl.dtx, l=lterror.dtx, m=ltpar.dtx, n=ltpara.dtx, o=ltspace.dtx, p=ltlogos.dtx, q=ltfiles.dtx, r=ltoutenc.dtx, s=ltcounts.dtx, t=ltlength.dtx, u=ltfssbas.dtx, v=ltfssaxes.dtx, w=ltfssstrc.dtx, x=ltfsscmp.dtx, y=ltfssdcl.dtx, z=ltfssini.dtx, A=fontdef.dtx, B=preload.dtx, C=ltfntcmd.dtx, D=lttextcomp.dtx, E=ltpageno.dtx, F=ltxref.dtx, G=ltmiscen.dtx, H=ltmath.dtx, I=ltlists.dtx, J=ltboxes.dtx, K=lttab.dtx, L=ltpictur.dtx, M=ltthm.dtx, N=ltsect.dtx, O=ltfloat.dtx, P=ltidxglo.dtx, Q=ltbibl.dtx, R=ltpage.dtx, S=ltclass.dtx, T=ltfilehook.dtx, U=ltshipout.dtx, V=ltoutput.dtx, W=lthyphen.dtx, X=ltfinal.dtx

- `\AtNextShipout` 831
`\atopwithdelims` H57, H58, H59
`\attribute` 41, d79
`\attributedef` d79, d215
`\attributezero` d215
`\AtVeryEndDocument` T535
`\AtVeryVeryEnd` T537
`\author` 706
`\author` N8, N24, N32
- B**
- `\b` r233, r389, r475, r759, r1240
`\backslash` A251, A590
`\bar` A520
`\baselineskip` b392, b422,
 b458, w186, w187, w188, w190,
 w191, A510, H158, H159, H178,
 H184, H188, J299, J318, K198,
 L136, L314, L373, U222, U273,
 V242, V273, V622, V637, V681, V696
`\baselinestretch` u319,
 w120, w121, w166, w167, w184, w245
`\batchmode` 317, e89, e95, e105,
 q593, q624, q625, x106, X590, X611
`\BeforeBeginEnvironment` 182, G299
`\BeforeClearDocument` T538
`\begin` 85,
 111, 165, 181–184, 202, 301, 586,
 g969, i46, i197, i199, w7, A4, A102,
 B4, G167, G168, H435, H447, L461,
 N14, N17, S649, U385, W3, X475, X479
`\begingroup` 301, 327, 586
`\belowdisplaysshortskip` b376, H498
`\belowdisplayskip` b375, H497
`\beta` A269
`\bezier` 670
`\bezier` L682, L683, L805, L806, L821, L823
`\bfdefault` 186,
 416, 482–485, 487–489, z15, z194,
 z201, z202, z203, z204, z213, z274,
 z285, z329, A92, A104, A106, A114
`\bfseries`
 186, 482, 485, 494, z13, z14, z189,
 z198, z268, z279, z280, z323, z327,
 z328, C19, F13, M36, M38, Q20, U386
`bfseries` z252
`bfseries/defaults` z252
`\bgroup` b406
`\bible` Q7, Q9, Q10
`\bibdata` Q25, Q29
`\bibitem` Q3
`\bibliography` 745
`\bibliography` Q27
`\bibliographystyle` 745
`\bibliographystyle` Q32
`\bibstyle` Q25, Q37
`\Big` A623, A626, A635, A637, H44, H45, H46
`\big` A624, A636, H41
`\bigbreak` b438, f774, f795
`\bigcap` A345
`\bigcirc` A392
`\bigcup` A346
`\Bigg` A630, A639, H50, H51, H52
`\bigg` A628, A638, H47, H48, H49
`\Biggl` H50
`\biggl` H47
`\Biggm` H51
`\biggm` H48
`\Biggr` H52
`\biggr` H49
`\Bigl` H44
`\bigl` H41
`\Bigr` H45
`\bigr` H42
`\bigodot` A353
`\bigoplus` A352
`\bigotimes` A351
`\Bigr` H46
`\bigr` H43
`\bigskip` b443, o400
`\bigskipamount` ... b442, o402, o403, O391
`\bigsqcup` A356
`\bigtriangledown` A361, A362
`\bigtriangleup` A360, A363
`\biguplus` A344
`\bigvee` A342
`\bigwedge` A343
`\binoppenalty` b323
`\bmod` H35
`\boldmath` p14, z466
 bool commands:
 `\bool_gset_false:N` h15, U15, U81, U90
 `\bool_gset_true:N` h10, U10, U122, U344
 `\bool_if:NTF` . 163, g97, g108, g120,
 g124, g133, g135, g145, g154, g155,
 g158, g163, g165, g391, g400, g402,
 g457, g471, g484, g534, g567, g611,
 g621, g654, g662, g667, g668, g676,
 g698, g711, g763, g809, g810, g859,
 g872, g889, g909, g1363, g1597,
 g1866, g1877, h21, h832, h841,
 i133, i138, i140, i194, U21, U88, U365
 `\bool_lazy_and:nnTF` .. g650, h133,
 h834, h1006, n52, U66, U113, U367
 `\bool_lazy_and_p:nn` h1009
 `\bool_lazy_any:nnTF` g411
 `\bool_lazy_or:nnTF` g69, h401

File Key: a=ltldirchk.dtx, b=ltplain.dtx, c=ltvers.dtx, d=ltluatex.dtx, e=ltexpl.dtx, f=ltdefns.dtx, g=ltcmd.dtx, h=lt hooks.dtx, i=ltcmdhooks.dtx, j=ltalloc.dtx, k=ltcntrl.dtx, l=lterror.dtx, m=ltpar.dtx, n=lt para.dtx, o=lt space.dtx, p=ltlogos.dtx, q=ltfiles.dtx, r=ltoutenc.dtx, s=ltcounts.dtx, t=ltlength.dtx, u=ltfssbas.dtx, v=ltfssaxes.dtx, w=ltfssstrc.dtx, x=ltfsscmp.dtx, y=ltfssdcl.dtx, z=ltfssini.dtx, A=fontdef.dtx, B=preload.dtx, C=ltfntcmd.dtx, D=lttextcomp.dtx, E=ltpageno.dtx, F=ltxref.dtx, G=ltmiscen.dtx, H=ltmath.dtx, I=ltlists.dtx, J=ltboxes.dtx, K=lttab.dtx, L=lt pictur.dtx, M=ltthm.dtx, N=ltsect.dtx, O=ltfloat.dtx, P=ltidxglo.dtx, Q=ltbibl.dtx, R=lt page.dtx, S=ltclass.dtx, T=ltfilehook.dtx, U=ltshipout.dtx, V=ltoutput.dtx, W=lthyphen.dtx, X=ltfinal.dtx

- \capitalogonek b253, b257, b259, b283, b298, b452,
r835, D180, D181, D204, D701, D941 b453, b454, d22, d26, d38, d47,
- \capitalring d48, d89, d157, d216, j2, q52, q121,
r844, D182, D183, D205, D702, D948 r18, u14, K4, K9, S1091, S1222,
- \capitaltie r850, D184, S1339, X28, X30, X34, X53, X157,
D185, D206, D703, D1013, D1014 X158, X159, X160, X161, X162, X163
- \capitaltilde \charsubdef X339
r841, D186, D187, D207, D704, D945 \charzero d216
- \caption O4 \check A522
- \cases H153, H154, H164, H166 \CheckCommand f187
- \catcode 582 \CheckEncodingSubset . 539, D16, D61,
D109, D110, D111, D156, D158,
- \catcodetable 41, d89, d109 D352, D621, D832, D882, D938,
D939, D1007, D1124, D1127, D1141
- \catcodetable@atletter 42 \chi A288
- \catcodetable@initex 42 \choose H57
- \catcodetable@latex 42 \circ A391
- \catcodetable@string 42 \circle L450, L605, L807, L824
- \cdot A394 \citation Q11, Q19, Q47, Q64
- \cdotp A502, A508 \cite 745
- \cdots A508 \cite Q12
- center (environment) G351 \clap J476
- \center G351 class/after 805
- \centering G351, class/after/... 805
G356, G357, G375, G377, G392, G394 class/before 805
- \centerline J472 class/before/... 805
- \changes 71, z520 \ClassError 184
- \char f737, \ClassInfo 184
f752, r391, r394, r430, r433, r444, \ClassWarning 184
r451, r477, r481, r486, r489, r491, \ClassWarningNoLine 184
r493, r733, r761, r764, r797, r804, \cleaders b472, A548, A551
r811, r834, r837, r866, r896, r1006, \cleardoublepage V138
r1042, r1157, r1159, r1161, r1208, \ClearHookRule 172, h1255, h1290
z479, z486, D631, D638, D640, \clearpage 184, 309, 805,
D760, G466, G581, H238, L232, 806, 845, 846, q295, q322, q326,
L282, L296, L304, L307, L448, q342, q360, G17, G72, G165, V125,
L563, L568, L576, L580, L616, V138, V143, V200, V407, V410,
L617, L619, L632, L633, L636, L663 V414, V455, V461, V2171, V2188
- char commands: \cline K367
- \char_generate:nn . e141, g1075, g1371
- \char_set_catcode_active:N ... g1622
- \char_set_catcode_active:n ... g1746
- \char_set_catcode_escape:N i235
- \char_set_catcode_group_begin:N i236
- \char_set_catcode_group_end:N .. i237
- \char_set_catcode_other:N g1360
- \char_set_catcode_other:n g1364
- \char_set_catcode_parameter:N .. i238
- \char_set_catcode_parameter:n . g1365
- \char_set_lccode:nn ... g1627, g1756
- \char_value_catcode:n i189
- \chardef
a64, a70, a71, b10, b16, b17, b18,
b19, b20, b58, b64, b66, b73, b79,
b82, b84, b94, b96, b97, b98, b99,
b108, b114, b115, b128, b130, b194,
- \charsubdef X339
- \charzero d216
- \check A522
- \CheckCommand f187
- \CheckEncodingSubset . 539, D16, D61,
D109, D110, D111, D156, D158,
D352, D621, D832, D882, D938,
D939, D1007, D1124, D1127, D1141
- \chi A288
- \choose H57
- \circ A391
- \circle L450, L605, L807, L824
- \citation Q11, Q19, Q47, Q64
- \cite 745
- \cite Q12
- \clap J476
- class/after 805
- class/after/... 805
- class/before 805
- class/before/... 805
- \ClassError 184
- \ClassInfo 184
- \ClassWarning 184
- \ClassWarningNoLine 184
- \cleaders b472, A548, A551
- \cleardoublepage V138
- \ClearHookRule 172, h1255, h1290
- \clearpage 184, 309, 805,
806, 845, 846, q295, q322, q326,
q342, q360, G17, G72, G165, V125,
V138, V143, V200, V407, V410,
V414, V455, V461, V2171, V2188
- \cline K367
- clist commands:
- \clist_gclear:N h634
- \clist_gput_left:Nn h580
- \clist_gput_right:Nn h582
- \clist_if_empty:NTF h851
- \clist_new:N h87
- \clist_use:Nn h853
- \clubpenalty 273, 301, b325,
q8, q25, q94, q151, u664, N100,
N106, N130, N135, I128, I194, I196
- \clubsuit A331
- cmd internal commands:
- __cmd_add_arg:n
. 131, g1011, g1022, g1027, g1028,

File Key: a=ltldirchk.dtx, b=ltplain.dtx, c=ltvers.dtx, d=ltluatex.dtx, e=ltexpl.dtx, f=ltdefns.dtx, g=ltcmd.dtx, h=lt hooks.dtx, i=ltcmdhooks.dtx, j=ltalloc.dtx, k=ltcntrl.dtx, l=ltterror.dtx, m=ltpar.dtx, n=lt para.dtx, o=lt space.dtx, p=ltlogos.dtx, q=ltfiles.dtx, r=ltoutenc.dtx, s=ltcounts.dtx, t=ltlength.dtx, u=ltfssbas.dtx, v=ltfssaxes.dtx, w=ltfssstrc.dtx, x=ltfsscmp.dtx, y=ltfssdcl.dtx, z=ltfssini.dtx, A=fontdef.dtx, B=preload.dtx, C=ltfntcmd.dtx, D=lttextcomp.dtx, E=ltpageno.dtx, F=ltxref.dtx, G=ltmiscen.dtx, H=ltmath.dtx, I=ltlists.dtx, J=ltboxes.dtx, K=lttab.dtx, L=lt pictur.dtx, M=ltthm.dtx, N=ltsect.dtx, O=ltfloat.dtx, P=ltidxglo.dtx, Q=ltbibl.dtx, R=ltpage.dtx, S=ltclass.dtx, T=ltfilehook.dtx, U=ltshipout.dtx, V=ltoutput.dtx, W=lthyphen.dtx, X=ltfinal.dtx

[g1062](#), [g1150](#), [g1157](#), [g1232](#), [g1245](#),
[g1246](#), [g1319](#), [g1377](#), [g1384](#), [g1397](#)
`__cmd_add_arg_spec:n`
..... [g500](#), [g514](#), [g574](#), [g648](#)
`__cmd_add_arg_spec_mandatory:n` .
..... [g546](#), [g556](#), [g562](#), [g648](#)
`__cmd_add_default:`
..... [124](#), [g740](#), [g762](#), [g779](#), [g820](#), [g907](#), [g916](#)
`__cmd_add_default:n`
... [124](#), [g747](#), [g771](#), [g787](#), [g820](#), [g851](#)
`__cmd_add_default_E:nn`
..... [g755](#), [g820](#), [g885](#)
`__cmd_add_expandable_grabber:nn`
... [g864](#), [g877](#), [g888](#), [g908](#), [g918](#), [g925](#)
`__cmd_add_expandable_type+:w` [g842](#)
`__cmd_add_expandable_type_D:w` [g847](#)
`__cmd_add_expandable_type_D-`
aux:NN [g847](#)
`__cmd_add_expandable_type_D-`
aux:NNN [g847](#)
`__cmd_add_expandable_type_D-`
aux:NNNn [g847](#), [g913](#)
`__cmd_add_expandable_type_E:w` [g883](#)
`__cmd_add_expandable_type_E-`
aux:n [g883](#)
`__cmd_add_expandable_type_m:w` [g905](#)
`__cmd_add_expandable_type_R:w` [g912](#)
`__cmd_add_expandable_type_t:w` [g914](#)
`__cmd_add_grabber:N` . [g741](#), [g748](#),
[g756](#), [g764](#), [g772](#), [g780](#), [g788](#), [g802](#)
`__cmd_add_type!:w` [g722](#)
`__cmd_add_type+:w` [g715](#)
`__cmd_add_type>:w` [g729](#)
`__cmd_add_type_b:w` [g737](#)
`__cmd_add_type_D:w` [g744](#)
`__cmd_add_type_E:w` [g752](#)
`__cmd_add_type_m:w` [g760](#)
`__cmd_add_type_R:w` [g768](#)
`__cmd_add_type_t:w` [g776](#)
`__cmd_add_type_v:w` [g784](#)
`__cmd_allowed_token_check:N` ...
..... [g498](#), [g510](#), [g531](#), [g552](#), [g590](#)
`\l__cmd_arg_spec_tl` ... [103](#), [117](#)–
[119](#), [g11](#), [g88](#), [g376](#), [g463](#), [g532](#), [g665](#)
`\l__cmd_args_i_tl`
..... [103](#), [g13](#), [g256](#), [g262](#), [g267](#), [g268](#), [g270](#)
`\l__cmd_args_ii_tl` [103](#),
[g14](#), [g266](#), [g268](#), [g270](#), [g308](#), [g313](#), [g320](#)
`__cmd_args_process:` [124](#), [g248](#), [g306](#)
`__cmd_args_process_aux:n` [g306](#)
`__cmd_args_process_loop:nn` .. [g306](#)
`\l__cmd_args_tl`
..... [103](#), [106](#), [113](#), [131](#), [136](#),
[g12](#), [g201](#), [g237](#), [g251](#), [g256](#), [g262](#),
[g281](#), [g310](#), [g313](#), [g326](#), [g327](#), [g971](#),
[g972](#), [g977](#), [g980](#), [g1128](#), [g1162](#), [g1399](#)
`__cmd_bad_arg_spec:wn` [g433](#), [g438](#),
[g447](#), [g456](#), [g470](#), [g483](#), [g497](#), [g507](#),
[g508](#), [g513](#), [g524](#), [g530](#), [g551](#), [g642](#)
`__cmd_bad_def:wn` [g389](#), [g397](#), [g426](#),
[g461](#), [g475](#), [g488](#), [g571](#), [g579](#), [g587](#),
[g606](#), [g615](#), [g627](#), [g642](#), [g658](#), [g680](#)
`__cmd_bool_reverse:N` . [g1595](#), [g2244](#)
`__cmd_break_point:n`
..... [g91](#), [g94](#), [g642](#), [g647](#)
`__cmd_check_definable:nNTF` [g1743](#),
[g2129](#), [g2142](#), [g2155](#), [g2160](#), [g2181](#),
[g2194](#), [g2207](#), [g2215](#), [g2251](#), [g2257](#)
`__cmd_check_definable_aux:nN` ...
..... [153](#), [g1743](#)
`__cmd_chk_if_free_cs:N`
..... [g1715](#), [g2261](#), [g2262](#)
`__cmd_cmd_if_xparse_aux:N` ... [g1810](#)
`__cmd_cmd_if_xparse_aux:w`
..... [g1816](#), [g1826](#)
`\l__cmd_current_arg_int`
..... [103](#), [118](#), [124](#), [g15](#), [g25](#),
[g105](#), [g134](#), [g141](#), [g210](#), [g280](#), [g284](#),
[g289](#), [g290](#), [g374](#), [g385](#), [g408](#), [g464](#),
[g478](#), [g492](#), [g517](#), [g687](#), [g825](#), [g832](#)
`__cmd_declare_cmd:Nnn`
..... [g51](#), [g2137](#), [g2145](#), [g2156](#), [g2161](#)
`__cmd_declare_cmd_aux:Nnn` [g51](#)
`__cmd_declare_cmd_code:Nnn` [g89](#), [g95](#)
`__cmd_declare_cmd_code_aux:Nnn` [g95](#)
`__cmd_declare_cmd_code_expandable:Nnn`
..... [g95](#)
`__cmd_declare_cmd_internal:Nnnn`
..... [110](#), [g51](#), [g196](#)
`__cmd_declare_env:nnnn`
... [g175](#), [g2167](#), [g2172](#), [g2176](#), [g2178](#)
`__cmd_declare_env_internal:nnnn`
..... [g175](#)
`__cmd_declare_expandable-`
cmd:Nnn
..... [g51](#), [g2189](#), [g2197](#), [g2210](#), [g2216](#)
`__cmd_defaults:` [g247](#), [g253](#)
`__cmd_defaults_aux:` [g253](#)
`\l__cmd_defaults_bool`
..... [104](#), [g16](#), [g120](#), [g135](#), [g158](#), [g691](#), [g826](#)
`__cmd_defaults_def:` [g253](#)
`__cmd_defaults_def:nn` [g253](#)
`__cmd_defaults_def:nnn` [g253](#)
`__cmd_defaults_error:w` [g253](#)

File Key: a=ltldirchk.dtx, b=ltplain.dtx, c=ltvers.dtx, d=ltluatex.dtx, e=ltexpl.dtx,
f=ltdefs.dtx, g=ltcmd.dtx, h=lt hooks.dtx, i=ltcmdhooks.dtx, j=ltalloc.dtx, k=ltcntrl.dtx,
l=lterror.dtx, m=ltpar.dtx, n=lt para.dtx, o=lt space.dtx, p=ltlogos.dtx, q=ltfiles.dtx,
r=ltoutenc.dtx, s=ltcounts.dtx, t=ltlength.dtx, u=ltfssbas.dtx, v=ltfssaxes.dtx,
w=ltfssstrc.dtx, x=ltfsscmp.dtx, y=ltfssdcl.dtx, z=ltfssini.dtx, A=fontdef.dtx,
B=preload.dtx, C=ltfntcmd.dtx, D=lttextcomp.dtx, E=ltpageno.dtx, F=ltxref.dtx,
G=ltmiscen.dtx, H=ltmath.dtx, I=ltlists.dtx, J=ltboxes.dtx, K=lttab.dtx, L=lt pictur.dtx,
M=ltthm.dtx, N=ltsect.dtx, O=ltfloat.dtx, P=ltidxglo.dtx, Q=ltbibl.dtx, R=lt page.dtx,
S=ltclass.dtx, T=ltfilehook.dtx, U=ltshipout.dtx, V=ltoutput.dtx, W=lthyphen.dtx,
X=ltfinal.dtx

\l__cmd_defaults_tl	__cmd_expandable_grab_E_find:w .
..... 104, g17, g121, g142, g1490
g240, g247, g281, g692, g827, g833	__cmd_expandable_grab_E_long:w .
__cmd_delimiter_check:nnn g1490
..... g545, g554, g631	__cmd_expandable_grab_E_-
__cmd_end_expandable:NNw g333, g335	loop:nnnNNw g1490
__cmd_end_expandable_aux:nnnnn g335	__cmd_expandable_grab_E_-
__cmd_end_expandable_aux:w .. g335	test:nnw g1490
__cmd_end_expandable_defaults:nnnNNn	__cmd_expandable_grab_m:w ... g1525
..... 114, g335	__cmd_expandable_grab_m_aux:wNn
__cmd_end_expandable_defaults:nnw g1525
..... g335	__cmd_expandable_grab_m_long:w .
__cmd_end_expandable_defaults:nw g1525
..... g335	__cmd_expandable_grab_R:w ... g1531
\l__cmd_environment_bool	__cmd_expandable_grab_R_alt:w g1558
.. 104, g18, g79, g108, g190, g220,	__cmd_expandable_grab_R_alt_-
g231, g402, g567, g1683, g1689, g1877	aux:NNwNNn g1558
__cmd_environment_or_command: ..	__cmd_expandable_grab_R_-
..... g275,	aux:NNwNNn g1531
g388, g420, g424, g474, g487, g570,	__cmd_expandable_grab_t:w ... g1585
g577, g586, g601, g645, g679, g1230,	__cmd_expandable_grab_t_-
g1374, g1381, g1699, g1703, g1875	aux:NNwn g1585
\l__cmd_environment_str	__cmd_flush_m_args: 124,
..... 104, g19, g111, g177,	125, g698, g717, g724, g731, g739,
g178, g179, g180, g183, g187, g192,	g746, g754, g770, g778, g786, g791
g219, g222, g223, g1690, g1692, g1878	\l__cmd_fn_code_tl 104, g29, g239, g251
\l__cmd_expandable_aux_name_tl ..	\l__cmd_fn_tl
104, g21, g22, g862, g866, g875, g879	... 104, 107, 131, g28, g85, g238,
\l__cmd_expandable_bool 104, g20,	g973, g1016, g1040, g1046, g1056,
g53, g58, g133, g145, g189, g391,	g1066, g1075, g1080, g1084, g1115,
g400, g457, g611, g621, g654, g698	g1141, g1149, g1151, g1156, g1158,
__cmd_expandable_grab_D:nnNNwNN	g1169, g1170, g1175, g1176, g1181,
..... g1403	g1182, g1187, g1188, g1193, g1194,
__cmd_expandable_grab_D:NNwNNn	g1199, g1200, g1205, g1206, g1211,
..... g1403	g1212, g1242, g1248, g1684, g1882
__cmd_expandable_grab_D:NNwNNnnn	\l__cmd_function_tl 105, g24, g30,
..... 144, g1403, g1546	g84, g86, g104, g115, g116, g132,
__cmd_expandable_grab_D:Nw .. g1403	g140, g150, g153, g157, g159, g167,
__cmd_expandable_grab_D:w ... g1403	g173, g396, g460, g614, g626, g657
__cmd_expandable_grab_D_-	__cmd_get_arg_spec:N . g1718, g2252
alt:NNwn g1470, g1477, g1573	__cmd_get_arg_spec:n . g1718, g2254
__cmd_expandable_grab_D_-	__cmd_get_arg_spec:NTF
alt:NNwNNn g1455	.. g1706, g1720, g1725, g1732, g1738
__cmd_expandable_grab_D.alt:Nwn	__cmd_get_arg_spec_error:N
..... g1455 g1681, g1721, g1734
__cmd_expandable_grab_D.alt:w g1455	__cmd_get_arg_spec_error:n
__cmd_expandable_grab_E:w ... g1490 g1681, g1728, g1741
__cmd_expandable_grab_E_aux:w g1490	__cmd_get_arg_spec_error_aux:n .
__cmd_expandable_grab_E_end:nnw g1681
..... g1490	__cmd_get_grabber:NN
__cmd_expandable_grab_E_- 129, g901, g917, g930
find:nnw g1490	__cmd_get_grabber_auxi:NN ... g930
	__cmd_get_grabber_auxii:NN .. g930

File Key: a=ltldirchk.dtx, b=ltplain.dtx, c=ltvers.dtx, d=ltluatex.dtx, e=ltexpl.dtx, f=ltdefns.dtx, g=ltcmd.dtx, h=lt hooks.dtx, i=ltcmdhooks.dtx, j=ltalloc.dtx, k=ltcntrl.dtx, l=lterror.dtx, m=ltpar.dtx, n=lt para.dtx, o=lt space.dtx, p=ltlogos.dtx, q=ltfiles.dtx, r=ltoutenc.dtx, s=ltcounts.dtx, t=ltlength.dtx, u=ltfssbas.dtx, v=ltfssaxes.dtx, w=ltfssstrc.dtx, x=ltfsscmp.dtx, y=ltfssdcl.dtx, z=ltfssini.dtx, A=fontdef.dtx, B=preload.dtx, C=ltfntcmd.dtx, D=lttextcomp.dtx, E=ltpageno.dtx, F=ltxref.dtx, G=ltmiscen.dtx, H=ltmath.dtx, I=ltlists.dtx, J=ltboxes.dtx, K=lttab.dtx, L=lt pictur.dtx, M=ltthm.dtx, N=ltsect.dtx, O=ltfloat.dtx, P=ltidxglo.dtx, Q=ltbibl.dtx, R=lt page.dtx, S=ltclass.dtx, T=ltfilehook.dtx, U=ltshipout.dtx, V=ltoutput.dtx, W=lthyphen.dtx, X=ltfinal.dtx

<code>__cmd_grab_b:w</code>	g959	<code>__cmd_grab_t_obey_spaces:w</code>	g1235
<code>__cmd_grab_b_aux:NNw</code>	g959	<code>__cmd_grab_v:w</code>	g1251
<code>__cmd_grab_b_end:Nw</code>	g959	<code>__cmd_grab_v_aux:w</code>	140 , g1251
<code>__cmd_grab_b_long:w</code>	g959	<code>__cmd_grab_v_aux_abort:n</code>	139 , g1277 , g1295 , g1301 , g1314 , g1333 , g1356 , g1358
<code>__cmd_grab_b_long_obey_spaces:w</code>	g959	<code>__cmd_grab_v_aux_catcodes:</code>	138 , g1292 , g1324 , g1358
<code>__cmd_grab_b_obey_spaces:w</code>	g959	<code>__cmd_grab_v_aux_loop:N</code>	g1287
<code>__cmd_grab_D:w</code>	g986	<code>__cmd_grab_v_aux_loop:NN</code>	g1287
<code>__cmd_grab_D_aux:NNnN</code>	g969 , g1006 , g1225	<code>__cmd_grab_v_aux_loop_end:</code>	g1287 , g1347
<code>__cmd_grab_D_aux:NNnN</code>	g988 , g993 , g998 , g1003 , g1006	<code>__cmd_grab_v_aux_put:N</code>	g1291 , g1310 , g1326 , g1344 , g1352 , g1387
<code>__cmd_grab_D_call:Nw</code>	132 , g1010 , g1070 , g1227	<code>__cmd_grab_v_aux_test:N</code>	g1276 , g1287
<code>__cmd_grab_D_long:w</code>	g986	<code>__cmd_grab_v_bgroup:</code>	g1272 , g1322
<code>__cmd_grab_D_long_obey_spaces:w</code>	g986	<code>__cmd_grab_v_bgroup_loop:</code>	g1322
<code>__cmd_grab_D_nested:NNnN</code>	g1019 , g1033	<code>__cmd_grab_v_bgroup_loop:N</code>	g1322
<code>__cmd_grab_D_nested:w</code>	g1033	<code>__cmd_grab_v_group_end:</code>	138 , g1251 , g1318 , g1369
<code>__cmd_grab_D_obey_spaces:w</code>	g986	<code>__cmd_grab_v_long:w</code>	g1251
<code>__cmd_grab_E:nnNN</code>	g1089	<code>__cmd_grab_v_token_if_char:NTF</code>	139 , g1289 , g1305 , g1337 , g1395
<code>__cmd_grab_E:w</code>	g1089	<code>\g__cmd_grabber_int</code>	104 , g27 , g943 , g947
<code>__cmd_grab_E_finalise:</code>	g1089	<code>\c__cmd_ignore_def_tl</code>	156 , g1873 , g1891 , g1898 , g1912 , g1926 , g1957 , g1967 , g1973 , g1981 , g1989 , g1999 , g2006 , g2013 , g2020 , g2032 , g2039
<code>__cmd_grab_E_long:w</code>	g1089	<code>\l__cmd_last_delimiters_tl</code>	105 , 115 , 118 , g33 , g375 , g393 , g501 , g515 , g537 , g573 , g623 , g633 , g682
<code>__cmd_grab_E_long_obey_spaces:w</code>	g1089	<code>\l__cmd_long_bool</code>	105 , 128 , g34 , g379 , g471 , g477 , g540 , g651 , g662 , g667 , g671 , g688 , g718 , g809 , g814 , g844 , g859 , g872 , g889 , g909 , g1253 , g1258 , g1363
<code>__cmd_grab_E_loop:NnN</code>	g1089	<code>\l__cmd_m_args_int</code>	105 , g35 , g690 , g765 , g793 , g796 , g798 , g800
<code>__cmd_grab_E_obey_spaces:w</code>	g1089	<code>\l__cmd_nesting_a_tl</code>	g1033
<code>\l__cmd_grab_expandably_bool</code>	105 , 115 , g31 , g97 , g377 , g401 , g403 , g465 , g502 , g516 , g538 , g555 , g617 , g660 , g711	<code>\l__cmd_nesting_b_tl</code>	g1033
<code>__cmd_grab_m:w</code>	g1146	<code>__cmd_normalize_arg_spec:n</code>	g87 , g372
<code>__cmd_grab_m_1:w</code>	g1160	<code>__cmd_normalize_arg_spec_loop:n</code>	g372 , g466 , g479 , g493 , g503 , g518 , g541 , g547 , g557 , g563
<code>__cmd_grab_m_2:w</code>	g1160	<code>__cmd_normalize_check_gv:N</code>	g561 , g609
<code>__cmd_grab_m_3:w</code>	g1160	<code>__cmd_normalize_check_lu:N</code>	g609
<code>__cmd_grab_m_4:w</code>	g1160	<code>__cmd_normalize_E_unique_-</code>	check:w g495
<code>__cmd_grab_m_5:w</code>	g1160	<code>__cmd_normalize_type_!:w</code>	g454
<code>__cmd_grab_m_6:w</code>	g1160	<code>__cmd_normalize_type_+:w</code>	g454
<code>__cmd_grab_m_7:w</code>	g1160		
<code>__cmd_grab_m_8:w</code>	g1160		
<code>__cmd_grab_m_9:w</code>	g1160		
<code>__cmd_grab_m_aux:Nnnnnnnnn</code>	g1160		
<code>__cmd_grab_m_long:w</code>	g1146		
<code>__cmd_grab_R:w</code>	g1219		
<code>__cmd_grab_R_aux:NNnN</code>	g1219		
<code>__cmd_grab_R_long:w</code>	g1219		
<code>__cmd_grab_t:w</code>	g1235		
<code>__cmd_grab_t_aux:NNw</code>	g1235		

File Key: a=ltldirchk.dtx, b=ltplain.dtx, c=ltvers.dtx, d=ltluatex.dtx, e=ltexpl.dtx, f=ltdefns.dtx, g=ltcmd.dtx, h=lt hooks.dtx, i=ltcmdhooks.dtx, j=ltalloc.dtx, k=ltcntrl.dtx, l=lterror.dtx, m=ltpar.dtx, n=lt para.dtx, o=lt space.dtx, p=ltlogos.dtx, q=ltfiles.dtx, r=ltoutenc.dtx, s=ltcounts.dtx, t=ltlength.dtx, u=ltfssbas.dtx, v=ltfssaxes.dtx, w=ltfssstrc.dtx, x=ltfsscmp.dtx, y=ltfssdcl.dtx, z=ltfssini.dtx, A=fontdef.dtx, B=preload.dtx, C=ltfntcmd.dtx, D=lttextcomp.dtx, E=ltpageno.dtx, F=ltxref.dtx, G=ltmiscen.dtx, H=ltmath.dtx, I=ltlists.dtx, J=ltboxes.dtx, K=lttab.dtx, L=lt pictur.dtx, M=ltthm.dtx, N=ltsect.dtx, O=ltfloat.dtx, P=ltidnglo.dtx, Q=ltbibl.dtx, R=lt page.dtx, S=ltclass.dtx, T=ltfilehook.dtx, U=ltshipout.dtx, V=ltoutput.dtx, W=lthyphen.dtx, X=ltfinal.dtx

__cmd_normalize_type_>:w [g454](#)
 __cmd_normalize_type_b:w [g565](#)
 __cmd_normalize_type_D:w
 [g434](#), [g442](#), [g444](#), [g495](#)
 __cmd_normalize_type_d:w [g429](#)
 __cmd_normalize_type_E:w [g439](#), [g495](#)
 __cmd_normalize_type_e:w [g429](#)
 __cmd_normalize_type_m:w [g543](#)
 __cmd_normalize_type_O:w [g429](#)
 __cmd_normalize_type_o:w [g429](#)
 __cmd_normalize_type_R:w [g448](#), [g543](#)
 __cmd_normalize_type_r:w [g429](#)
 __cmd_normalize_type_s:w [g429](#)
 __cmd_normalize_type_t:w [g451](#), [g495](#)
 __cmd_normalize_type_v:w [g543](#)
 \l__cmd_obey_spaces_bool [105](#), [126](#),
 [g32](#), [g378](#), [g484](#), [g490](#), [g534](#), [g539](#),
 [g668](#), [g672](#), [g689](#), [g725](#), [g810](#), [g815](#)
 __cmd_peek_cs_check_equal:NNN [g1837](#)
 __cmd_peek_meaning:NTF [g1828](#), [g1837](#)
 __cmd_peek_meaning_aux:NNTF . [g1837](#)
 __cmd_peek_meaning_remove:NTF ..
 [g999](#), [g1004](#),
 [g1105](#), [g1111](#), [g1238](#), [g1830](#), [g1837](#)
 __cmd_peek_nonspace:NTF [g1827](#)
 __cmd_peek_nonspace_aux:nNNTF [g1827](#)
 __cmd_peek_nonspace_remove:NTF .
 [g989](#), [g994](#),
 [g1093](#), [g1099](#), [g1226](#), [g1236](#), [g1827](#)
 __cmd_peek_true_remove:NNw .. [g1837](#)
 __cmd_peek_true_remove:Nw
 [g1848](#), [g1852](#), [g1860](#), [g1864](#)
 \l__cmd_prefixed_bool
 [105](#), [g36](#), [g702](#), [g719](#), [g726](#), [g732](#), [g763](#)
 __cmd_prepare_signature:N
 [124](#), [g685](#),
 [g742](#), [g750](#), [g758](#), [g766](#), [g774](#), [g782](#),
 [g789](#), [g845](#), [g855](#), [g897](#), [g910](#), [g923](#)
 __cmd_prepare_signature:n [g88](#), [g685](#)
 __cmd_prepare_signature_-
 bypass:N [124](#), [g685](#), [g720](#), [g727](#), [g735](#)
 \l__cmd_process_all_tl
 [105](#), [113](#), [g37](#), [g125](#),
 [g241](#), [g248](#), [g311](#), [g693](#), [g797](#), [g816](#)
 \l__cmd_process_one_tl
 [105](#), [g38](#), [g694](#), [g734](#), [g817](#), [g818](#)
 \l__cmd_process_some_bool
 [105](#), [g39](#), [g124](#), [g695](#), [g733](#)
 __cmd_put_arg_expandable:nw ...
 [g1439](#), [g1444](#),
 [g1445](#), [g1480](#), [g1485](#), [g1486](#), [g1593](#)
 __cmd_run_code:
 [111](#), [g242](#), [g245](#), [g967](#), [g975](#), [g983](#),
 [g986](#), [g991](#), [g996](#), [g1001](#), [g1089](#),
 [g1095](#), [g1101](#), [g1107](#), [g1131](#), [g1146](#),
 [g1153](#), [g1164](#), [g1166](#), [g1172](#), [g1178](#),
 [g1184](#), [g1190](#), [g1196](#), [g1202](#), [g1208](#),
 [g1219](#), [g1221](#), [g1239](#), [g1261](#), [g1400](#)
 \l__cmd_saved_args_tl
 [106](#), [g40](#), [g971](#), [g979](#)
 __cmd_show_arg_spec:N . [g1730](#), [g2258](#)
 __cmd_show_arg_spec:n . [g1730](#), [g2260](#)
 \l__cmd_signature_tl
 .. [106](#), [131](#), [142](#), [g41](#), [g118](#), [g161](#),
 [g696](#), [g749](#), [g757](#), [g773](#), [g781](#), [g795](#),
 [g804](#), [g927](#), [g970](#), [g1015](#), [g1121](#),
 [g1131](#), [g1148](#), [g1155](#), [g1164](#), [g1168](#),
 [g1174](#), [g1180](#), [g1186](#), [g1192](#), [g1198](#),
 [g1204](#), [g1210](#), [g1241](#), [g1263](#), [g1400](#)
 __cmd_single_token_check:n [g498](#),
 [g499](#), [g509](#), [g531](#), [g552](#), [g553](#), [g581](#)
 \l__cmd_some_long_bool [106](#),
 [115](#), [g43](#), [g155](#), [g163](#), [g381](#), [g652](#), [g663](#)
 \l__cmd_some_obey_spaces_bool ...
 [106](#), [g42](#), [g380](#), [g491](#), [g676](#)
 \l__cmd_some_short_bool
 [106](#), [115](#), [g44](#), [g154](#), [g165](#), [g382](#), [g664](#)
 __cmd_split_argument:nnn
 [g1636](#), [g2245](#)
 __cmd_split_argument_aux:n .. [g1636](#)
 __cmd_split_argument_aux:nnnn [g1636](#)
 __cmd_split_argument_aux:wn . [g1636](#)
 __cmd_split_list:nn
 [g1601](#), [g1638](#), [g2246](#)
 __cmd_split_list_multi:nn ... [g1601](#)
 \l__cmd_split_list_seq
 [149](#), [g1601](#), [g1615](#), [g1617](#)
 __cmd_split_list_single:Nn .. [g1601](#)
 \l__cmd_split_list_tl
 [149](#), [g1602](#), [g1625](#), [g1631](#), [g1633](#)
 __cmd_start:nNNnnn
 [108](#), [154](#), [g114](#), [g215](#), [g1821](#)
 __cmd_start_aux:NNnnnn
 [g221](#), [g232](#), [g235](#)
 __cmd_start_env:nnnnn
 [154](#), [g110](#), [g215](#), [g1823](#)
 __cmd_start_expandable:nNNNNn ..
 [108](#), [154](#), [g148](#), [g330](#), [g1822](#)
 __cmd_tl_mapthread_function:NNN
 [g281](#), [g309](#), [g1784](#)
 __cmd_tl_mapthread_function:nnN
 [g357](#), [g1784](#)
 __cmd_tl_mapthread_loop:w ... [g1784](#)

File Key: a=ltldirchk.dtx, b=ltplain.dtx, c=ltvers.dtx, d=ltluatex.dtx, e=ltexpl.dtx,
 f=ltdefs.dtx, g=ltcmd.dtx, h=lthooks.dtx, i=ltcmdhooks.dtx, j=ltalloc.dtx, k=ltcntrl.dtx,
 l=lterror.dtx, m=ltpar.dtx, n=ltpara.dtx, o=ltspace.dtx, p=ltlogos.dtx, q=ltfiles.dtx,
 r=ltoutenc.dtx, s=ltcounts.dtx, t=ltlength.dtx, u=ltfssbas.dtx, v=ltfssaxes.dtx,
 w=ltfssstrc.dtx, x=ltfsscmp.dtx, y=ltfssdcl.dtx, z=ltfssini.dtx, A=fontdef.dtx,
 B=preload.dtx, C=ltfntcmd.dtx, D=lttextcomp.dtx, E=ltpageno.dtx, F=ltxref.dtx,
 G=ltmiscen.dtx, H=ltmath.dtx, I=ltlists.dtx, J=ltboxes.dtx, K=lttab.dtx, L=ltpictur.dtx,
 M=ltthm.dtx, N=ltsect.dtx, O=ltfloat.dtx, P=ltidxglo.dtx, Q=ltbibl.dtx, R=ltpage.dtx,
 S=ltclass.dtx, T=ltfilehook.dtx, U=ltshipout.dtx, V=ltoutput.dtx, W=lthyphen.dtx,
 X=ltfinal.dtx

<code>__cmd_tmp:w</code>	<code>r732, r760, r764, r767, r834, r837,</code>
.. 112 , 116 , 143 , g45 , g267 , g283 ,	<code>r895, r1245, z504, A337, A338,</code>
g325 , g327 , g429 , g453 , g932 , g938 ,	<code>A340, A459, A462, A466, A530,</code>
g956 , g1405 , g1424 , g1457 , g1476 ,	<code>A531, A532, A533, A534, A535,</code>
g1533 , g1557 , g1560 , g1584 , g1869	<code>A537, A538, A539, A540, A541,</code>
<code>\l__cmd_tmp_prop</code>	<code>A543, D106, D935, H155, H157,</code>
..... 106 , g45 , g1117 , g1120 , g1126	<code>H158, H159, H175, H177, H178,</code>
<code>\l__cmd_tmpa_tl</code> .. 106 , g46 , g279 ,	<code>H179, H197, H198, K171, K172, L141</code>
g284 , g294 , g901 , g903 , g917 , g920 ,	create commands:
g1054 , g1057 , g1843 , g1868 , g1871	<code>create_callback</code>
<code>\l__cmd_tmpb_tl</code>	d661
..... 106 ,	<code>\create_callback</code>
129 , g47 , g886 , g891 , g902 , g1126 ,	45
g1127 , g1129 , g1844 , g1855 , g1861	<code>\CS</code>
<code>__cmd_token_if_cs:NTF</code>	75
..... 120 , g1078 , g1775 , g1850	cs commands:
<code>__cmd_trim_spaces:n</code> .. g1679 , g2247	<code>\cs:w</code>
<code>\l__cmd_v_arg_tl</code>	196 ,
..... 138 , 139 , g1250 , g1266 ,	197 , g1081 , h229 , h539 , h573 , h574 ,
g1285 , g1319 , g1375 , g1382 , g1389	h626 , h646 , h649 , h665 , h666 , h689 ,
<code>\l__cmd_v_nesting_int</code>	h690 , h691 , h698 , h705 , h947 , h957 ,
140 , g1321 , g1325 , g1341 , g1342 , g1351	h987 , h1038 , T209 , T223 , T235 , T236
<code>\colon</code>	<code>\cs_argument_spec:N</code>
<code>\columnsep</code> q27 , q96 , q153 , V81 , V202 239 , 241 , e138 , g1814 , i132 , i184
<code>\columnseprule</code>	<code>\cs_end:</code>
<code>\columnwidth</code> g1081 ,
..... q24 , h229 , h542 , h573 , h574 , h626 , h646 ,
..... q27 , q28 , q30 , q93 , q96 , q97 , q99 , h649 , h665 , h666 , h680 , h690 , h691 ,
..... q150 , q153 , q154 , q157 , J346 , J377 , h698 , h705 , h891 , h947 , h953 , h966 ,
..... J395 , O99 , O168 , O470 , O489 , h987 , h1038 , T209 , T223 , T235 , T236
..... V80 , V144 , V145 , V146 , V201 ,	<code>\cs_generate_from_arg_count:NNnn</code>
..... V202 , V203 , V204 , V205 , V1843 , g103 ,
..... V1845 , V2224 , V2228 , V2256 , V2262 g131 , g139 , g208 , g283 , g286 , g325
<code>\cong</code>	<code>\cs_generate_variant:Nn</code>
<code>\contentsline</code> 715 , N164 , N171 , N177 , N184 g93 , g244 , g286 ,
<code>\coprod</code> g1402 , g1594 , g1620 , h37 , h38 , h39 ,
<code>\copyright</code> h45 , h46 , h53 , h54 , h55 , h58 , h64 ,
<code>\cos</code> h68 , h288 , h668 , i14 , T426 , T434
<code>\cosh</code>	<code>\cs_gset:Npn</code>
<code>\cot</code> T240 , U172
<code>\coth</code>	<code>\cs_gset:Npx</code>
<code>\countdef</code> T211
..... a66 , b37 ,	<code>\cs_gset_eq:NN</code>
..... b38 , b39 , b41 , b51 , b90 , d75 , d85 , e135 , e136 , e137 , e138 , e139 , e140 ,
..... d174 , d182 , d190 , d198 , d217 , E3 , X61 e141 , g1716 , g2261 , h546 , h559 ,
<code>\counterwithin</code> h560 , h941 , i36 , U52 , U151 , U165 ,
..... 366 U166 , U171 , U181 , U185 , U321 , U378
<code>\counterwithin</code>	<code>\cs_gset_nopar:Npx</code>
..... s77 , s97 h48 , h50 , h52
<code>\counterwithout</code>	<code>\cs_gset_protected:Npn</code>
..... 366 h1238
<code>\counterwithout</code>	<code>\cs_gset_protected:Npx</code>
..... s67 , s94 h20 , U20
<code>\CountZero</code>	<code>\cs_if_eq:NNTF</code>
<code>\cr</code> g938 , g1137
..... b402 , r500 , r506 , r516 ,	<code>\cs_if_exist:NTF</code>
..... r522 , D101 , D105 , D930 , D934 , 819 , 820 , g63 , g180 ,
..... H175 , H179 , H364 , H410 , H508 , g941 , g1685 , g1692 , g2131 , g2144 ,
..... K192 , K203 , K210 , K219 , K224 , g2156 , g2165 , g2171 , g2176 , g2183 ,
..... K230 , K377 , L141 , L143 , L148 , L154 g2196 , g2209 , i53 , i61 , T93 , T403 , T407
<code>\crrc</code>	<code>\cs_if_exist_p:N</code>
..... b459 , r327 , r362 , r363 , r390 , r394 , g70 , g71
..... r397 , r476 , r480 , r484 , r486 , r489 ,	<code>\cs_if_exist_use:NTF</code>
 g409 , h487 , U313 , U314 , U318 , U319
	<code>\cs_new:Npn</code>
 g330 , g335 , g337 , g339 , g351 ,

File Key: a=ltldirchk.dtx, b=ltplain.dtx, c=ltvers.dtx, d=ltluatex.dtx, e=ltexpl.dtx, f=ltdefns.dtx, g=ltcmd.dtx, h=lthooks.dtx, i=ltcmdhooks.dtx, j=ltalloc.dtx, k=ltcntrl.dtx, l=lterror.dtx, m=ltpar.dtx, n=ltpara.dtx, o=ltspace.dtx, p=ltlogos.dtx, q=ltfiles.dtx, r=ltoutenc.dtx, s=ltcounts.dtx, t=ltlength.dtx, u=ltfssbas.dtx, v=ltfssaxes.dtx, w=ltfssstrc.dtx, x=ltfsscmp.dtx, y=ltfssdcl.dtx, z=ltfssini.dtx, A=fontdef.dtx, B=preload.dtx, C=ltfntcmd.dtx, D=lttextcomp.dtx, E=ltpageno.dtx, F=ltxref.dtx, G=ltmiscen.dtx, H=ltmath.dtx, I=ltlists.dtx, J=ltboxes.dtx, K=lttab.dtx, L=lt pictur.dtx, M=ltthm.dtx, N=ltsect.dtx, O=ltfloat.dtx, P=ltidxglo.dtx, Q=ltbibl.dtx, R=ltpage.dtx, S=ltclass.dtx, T=ltfilehook.dtx, U=ltshipout.dtx, V=ltoutput.dtx, W=lthyphen.dtx, X=ltfinal.dtx

g363, g369, g1068, g1403, g1407,
 g1425, g1432, g1433, g1455, g1459,
 g1477, g1490, g1492, g1494, g1496,
 g1502, g1519, g1521, g1523, g1525,
 g1527, g1529, g1531, g1535, g1558,
 g1562, g1585, g1587, g1593, g1671,
 g1673, g1784, g1797, g1803, g1826,
 g1875, g2218, g2235, g2236, g2240,
 g2241, g2242, h180, h186, h199,
 h209, h210, h212, h226, h232, h413,
 h415, h417, h473, h521, h537, h591,
 h592, h669, h670, h867, h951, h959,
 h967, h1244, h1245, i173, i300, n43,
 n46, n74, n86, T30, T38, T44,
 T57, T95, T100, T105, T112, T127,
 T232, T238, T376, T378, T380,
 T383, T387, T401, T417, T427,
 T543, U54, U59, U77, U141, U146,
 U162, U178, U183, U189, U192,
 U206, U257, U310, U323, U337,
 U340, U350, U383, U502, U503, U504
 \cs_new_eq:NN
 ... g48, g94, g1715, g2125, g2126,
 g2237, g2238, g2239, g2244, g2245,
 g2246, g2247, g2248, g2254, g2260,
 h7, h23, h36, h65, h500, h506, h544,
 h744, h745, h746, h747, h748, h749,
 h966, h1257, h1258, h1260, h1262,
 i232, n96, T250, T251, T470, T472,
 T474, T476, T478, T480, T482,
 T484, T486, T488, U7, U152,
 U347, U349, U402, U405, U406, U501
 \cs_new_protected:Npn
 847, g51, g56, g61, g82, g95, g101,
 g129, g175, g194, g215, g235, g245,
 g253, g264, g272, g277, g287, g292,
 g306, g315, g323, g372, g405, g431,
 g436, g441, g443, g445, g450, g454,
 g468, g481, g495, g505, g520, g528,
 g543, g549, g559, g565, g581, g590,
 g609, g619, g631, g642, g647, g648,
 g674, g685, g700, g705, g715, g722,
 g729, g737, g744, g752, g760, g768,
 g776, g784, g791, g802, g820, g830,
 g835, g842, g847, g849, g857, g870,
 g883, g899, g905, g912, g914, g925,
 g930, g936, g954, g959, g961, g963,
 g965, g967, g975, g986, g991, g996,
 g1001, g1006, g1013, g1036, g1089,
 g1095, g1101, g1107, g1113, g1135,
 g1145, g1146, g1153, g1166, g1172,
 g1178, g1184, g1190, g1196, g1202,
 g1208, g1219, g1221, g1223, g1235,
 g1237, g1239, g1251, g1256, g1261,
 g1281, g1287, g1297, g1303, g1316,
 g1329, g1335, g1358, g1367, g1387,
 g1395, g1397, g1595, g1603, g1613,
 g1623, g1636, g1643, g1679, g1681,
 g1687, g1694, g1706, g1718, g1723,
 g1730, g1736, g1743, g1747, g1775,
 g1810, g1827, g1829, g1831, g1837,
 g1839, g1841, g1857, g1864, g2127,
 g2140, g2153, g2158, g2163, g2169,
 g2175, g2177, g2179, g2192, g2205,
 g2213, g2249, g2255, h8, h13, h18,
 h41, h43, h47, h49, h51, h56, h59,
 h66, h69, h71, h80, h92, h101,
 h103, h108, h110, h124, h126, h143,
 h148, h150, h152, h169, h175, h176,
 h177, h233, h242, h247, h255, h265,
 h267, h289, h313, h318, h323, h328,
 h337, h428, h430, h455, h457, h462,
 h478, h483, h495, h501, h507, h512,
 h515, h518, h519, h545, h562, h598,
 h671, h678, h687, h694, h701, h708,
 h716, h722, h733, h750, h757, h769,
 h774, h779, h781, h783, h869, h884,
 h889, h898, h914, h916, h927, h932,
 h940, h942, h964, h973, h982, h990,
 h1000, h1180, h1182, h1198, h1206,
 h1225, h1246, h1247, h1248, h1249,
 i12, i15, i17, i26, i34, i43, i57, i77,
 i97, i110, i117, i122, i127, i181,
 i216, i233, i259, n51, T49, T62,
 T70, T79, T206, T220, T357, T362,
 T385, T508, U8, U13, U18, U343, U464
 \cs_new_protected:Npx
 g226, g1214, g1322
 \cs_new_protected_nopar:Npn
 g1070, g1160
 \cs_prefix_spec:N e137, i157
 \cs_replacement_spec:N .. e139, g1817
 \cs_set:Npn 131,
 g141, g209, g283, g325, g932, i248, i265
 \cs_set:Npx g170, g860, g873, T209, T223
 \cs_set_eq:NN g211,
 g212, g956, g1169, g1175, g1181,
 g1187, g1193, g1199, g1205, g1211,
 g1360, g1869, h579, h580, h581,
 h582, h771, h776, i244, i245, i261,
 i262, i281, i313, n49, n50, n97, n98,
 n99, n101, n102, n103, n137, n138,
 n139, U24, U44, U83, U95, U125,
 U132, U408, U410, U412, U414, U416
 \cs_set_nopar:Npn 131
 \cs_set_nopar:Npx g146, g167,
 g172, g198, g206, g861, g874, h42, h44
 \cs_set_protected:Npn g105,

File Key: a=ltldirchk.dtx, b=ltplain.dtx, c=ltvers.dtx, d=ltluatex.dtx, e=ltexpl.dtx,
 f=ltdefns.dtx, g=ltcmd.dtx, h=lthooks.dtx, i=ltcmdhooks.dtx, j=ltalloc.dtx, k=ltcntrl.dtx,
 l=lterror.dtx, m=ltpar.dtx, n=ltpara.dtx, o=ltspace.dtx, p=ltlogos.dtx, q=ltfiles.dtx,
 r=ltoutenc.dtx, s=ltcounts.dtx, t=ltlength.dtx, u=ltfssbas.dtx, v=ltfssaxes.dtx,
 w=ltfssstrc.dtx, x=ltfsscmp.dtx, y=ltfssdcl.dtx, z=ltfssini.dtx, A=fontdef.dtx,
 B=preload.dtx, C=ltfntcmd.dtx, D=lttextcomp.dtx, E=ltpageno.dtx, F=ltxref.dtx,
 G=ltmiscen.dtx, H=ltmath.dtx, I=ltlists.dtx, J=ltboxes.dtx, K=lttab.dtx, L=ltpictur.dtx,
 M=ltthm.dtx, N=ltsect.dtx, O=ltfloat.dtx, P=ltidxglo.dtx, Q=ltbibl.dtx, R=ltpage.dtx,
 S=ltclass.dtx, T=ltfilehook.dtx, U=ltshipout.dtx, V=ltoutput.dtx, W=lthyphen.dtx,
 X=ltfinal.dtx

g429, g962, g966, g993, g1003, g1098, g1110, g1122, g1156, g1222, g1242, g1405, g1457, g1533, g1560, h871, h901, h1169, U46, U135, U403	\cyrb	r1501
\cs_set_protected_nopar:Npn	\CYRBYUS	r1502
g960, g964, g988, g998, g1092, g1104, g1149, g1220	\cyrbys	r1501
\cs_set_protected_nopar:Npx	\CYRC	r1502
g106, g146	\cyrc	r1502
\cs_to_str:N 105, 196, e135, g70, g71, g84, g1081, i92, i112, i115, i120, i125	\CYRCH	r1502
\cs_undefine:N	\cyrch	r1502
g2262, h129, h520, i39, T225	\CYRCHLDSC	r1502
cs\check@icr commands:	\cyrchldsc	r1502
\cs_gset_eq:NN 71	\CYRCHRDSC	r1503
\csc H21	\cyrchrdsc	r1502
\csname 169, 486, 586, 815, 819	\CYRCHVCRS	r1503
\csname\endcsname 815, 817	\cyrchvcrs	r1503
\cup A373	\CYRD	r1503
\CurrentFile 309, 780, 804, 806, 809, 812, 813, 817, 818, S824, T3, T67, T84, T166, T171, T174, T342, T346, T516, T517	\cyrd	r1503
\CurrentFilePath 804, 809, 812, 818, T3, T67, T83, T165, T343, T346	\CYRDELTA	r1503
\CurrentFilePathUsed	\cyrdelta	r1503
804, T3, T66, T81, T163, T340, T343	\CYRDJE	r1504
\CurrentFileUsed 780, 804, 806, 812, 818, S824, T3, T66, T82, T164, T340, T342, T515, T516	\cyrdje	r1504
\CurrentOption r1528, r1531, r1536, r1548, S14, S441, S452, S465, S466, S471, S484, S485, S487, S501, S502, S505, S519, S524, S525, S538, S543, S544, S557, S559, S565, S567, S579, S580, S581, S589, S590, S591, S857, S915, S1013, S1014, S1024	\CYRDZE	r1504
CurrentOption commands:	\cyrdze	r1504
\CurrentOption:	\CYRDZHE	r1504
S464, S483, S500, S518, S523, S537, S542, S578, S588, S1023	\cyrdzhe	r1504
\CYRA r1500	\CYRE	r1504
\cyra r1500, r1551	\cyre	r1504
\CYRABHCH r1500	\CYREPS	r1505
\cyrabhch r1500	\cyreps	r1504
\CYRABHCHDSC r1500	\CYREREV	r1505
\cyrabhchdsc r1500	\cyrerev	r1505
\CYRABHDZE r1501	\CYRERY	r1505
\cyrabhdze r1500	\cyrery	r1505
\CYRABHHA r1501	\CYRF	r1505
\cyrabhha r1501	\cyrf	r1505
\CYRAE r1501	\CYRFITA	r1506
\cyrae r1501	\cyrfita	r1505
\CYRB r1501	\CYRG	r1506
	\cyrg	r1506
	\CYRGDSC	r1506
	\cyrgdsc	r1506
	\CYRGDSCHCRS	r1506
	\cyrghcrs	r1507
	\CYRGHCRS	r1507
	\cyrghcrs	r1507
	\CYRGHK	r1507
	\cyrghk	r1507
	\CYRGUP	r1507
	\cyrgup	r1507
	\CYRH	r1507
	\cyrh	r1507
	\CYRHDSC	r1508
	\cyrhdsc	r1508
	\CYRHHCRS	r1508
	\cyrhhcrs	r1508
	\CYRHHK	r1508

File Key: a=ltldirchk.dtx, b=ltplain.dtx, c=ltvers.dtx, d=ltluatex.dtx, e=ltexpl.dtx, f=ltdefns.dtx, g=ltcmd.dtx, h=lthooks.dtx, i=ltcmdhooks.dtx, j=ltalloc.dtx, k=ltcntrl.dtx, l=lterror.dtx, m=ltpar.dtx, n=ltpara.dtx, o=ltspace.dtx, p=ltlogos.dtx, q=ltfiles.dtx, r=ltoutenc.dtx, s=ltcounts.dtx, t=ltlength.dtx, u=ltfssbas.dtx, v=ltfssaxes.dtx, w=ltfssstrc.dtx, x=ltfsscmp.dtx, y=ltfssdcl.dtx, z=ltfssini.dtx, A=fontdef.dtx, B=preload.dtx, C=ltfntcmd.dtx, D=lttextcomp.dtx, E=ltpageno.dtx, F=ltxref.dtx, G=ltmiscen.dtx, H=ltmath.dtx, I=ltlists.dtx, J=ltboxes.dtx, K=lttab.dtx, L=ltpictur.dtx, M=ltthm.dtx, N=ltsect.dtx, O=ltfloat.dtx, P=ltidxglo.dtx, Q=ltbibl.dtx, R=ltpage.dtx, S=ltclass.dtx, T=ltfilehook.dtx, U=ltshipout.dtx, V=ltoutput.dtx, W=lthyphen.dtx, X=ltfinal.dtx

<code>\cyrrhk</code>	r1508	<code>\cyrnlhk</code>	r1515
<code>\CYRHRDSN</code>	r1509	<code>\CYRO</code>	r1515
<code>\cyhrdsn</code>	r1508	<code>\cyro</code>	r1515
<code>\CYRI</code>	r1509	<code>\CYROTLD</code>	r1515
<code>\cyri</code>	r1509	<code>\cyrotld</code>	r1515
<code>\CYRIE</code>	r1509	<code>\CYRP</code>	r1515
<code>\cyrie</code>	r1509	<code>\cyrp</code>	r1515
<code>\CYRII</code>	r1509	<code>\CYRPHK</code>	r1516
<code>\cyrii</code>	r1509	<code>\cyrphk</code>	r1516
<code>\CYRISHRT</code>	r1509	<code>\CYRQ</code>	r1516
<code>\cyrishrt</code>	r1509	<code>\cyrq</code>	r1516
<code>\CYRISHRTDSC</code>	r1510	<code>\CYRR</code>	r1516
<code>\cyrishrtdsc</code>	r1510	<code>\cyrr</code>	r1516
<code>\CYRIZH</code>	r1510	<code>\CYRRDSC</code>	r1516
<code>\cyrizh</code>	r1510	<code>\cyrrdsc</code>	r1516
<code>\CYRJE</code>	r1510	<code>\CYRRHK</code>	r1517
<code>\cyrje</code>	r1510	<code>\cyr rhk</code>	r1516
<code>\CYRK</code>	r1510	<code>\CYRRTICK</code>	r1517
<code>\cyrk</code>	r1510	<code>\cyr rtick</code>	r1517
<code>\CYRKBEAK</code>	r1511	<code>\CYRS</code>	r1517
<code>\cyrkbeak</code>	r1511	<code>\cyrs</code>	r1517
<code>\CYRKDSC</code>	r1511	<code>\CYRSACRS</code>	r1517
<code>\cyrkdsc</code>	r1511	<code>\cyrsacrs</code>	r1517
<code>\CYRKHCRS</code>	r1511	<code>\CYRSCHWA</code>	r1518
<code>\cyrkhc rs</code>	r1511	<code>\cyrschwa</code>	r1518
<code>\CYRKHK</code>	r1512	<code>\CYRSDSC</code>	r1518
<code>\cyrk hk</code>	r1511	<code>\cyrsdsc</code>	r1518
<code>\CYRKVCRS</code>	r1512	<code>\CYRSEMISFTSN</code>	r1518
<code>\cyrkvc rs</code>	r1512	<code>\cyrsemisftsn</code>	r1518
<code>\CYRL</code>	r1512	<code>\CYRSFTSN</code>	r1519
<code>\cyrl</code>	r1512	<code>\cyrsftsn</code>	r1519
<code>\CYRLDSC</code>	r1512	<code>\CYRSH</code>	r1519
<code>\cyrl dsc</code>	r1512	<code>\cyrsh</code>	r1519
<code>\CYRLHK</code>	r1513	<code>\CYRSHCH</code>	r1519
<code>\cyrlhk</code>	r1512	<code>\cyrshch</code>	r1519
<code>\CYRLJE</code>	r1513	<code>\CYRSHHA</code>	r1519
<code>\cyrlje</code>	r1513	<code>\cyrshha</code>	r1519
<code>\CYRM</code>	r1513	<code>\CYRT</code>	r1520
<code>\cyr m</code>	r1513	<code>\cyrt</code>	r1520
<code>\CYRMDSC</code>	r1513	<code>\CYRTDSC</code>	r1520
<code>\cyrmdsc</code>	r1513	<code>\cyrt dsc</code>	r1520
<code>\CYRMHK</code>	r1513	<code>\CYRTETSE</code>	r1520
<code>\cyr mhk</code>	r1513	<code>\cyrtetse</code>	r1520
<code>\CYRN</code>	r1514	<code>\CYRTSHE</code>	r1520
<code>\cyr n</code>	r1514	<code>\cyrtshe</code>	r1520
<code>\CYRNDSC</code>	r1514	<code>\CYRU</code>	r1521
<code>\cyrndsc</code>	r1514	<code>\cyru</code>	r1521
<code>\CYRNG</code>	r1514	<code>\CYRUSHRT</code>	r1521
<code>\cyrng</code>	r1514	<code>\cyrushrt</code>	r1521
<code>\CYRNHK</code>	r1514	<code>\CYRV</code>	r1521
<code>\cyr nhk</code>	r1514	<code>\cyrv</code>	r1521
<code>\CYRNJE</code>	r1515	<code>\CYRW</code>	r1521
<code>\cyrnje</code>	r1514	<code>\cyrw</code>	r1521
<code>\CYRNLHK</code>	r1515	<code>\CYRY</code>	r1521

File Key: a=lt dirchk.dtx, b=lt plain.dtx, c=lt vers.dtx, d=lt luatex.dtx, e=lt expl.dtx, f=lt defs.dtx, g=lt cmd.dtx, h=lt hooks.dtx, i=lt cmdhooks.dtx, j=lt alloc.dtx, k=lt cntrl.dtx, l=lt error.dtx, m=lt par.dtx, n=lt para.dtx, o=lt space.dtx, p=lt logos.dtx, q=lt files.dtx, r=lt outenc.dtx, s=lt counts.dtx, t=lt length.dtx, u=lt fssbas.dtx, v=lt fssaxes.dtx, w=lt fsstrc.dtx, x=lt fsscmp.dtx, y=lt fssdcl.dtx, z=lt fssini.dtx, A=fontdef.dtx, B=preload.dtx, C=lt fntcmd.dtx, D=lt textcomp.dtx, E=lt pageno.dtx, F=lt xref.dtx, G=lt miscen.dtx, H=lt math.dtx, I=lt lists.dtx, J=lt boxes.dtx, K=lt tab.dtx, L=lt pictur.dtx, M=lt thm.dtx, N=lt sect.dtx, O=lt float.dtx, P=lt idxglo.dtx, Q=lt bibl.dtx, R=lt page.dtx, S=lt class.dtx, T=lt filehook.dtx, U=lt shipout.dtx, V=lt output.dtx, W=lt hyphen.dtx, X=lt final.dtx

\cyr	r1521	\debug_suspend:	188
\CYRYA	r1522	\DebugHooksOff	174, h1248 , h1286
\cyrYA	r1522	\DebugHooksOn	174, h1248 , h1285
\CYRYAT	r1522	\DebugShipoutsOff	830, U405 , U437
\cyrYAT	r1522	\DebugShipoutsOn	830, U405 , U436
\CYRYHCRS	r1522	\DeclareCommandCopy	89, f460 , f461 , f463
\cyrYHCRS	r1522	\DeclareCurrentRelease	D812, D815, S1592
\CYRYI	r1522	\DeclareDefaultHookRule	172, h1252 , h1289
\cyrYI	r1522	\DeclareDocumentCommand	g2127
\CYRYO	r1523	\DeclareDocumentEnvironment	g2163
\cyrYO	r1522	\DeclareEmphSequence	496
\CYRYU	r1523	\DeclareEmphSequence	497, z394 , z430 , z431 , z443
\cyrYU	r1523	\DeclareEncodingSubset	D40, D397, D398, D399,
\CYRZ	r1523		D400, D401, D402, D403, D404,
\cyrZ	r1523		D405, D406, D407, D408, D409,
\CYRZDSC	r1523		D410, D411, D412, D413, D414,
\cyrZDSC	r1523		D415, D416, D417, D418, D419,
\CYRZH	r1523		D420, D421, D422, D423, D424,
\cyrZH	r1523		D425, D427, D428, D429, D430,
\CYRZHDSC	r1524		D431, D432, D433, D434, D435,
\cyrZHDSC	r1524		D436, D437, D438, D439, D440,
			D441, D442, D443, D444, D445,
			D446, D447, D448, D449, D450,
			D451, D452, D453, D454, D455,
			D456, D457, D458, D459, D460,
			D461, D462, D463, D464, D465,
			D466, D467, D468, D469, D470,
			D471, D472, D473, D474, D475,
			D476, D477, D478, D479, D480,
			D481, D482, D483, D484, D485,
			D486, D487, D488, D489, D490,
			D491, D492, D493, D494, D495,
			D496, D497, D498, D499, D500,
			D501, D502, D503, D504, D505,
			D506, D507, D508, D509, D510,
			D511, D512, D513, D514, D515,
			D516, D517, D518, D519, D520,
			D521, D522, D523, D524, D525,
			D526, D527, D528, D529, D530,
			D531, D532, D533, D534, D535,
			D536, D537, D538, D539, D540,
			D541, D542, D543, D544, D545,
			D546, D547, D548, D549, D550,
			D551, D552, D553, D554, D555,
			D556, D557, D558, D559, D560,
			D561, D562, D563, D564, D565,
			D566, D567, D568, D569, D570,
			D571, D572, D573, D574, D575,
			D620, D820, D821, D822, D823,
			D865 , D872 , D873 , D874 , D875 ,
			D1151, D1152, D1153, D1154,
			D1155, D1156, D1157, D1158,
D			
\d	r235, r395,		
	r482 , r765 , r1241 , r1455 , r1456 ,		
	r1457 , r1458 , r1461 , r1462 , r1463 ,		
	r1464 , r1465 , r1466 , r1467 , r1468 ,		
	r1469 , r1470 , r1471 , r1472 , r1473 ,		
	r1474 , r1475 , r1476 , r1477 , r1478 ,		
	r1479 , r1480 , r1481 , r1482 , r1483 ,		
	r1484 , r1485 , r1486 , r1487 , r1488 ,		
	r1489 , r1490 , r1491 , r1492 , r1493 , r1494		
\dag	r312		
\dagger	r312, s131, s137, s145, s146, A375		
\dashbox	L309 , L808 , L825		
\dashv	A403		
\date	706		
\date	N9 , N25		
\day	a185 , c11 , S1135 , S1268 , S1357		
\dblfigrule	V767 , V2325		
\dblfloatpagefraction	O287 , O301 , V2302		
\dblfloatsep	V753 , V765 , V1640 , V1766 , V2309		
\dbltextfloatsep	V222 , V230 , V769 , V1639 , V1765 , V2309		
\dbltopfraction	O284 , O298 , V2301		
\ddag	r313		
\ddagger	r313, s132, s138, s145, s147, A374		
\ddot	A518		
\ddots	A513		
\deadcycles	q333 , q366 , G32 , G99 , G165 , V299		
debug commands:			
\debug_resume:	188		

File Key: a=ltldirchk.dtx, b=ltplain.dtx, c=ltvers.dtx, d=ltluatex.dtx, e=ltexpl.dtx, f=ltdefns.dtx, g=ltcmd.dtx, h=lt hooks.dtx, i=ltcmdhooks.dtx, j=ltalloc.dtx, k=ltcntrl.dtx, l=ltterror.dtx, m=ltpar.dtx, n=lt para.dtx, o=lt space.dtx, p=lt logos.dtx, q=lt files.dtx, r=ltoutenc.dtx, s=lt counts.dtx, t=lt length.dtx, u=ltfssbas.dtx, v=ltfssaxes.dtx, w=ltfssstrc.dtx, x=ltfsscmp.dtx, y=ltfssdcl.dtx, z=ltfssini.dtx, A=fontdef.dtx, B=preload.dtx, C=ltfntcmd.dtx, D=lttextcomp.dtx, E=ltpageno.dtx, F=ltxref.dtx, G=ltmiscen.dtx, H=ltmath.dtx, I=ltlists.dtx, J=ltboxes.dtx, K=lttab.dtx, L=lt pictur.dtx, M=ltthm.dtx, N=ltsect.dtx, O=ltfloat.dtx, P=ltidxglo.dtx, Q=ltbibl.dtx, R=lt page.dtx, S=ltclass.dtx, T=ltfilehook.dtx, U=ltshipout.dtx, V=ltoutput.dtx, W=lthyphen.dtx, X=ltfinal.dtx

D1159, D1160, D1161, D1162,
 D1163, D1164, D1165, D1166,
 D1167, D1168, D1169, D1170,
 D1171, D1172, D1173, D1174,
 D1175, D1176, D1177, D1178,
 D1179, D1180, D1181, D1182,
 D1183, D1184, D1185, D1186,
 D1187, D1188, D1189, D1190,
 D1191, D1192, D1193, D1194,
 D1195, D1196, D1197, D1198,
 D1199, D1200, D1201, D1202,
 D1203, D1204, D1205, D1206,
 D1207, D1208, D1209, D1210, D1211
 \DeclareErrorFont . [u477](#), y264, z588, A49
 \DeclareExpandableDocumentCommand [g2179](#)
 \DeclareFixedFont [u75](#)
 \DeclareFontEncoding
 r377, r463, r716, r738, r744, r830,
 r1039, [u118](#), A126, A127, A128, A129
 \DeclareFontEncodingDefaults
 [u168](#), x90, x91, A36
 \DeclareFontFamily [392](#), [u93](#), x85, x86
 \DeclareFontFamilySubstitution . . [u437](#)
 \DeclareFontSeriesChangeRule [403](#), v3,
 [v4](#), v6, v7, v8, v9, v10, v11, v12,
 v13, v14, v15, v16, v17, v18, v19,
 v20, v21, v22, v23, v24, v25, v26,
 v27, v28, v29, v30, v31, v32, v33,
 v34, v35, v36, v37, v38, v39, v40,
 v41, v42, v43, v44, v45, v46, v47,
 v48, v49, v50, v51, v52, v53, v54,
 v55, v56, v57, v58, v59, v60, v61,
 v62, v63, v64, v65, v66, v67, v68,
 v69, v70, v71, v72, v73, v74, v75,
 v76, v77, v78, v79, v80, v81, v82,
 v83, v84, v85, v86, v87, v88, v89,
 v90, v91, v92, v93, v94, v95, v96,
 v97, v98, v99, v100, v101, v102,
 v103, v104, v105, v106, v107, v108,
 v109, v110, v111, v112, v113, v114,
 v115, v116, v117, v118, v119, v120,
 v121, v122, v123, v124, v125, v126,
 v127, v128, v129, v130, v131, v132,
 v133, v134, v135, v136, v137, v138,
 v139, v140, v141, v142, v143, v144,
 v145, v146, v147, v148, v149, v150,
 v151, v152, v153, v154, v155, v156,
 v157, v158, v159, v160, v161, v162,
 v163, v164, v165, v166, v167, v168,
 v169, v170, v171, v172, v173, v174,
 v175, v176, v177, v178, v179, v180,
 v181, v182, v183, v184, v185, v186,
 v187, v188, v189, v190, v191, v192,
 v193, v194, v195, v196, v197, v198,
 v199, v200, v201, v202, v203, v204,
 v205, v206, v207, v208, v209, v210,
 v211, v212, v213, v214, v215, v216,
 v217, v218, v219, v220, v221, v222,
 v223, v224, v225, v226, v227, v228,
 v229, v230, v231, v232, v233, v234,
 v235, v236, v237, v238, v239, v240,
 v241, v242, v243, v244, v245, v246,
 v247, v248, v249, v250, v251, v252,
 v253, v254, v255, v256, v257, v258,
 v259, v260, v261, v262, v263, v264,
 v265, v266, v267, v268, v269, v270,
 v271, v272, v273, v274, v275, v276,
 v277, v278, v279, v280, v281, v282,
 v283, v284, v285, v286, v287, v288,
 v289, v290, v291, v292, v293, v294,
 v295, v296, v297, v298, v299, v300,
 v301, v302, v303, v304, v305, v306,
 v307, v308, v309, v310, v311, v312,
 v313, v314, v315, v316, v317, v318,
 v319, v320, v321, v322, v323, v324,
 v325, v326, v327, v328, v329, v330,
 v331, v332, v333, v334, v335, v336,
 v337, v338, v339, v340, v341, v342,
 v343, v344, v345, v346, v347, v348,
 v349, v350, v351, v352, v353, v354,
 v355, v356, v357, v358, v359, v360,
 v361, v362, v363, v364, v365, v366,
 v367, v368, v369, v370, v371, v372,
 v373, v374, v375, v376, v377, v378,
 v379, v380, v381, v382, v386, v388
 \DeclareFontSeriesDefault [482](#)
 \DeclareFontSeriesDefault [482](#),
 [483](#), [487](#), [488](#), [494](#), z34, [z35](#), z164, z166
 \DeclareFontShape [u18](#),
 u446, u447, u448, u449, u450, u451,
 u452, u453, u454, u455, u456, u457,
 u458, u459, u460, u461, u462, u463,
 u464, u465, u466, x25, x27, x81, x82
 \DeclareFontShapeChangeRule . . . v523,
 [v524](#), v541, v542, v543, v544, v545,
 v546, v547, v548, v549, v550, v551,
 v552, v553, v554, v555, v556, v557,
 v558, v559, v560, v561, v562, v563,
 v564, v565, v566, v567, v568, v569,
 v570, v571, v572, v573, v574, v575,
 v576, v577, v578, v579, v580, v581,
 v582, v583, v584, v585, v586, v587,
 v588, v589, v590, v591, v592, v593,
 v594, v595, v596, v597, v601, v603
 \DeclareFontSubstitution
 r745, r831, [u141](#), A26, A34,
 A37, A38, A130, A131, A132, A133
 \DeclareHookRule [165](#),

File Key: a=ltldirchk.dtx, b=ltplain.dtx, c=ltvers.dtx, d=ltluatex.dtx, e=ltexpl.dtx,
 f=ltdefs.dtx, g=ltcmd.dtx, h=lthooks.dtx, i=ltcmdhooks.dtx, j=ltalloc.dtx, k=ltcntrl.dtx,
 l=lterror.dtx, m=ltpar.dtx, n=ltpara.dtx, o=ltspace.dtx, p=ltlogos.dtx, q=ltfiles.dtx,
 r=ltoutenc.dtx, s=ltcounts.dtx, t=ltlength.dtx, u=ltfssbas.dtx, v=ltfssaxes.dtx,
 w=ltfssstrc.dtx, x=ltfsscmp.dtx, y=ltfssdcl.dtx, z=ltfssini.dtx, A=fontdef.dtx,
 B=preload.dtx, C=ltfntcmd.dtx, D=lttextcomp.dtx, E=ltpageno.dtx, F=ltxref.dtx,
 G=ltmiscen.dtx, H=ltmath.dtx, I=ltlists.dtx, J=ltboxes.dtx, K=lttab.dtx, L=ltpictur.dtx,
 M=ltthm.dtx, N=ltsect.dtx, O=ltfloat.dtx, P=ltidxglo.dtx, Q=ltbibl.dtx, R=ltpage.dtx,
 S=ltclass.dtx, T=ltfilehook.dtx, U=ltshipout.dtx, V=ltoutput.dtx, W=lthyphen.dtx,
 X=ltfinal.dtx

- 167, 172, 174, 176, [h1250](#), [h1288](#), [G40](#)
`\DeclareHookrule` 172
`\DeclareMathAccent` . [y605](#), [A516](#), [A517](#),
[A518](#), [A519](#), [A520](#), [A521](#), [A522](#),
[A523](#), [A524](#), [A525](#), [A526](#), [A527](#), [A528](#)
`\DeclareMathAlphabet`
..... [x119](#), [x123](#), [x125](#), [x132](#),
[y431](#), [y594](#), [A151](#), [A152](#), [A153](#), [A154](#)
`\DeclareMathAlphabetCharacter` ... [y780](#)
`\DeclareMathDelimiter`
... [523](#), [y782](#), [A255](#), [A256](#), [A257](#),
[A258](#), [A259](#), [A260](#), [A263](#), [A265](#),
[A266](#), [A563](#), [A565](#), [A567](#), [A569](#),
[A571](#), [A574](#), [A576](#), [A578](#), [A580](#),
[A582](#), [A584](#), [A586](#), [A588](#), [A590](#),
[A592](#), [A594](#), [A596](#), [A598](#), [A600](#),
[A602](#), [A604](#), [A606](#), [A608](#), [A610](#), [A612](#)
`\DeclareMathRadical` [y917](#), [A529](#)
`\DeclareMathSizes`
..... [u205](#), [u211](#), [u233](#), [A157](#),
[A158](#), [A159](#), [A160](#), [A161](#), [A162](#),
[A163](#), [A164](#), [A165](#), [A166](#), [A167](#), [A168](#)
`\DeclareMathSizes*` [u205](#)
`\DeclareMathSymbol` .. [519](#), [y718](#), [y781](#),
[y798](#), [A169](#), [A170](#), [A171](#), [A172](#),
[A173](#), [A174](#), [A175](#), [A176](#), [A177](#),
[A178](#), [A179](#), [A180](#), [A181](#), [A182](#),
[A183](#), [A184](#), [A185](#), [A186](#), [A187](#),
[A188](#), [A189](#), [A190](#), [A191](#), [A192](#),
[A193](#), [A194](#), [A195](#), [A196](#), [A197](#),
[A198](#), [A199](#), [A200](#), [A201](#), [A202](#),
[A203](#), [A204](#), [A205](#), [A206](#), [A207](#),
[A208](#), [A209](#), [A210](#), [A211](#), [A212](#),
[A213](#), [A214](#), [A215](#), [A216](#), [A217](#),
[A218](#), [A219](#), [A220](#), [A221](#), [A222](#),
[A223](#), [A224](#), [A225](#), [A226](#), [A227](#),
[A228](#), [A229](#), [A230](#), [A231](#), [A232](#),
[A233](#), [A234](#), [A235](#), [A236](#), [A237](#),
[A238](#), [A239](#), [A240](#), [A241](#), [A242](#),
[A243](#), [A244](#), [A245](#), [A246](#), [A247](#),
[A248](#), [A249](#), [A250](#), [A251](#), [A261](#),
[A262](#), [A264](#), [A268](#), [A269](#), [A270](#),
[A271](#), [A272](#), [A273](#), [A274](#), [A275](#),
[A276](#), [A277](#), [A278](#), [A279](#), [A280](#),
[A281](#), [A282](#), [A283](#), [A284](#), [A285](#),
[A286](#), [A287](#), [A288](#), [A289](#), [A290](#),
[A291](#), [A292](#), [A293](#), [A294](#), [A295](#),
[A296](#), [A297](#), [A298](#), [A299](#), [A300](#),
[A301](#), [A302](#), [A303](#), [A304](#), [A305](#),
[A306](#), [A307](#), [A308](#), [A309](#), [A310](#),
[A311](#), [A312](#), [A313](#), [A314](#), [A315](#),
[A316](#), [A317](#), [A318](#), [A319](#), [A320](#),
[A321](#), [A322](#), [A323](#), [A324](#), [A325](#),
[A327](#), [A328](#), [A329](#), [A330](#), [A331](#),
[A332](#), [A333](#), [A334](#), [A341](#), [A342](#),
[A343](#), [A344](#), [A345](#), [A346](#), [A347](#),
[A349](#), [A350](#), [A351](#), [A352](#), [A353](#),
[A354](#), [A356](#), [A357](#), [A358](#), [A359](#),
[A360](#), [A361](#), [A364](#), [A365](#), [A366](#),
[A367](#), [A370](#), [A371](#), [A372](#), [A373](#),
[A374](#), [A375](#), [A376](#), [A377](#), [A378](#),
[A379](#), [A380](#), [A381](#), [A382](#), [A383](#),
[A384](#), [A385](#), [A386](#), [A387](#), [A388](#),
[A389](#), [A390](#), [A391](#), [A392](#), [A393](#),
[A394](#), [A395](#), [A396](#), [A397](#), [A398](#),
[A399](#), [A400](#), [A401](#), [A402](#), [A403](#),
[A404](#), [A405](#), [A406](#), [A407](#), [A408](#),
[A409](#), [A410](#), [A411](#), [A414](#), [A415](#),
[A418](#), [A419](#), [A420](#), [A421](#), [A422](#),
[A423](#), [A424](#), [A425](#), [A426](#), [A427](#),
[A428](#), [A429](#), [A430](#), [A432](#), [A433](#),
[A434](#), [A435](#), [A436](#), [A437](#), [A438](#),
[A441](#), [A442](#), [A443](#), [A445](#), [A446](#),
[A447](#), [A448](#), [A449](#), [A450](#), [A451](#),
[A452](#), [A453](#), [A454](#), [A455](#), [A477](#),
[A479](#), [A501](#), [A502](#), [A503](#), [A553](#),
[A554](#), [A555](#), [A556](#), [A614](#), [A615](#), [A616](#)
`\DeclareMathVersion` [y277](#), [z2](#), [z3](#)
`\DeclareOldFontCommand` [C125](#), [C141](#)
`\DeclareOption` 757
`\DeclareOption` [r1527](#), [w29](#), [w37](#),
[w45](#), [w53](#), [w56](#), [w60](#), [D820](#), [D821](#),
[D822](#), [D823](#), [D824](#), [D826](#), [D828](#),
[D830](#), [D831](#), [D872](#), [D873](#), [D874](#),
[D875](#), [D876](#), [D878](#), [D879](#), [S419](#), [S1034](#)
`\DeclareOption*` 757
`\DeclareOption*` [S419](#)
`\DeclarePreloadSizes` . [u185](#), [x95](#), [x96](#),
[B19](#), [B21](#), [B22](#), [B23](#), [B25](#), [B26](#), [B27](#),
[B28](#), [B29](#), [B30](#), [B34](#), [B38](#), [B43](#), [B45](#),
[B49](#), [B50](#), [B53](#), [B54](#), [B57](#), [B58](#), [B64](#)
`\DeclareRelease` [D811](#), [D814](#), [S1499](#)
DeclareRelease commands:
 `\DeclareRelease:` [S1502](#)
`\DeclareRobustCommand` . [85](#), [86](#), [89](#), [92](#),
[93](#), [95](#), [96](#), [335](#), [422](#), [586](#), [f221](#), [f728](#),
[f729](#), [f735](#), [f750](#), [l4](#), [l11](#), [l30](#), [l57](#), [o7](#),
[o8](#), [o9](#), [o10](#), [o11](#), [o68](#), [o92](#), [o330](#),
[o406](#), [o420](#), [o437](#), [o461](#), [p2](#), [p3](#),
[p13](#), [q405](#), [q518](#), [r150](#), [r158](#), [r307](#),
[r310](#), [r311](#), [r312](#), [r313](#), [r314](#), [r316](#),
[r318](#), [r320](#), [s155](#), [t19](#), [t20](#), [t21](#), [u251](#),
[u279](#), [u280](#), [u281](#), [u286](#), [u298](#), [u308](#),
[u316](#), [u318](#), [u336](#), [u641](#), [u650](#), [v394](#),
[v398](#), [v407](#), [v409](#), [v419](#), [v526](#), [v531](#),
[v536](#), [v615](#), [v618](#), [v626](#), [v627](#), [v633](#),
[v713](#), [w115](#), [w164](#), [z4](#), [z7](#), [z10](#), [z13](#),
[z16](#), [z19](#), [z22](#), [z25](#), [z28](#), [z198](#), [z218](#),

File Key: a=ltldirchk.dtx, b=ltplain.dtx, c=ltvers.dtx, d=ltluatex.dtx, e=ltexpl.dtx,
f=ltdefs.dtx, g=ltcmd.dtx, h=lthooks.dtx, i=ltcmdhooks.dtx, j=ltalloc.dtx, k=ltcntrl.dtx,
l=lterror.dtx, m=ltpar.dtx, n=ltpara.dtx, o=ltspace.dtx, p=ltlogos.dtx, q=ltfiles.dtx,
r=ltoutenc.dtx, s=ltcounts.dtx, t=ltlength.dtx, u=ltfssbas.dtx, v=ltfssaxes.dtx,
w=ltfssstrc.dtx, x=ltfsscmp.dtx, y=ltfssdcl.dtx, z=ltfssini.dtx, A=fontdef.dtx,
B=preload.dtx, C=ltfntcmd.dtx, D=lttextcomp.dtx, E=ltpageno.dtx, F=ltxref.dtx,
G=ltmiscen.dtx, H=ltmath.dtx, I=ltlists.dtx, J=ltboxes.dtx, K=lttab.dtx, L=ltpictur.dtx,
M=ltthm.dtx, N=ltsect.dtx, O=ltfloat.dtx, P=ltidxglo.dtx, Q=ltbibl.dtx, R=ltpage.dtx,
S=ltclass.dtx, T=ltfilehook.dtx, U=ltshipout.dtx, V=ltoutput.dtx, W=lthyphen.dtx,
X=ltfinal.dtx

z237, z242, z247, z279, z290, z301,
 z306, z311, z327, z330, z333, z336,
 z339, z353, z402, z403, z438, z444,
 z466, z468, z477, z479, z486, z502,
 z510, z528, z544, z561, A335, A336,
 A337, A348, A355, A412, A413,
 A444, A456, A460, A463, A468,
 A470, A472, A475, A478, A480,
 A481, A483, A485, A487, A489,
 A491, A493, A495, A497, A499,
 A505, A507, A509, A512, A530,
 A533, A536, A540, A544, A547,
 A550, A557, A560, A617, A618,
 A619, A624, A626, A628, A630,
 C3, C126, D4, D11, D610, D1136,
 F47, F59, G172, G228, G357, G362,
 G367, G377, G381, G385, G466,
 G470, H3, H4, H5, H6, H7, H8,
 H9, H10, H11, H12, H13, H14,
 H15, H16, H17, H18, H19, H20,
 H21, H22, H23, H24, H25, H26,
 H27, H28, H29, H30, H31, H32,
 H33, H34, H35, H39, H41, H42,
 H43, H44, H45, H46, H47, H48,
 H49, H50, H51, H52, H81, H82,
 H83, H84, H126, H154, H156, H160,
 H205, H207, H209, H211, H214,
 H215, H217, H224, H226, H227,
 H238, H261, H263, H279, H290,
 H340, H341, H342, H415, H432,
 H456, J7, J24, J120, J133, J156,
 J157, J173, J184, J238, J411, J429,
 J437, J473, J474, J475, J476, J477,
 J478, K139, K142, K154, L124,
 L127, L130, N7, N8, N9, N10,
 N14, N207, O402, O423, Q12, R22,
 R30, R53, R55, R57, R64, S990,
 S991, S992, S998, S999, S1622,
 T148, T182, U439, V87, X551, X561
 \DeclareSizeFunction
 w417, w490, w491,
 w502, w503, w507, w508, w514,
 w515, w543, w557, w558, w565, w566
 \DeclareSymbolFont
 x136, y312, A141, A142, A143, A144
 \DeclareSymbolFontAlphabet
 y991, A148, A149, A150
 \DeclareTextAccent r64,
 r378, r379, r380, r381, r382, r383,
 r384, r385, r386, r387, r388, r464,
 r465, r466, r467, r468, r469, r470,
 r471, r472, r473, r474, r741, r746,
 r747, r748, r749, r750, r751, r752,
 r753, r754, r755, r756, r838, r839,
 r840, r841, r842, r843, r844, r845,
 r846, r847, r848, r849, r850, r851, r852
 \DeclareTextAccentDefault
 r185, r226, r227, r228,
 r229, r230, r231, r232, r233, r234,
 r235, r236, r237, r238, r239, r279,
 r282, D687, D688, D940, D941,
 D942, D943, D944, D945, D946,
 D947, D948, D949, D950, D951, D952
 \DeclareTextCommand r3, r58, r65,
 r83, r389, r392, r395, r409, r410,
 r411, r414, r415, r422, r424, r426,
 r428, r434, r436, r438, r445, r475,
 r478, r482, r485, r487, r490, r492,
 r494, r510, r568, r569, r570, r730,
 r757, r759, r762, r765, r798, r805,
 r832, r835, r865, r893, r1044, r1070,
 r1072, r1074, r1156, r1158, r1160,
 r1207, r1244, D376, D377, D378,
 D379, D380, D381, D382, D383,
 D384, D385, D625, D632, D639, D759
 \DeclareTextCommandDefault r57, r186,
 r188, r283, r286, r287, r288, r290,
 r292, r296, r300, r301, r303, r304,
 r305, r306, r326, r355, D155, D157,
 D160, D162, D164, D166, D168,
 D170, D172, D174, D176, D178,
 D180, D182, D184, D186, D188,
 D190, D193, D194, D195, D196,
 D197, D198, D199, D200, D201,
 D202, D203, D204, D205, D206,
 D207, D208, D210, D212, D214,
 D216, D218, D220, D222, D224,
 D226, D228, D230, D232, D234,
 D236, D238, D240, D242, D244,
 D246, D248, D250, D252, D254,
 D256, D258, D260, D262, D264,
 D266, D268, D270, D272, D274,
 D276, D278, D280, D282, D284,
 D286, D288, D290, D292, D294,
 D296, D298, D300, D302, D304,
 D306, D309, D311, D313, D315,
 D317, D319, D321, D323, D325,
 D327, D329, D331, D333, D335,
 D337, D339, D341, D343, D345,
 D347, D349, D351, D353, D355,
 D357, D359, D361, D363, D365,
 D654, D662, D664, D670, D678,
 D1006, D1008, D1009, D1011,
 D1013, D1015, D1017, D1019,
 D1021, D1023, D1025, D1027,
 D1029, D1031, D1033, D1035,
 D1037, D1039, D1041, D1043,
 D1045, D1047, D1049, D1051,

File Key: a=ltldirchk.dtx, b=ltplain.dtx, c=ltvers.dtx, d=ltluatex.dtx, e=ltexpl.dtx,
 f=ltdefns.dtx, g=ltcmd.dtx, h=lthooks.dtx, i=ltcmdhooks.dtx, j=ltalloc.dtx, k=ltcntrl.dtx,
 l=ltterror.dtx, m=ltpar.dtx, n=ltpara.dtx, o=ltspace.dtx, p=ltlogos.dtx, q=ltfiles.dtx,
 r=ltoutenc.dtx, s=ltcounts.dtx, t=ltlength.dtx, u=ltfssbas.dtx, v=ltfssaxes.dtx,
 w=ltfssstrc.dtx, x=ltfsscmp.dtx, y=ltfssdcl.dtx, z=ltfssini.dtx, A=fontdef.dtx,
 B=preload.dtx, C=ltfntcmd.dtx, D=lttextcomp.dtx, E=ltpageno.dtx, F=ltxref.dtx,
 G=ltmiscen.dtx, H=ltmath.dtx, I=ltlists.dtx, J=ltboxes.dtx, K=lttab.dtx, L=ltpicture.dtx,
 M=ltthm.dtx, N=ltsect.dtx, O=ltfloat.dtx, P=ltidxglo.dtx, Q=ltbibl.dtx, R=ltpage.dtx,
 S=ltclass.dtx, T=ltfilehook.dtx, U=ltshipout.dtx, V=ltoutput.dtx, W=lthyphen.dtx,
 X=ltfinal.dtx

D1053, D1055, D1057, D1059,
 D1061, D1063, D1065, D1067,
 D1069, D1071, D1073, D1075,
 D1077, D1079, D1081, D1083,
 D1085, D1087, D1089, D1091,
 D1093, D1095, D1097, D1099,
 D1101, D1103, D1105, D1107,
 D1109, D1111, D1113, D1115,
 D1117, D1119, D1121, D1123, D1126
 \DeclareTextComposite [r76](#), [r452](#), [r453](#),
[r587](#), [r588](#), [r589](#), [r590](#), [r591](#), [r592](#),
[r593](#), [r594](#), [r595](#), [r596](#), [r597](#), [r598](#),
[r599](#), [r600](#), [r601](#), [r602](#), [r603](#), [r604](#),
[r605](#), [r606](#), [r607](#), [r608](#), [r609](#), [r610](#),
[r611](#), [r612](#), [r613](#), [r614](#), [r615](#), [r616](#),
[r617](#), [r618](#), [r619](#), [r620](#), [r621](#), [r622](#),
[r623](#), [r624](#), [r625](#), [r626](#), [r627](#), [r628](#),
[r629](#), [r630](#), [r631](#), [r632](#), [r633](#), [r634](#),
[r635](#), [r636](#), [r637](#), [r638](#), [r639](#), [r640](#),
[r641](#), [r642](#), [r643](#), [r644](#), [r645](#), [r646](#),
[r647](#), [r648](#), [r649](#), [r650](#), [r651](#), [r652](#),
[r653](#), [r654](#), [r655](#), [r656](#), [r657](#), [r658](#),
[r659](#), [r660](#), [r661](#), [r662](#), [r663](#), [r664](#),
[r665](#), [r666](#), [r667](#), [r668](#), [r669](#), [r670](#),
[r671](#), [r672](#), [r673](#), [r674](#), [r675](#), [r676](#),
[r677](#), [r678](#), [r679](#), [r680](#), [r681](#), [r682](#),
[r683](#), [r684](#), [r685](#), [r686](#), [r687](#), [r688](#),
[r689](#), [r690](#), [r691](#), [r692](#), [r693](#), [r694](#),
[r695](#), [r696](#), [r697](#), [r812](#), [r813](#), [r814](#),
[r815](#), [r816](#), [r817](#), [r818](#), [r819](#), [r820](#),
[r821](#), [r822](#), [r823](#), [r824](#), [r825](#), [r826](#), [r827](#)
 \DeclareTextCompositeCommand [r76](#), [r431](#),
[r454](#), [r455](#), [r456](#), [r457](#), [r459](#), [r698](#),
[r699](#), [r701](#), [r704](#), [r705](#), [r706](#), [r707](#),
[r708](#), [r709](#), [r710](#), [r711](#), [r712](#), [r795](#), [r1050](#)
 \DeclareTextFontCommand [C1](#), [C15](#), [C16](#),
[C17](#), [C18](#), [C19](#), [C20](#), [C21](#), [C22](#),
[C23](#), [C24](#), [C29](#), [C30](#), [C31](#), [C42](#), [C140](#)
 \DeclareTextSymbol [r3](#), [r398](#), [r399](#),
[r400](#), [r401](#), [r402](#), [r403](#), [r404](#), [r405](#),
[r406](#), [r407](#), [r408](#), [r412](#), [r413](#), [r416](#),
[r417](#), [r418](#), [r419](#), [r420](#), [r421](#), [r526](#),
[r527](#), [r528](#), [r529](#), [r530](#), [r531](#), [r532](#),
[r533](#), [r534](#), [r535](#), [r536](#), [r537](#), [r538](#),
[r539](#), [r540](#), [r541](#), [r543](#), [r544](#), [r545](#),
[r546](#), [r547](#), [r548](#), [r549](#), [r550](#), [r551](#),
[r552](#), [r553](#), [r554](#), [r555](#), [r556](#), [r557](#),
[r558](#), [r559](#), [r560](#), [r561](#), [r562](#), [r563](#),
[r564](#), [r565](#), [r566](#), [r567](#), [r571](#), [r572](#),
[r573](#), [r574](#), [r575](#), [r576](#), [r577](#), [r578](#),
[r579](#), [r580](#), [r581](#), [r582](#), [r583](#), [r584](#),
[r585](#), [r586](#), [r717](#), [r718](#), [r719](#), [r720](#),
[r721](#), [r722](#), [r723](#), [r724](#), [r725](#), [r726](#),
[r727](#), [r728](#), [r729](#), [r739](#), [r740](#), [r768](#),
[r769](#), [r770](#), [r771](#), [r772](#), [r773](#), [r774](#),
[r776](#), [r777](#), [r778](#), [r779](#), [r780](#), [r781](#),
[r782](#), [r783](#), [r784](#), [r785](#), [r786](#), [r787](#),
[r788](#), [r789](#), [r790](#), [r791](#), [r792](#), [r793](#),
[r794](#), [r853](#), [r854](#), [r855](#), [r856](#), [r857](#),
[r858](#), [r859](#), [r860](#), [r861](#), [r862](#), [r863](#),
[r864](#), [r876](#), [r877](#), [r878](#), [r879](#), [r880](#),
[r881](#), [r882](#), [r883](#), [r884](#), [r885](#), [r886](#),
[r887](#), [r888](#), [r889](#), [r890](#), [r891](#), [r892](#),
[r899](#), [r900](#), [r901](#), [r902](#), [r903](#), [r904](#),
[r905](#), [r906](#), [r907](#), [r908](#), [r909](#), [r910](#),
[r911](#), [r912](#), [r913](#), [r914](#), [r915](#), [r916](#),
[r917](#), [r918](#), [r919](#), [r920](#), [r921](#), [r922](#),
[r923](#), [r924](#), [r925](#), [r926](#), [r927](#), [r928](#),
[r929](#), [r930](#), [r931](#), [r932](#), [r933](#), [r934](#),
[r935](#), [r936](#), [r937](#), [r938](#), [r939](#), [r940](#),
[r941](#), [r942](#), [r943](#), [r944](#), [r945](#), [r946](#),
[r947](#), [r948](#), [r949](#), [r950](#), [r951](#), [r952](#),
[r953](#), [r954](#), [r955](#), [r956](#), [r957](#), [r958](#),
[r959](#), [r960](#), [r961](#), [r962](#), [r963](#), [r964](#),
[r965](#), [r966](#), [r967](#), [r968](#), [r969](#), [r970](#),
[r971](#), [r972](#), [r973](#), [r974](#), [r975](#), [r976](#),
[r977](#), [r978](#), [r1076](#), [r1077](#), [r1078](#),
[r1079](#), [r1080](#), [r1081](#), [r1082](#), [r1083](#),
[r1084](#), [r1085](#), [r1086](#), [r1087](#), [r1088](#),
[r1089](#), [r1090](#), [r1091](#), [r1092](#), [r1093](#),
[r1094](#), [r1095](#), [r1096](#), [r1098](#), [r1099](#),
[r1100](#), [r1101](#), [r1102](#), [r1103](#), [r1104](#),
[r1105](#), [r1106](#), [r1107](#), [r1108](#), [r1109](#),
[r1110](#), [r1111](#), [r1112](#), [r1114](#), [r1115](#),
[r1116](#), [r1117](#), [r1118](#), [r1119](#), [r1120](#),
[r1121](#), [r1122](#), [r1123](#), [r1124](#), [r1125](#),
[r1126](#), [r1127](#), [r1128](#), [r1129](#), [r1130](#),
[r1131](#), [r1132](#), [r1133](#), [r1134](#), [r1135](#),
[r1136](#), [r1137](#), [r1138](#), [r1139](#), [r1140](#),
[r1141](#), [r1142](#), [r1143](#), [r1144](#), [r1145](#),
[r1146](#), [r1147](#), [r1148](#), [r1149](#), [r1150](#),
[r1151](#), [r1152](#), [r1153](#), [r1154](#), [r1155](#),
[r1162](#), [r1163](#), [r1164](#), [r1165](#), [r1166](#),
[r1167](#), [r1168](#), [r1169](#), [r1170](#), [r1171](#),
[r1172](#), [r1173](#), [r1174](#), [r1175](#), [r1176](#),
[r1177](#), [r1178](#), [r1179](#), [r1180](#), [r1181](#),
[r1182](#), [r1183](#), [r1184](#), [r1185](#), [r1186](#),
[r1187](#), [r1188](#), [r1189](#), [r1190](#), [r1191](#),
[r1192](#), [r1193](#), [r1194](#), [r1195](#), [r1196](#),
[r1197](#), [r1198](#), [r1199](#), [r1200](#), [r1201](#),
[r1202](#), [r1203](#), [r1205](#), [r1206](#), [r1218](#),
[r1219](#), [r1220](#), [r1221](#), [r1222](#), [r1223](#),
[r1224](#), [r1225](#), [r1226](#), [r1227](#), [r1228](#),
[D368](#), [D369](#), [D370](#), [D371](#), [D372](#),
[D373](#), [D374](#), [D375](#), [D386](#), [D387](#),
[D388](#), [D389](#), [D390](#), [D391](#), [D392](#),
[D393](#), [D394](#), [D395](#), [D589](#), [D590](#),

File Key: a=ltldirchk.dtx, b=ltplain.dtx, c=ltvers.dtx, d=ltluatex.dtx, e=ltexpl.dtx,
 f=ltdefns.dtx, g=ltcmd.dtx, h=lt hooks.dtx, i=ltcmdhooks.dtx, j=ltalloc.dtx, k=ltcntrl.dtx,
 l=ltterror.dtx, m=ltpar.dtx, n=ltpara.dtx, o=lt space.dtx, p=ltlogos.dtx, q=ltfiles.dtx,
 r=ltoutenc.dtx, s=ltcounts.dtx, t=ltlength.dtx, u=ltfssbas.dtx, v=ltfssaxes.dtx,
 w=ltfssstrc.dtx, x=ltfsscmp.dtx, y=ltfssdcl.dtx, z=ltfssini.dtx, A=fontdef.dtx,
 B=preload.dtx, C=ltfntcmd.dtx, D=lttextcomp.dtx, E=ltpageno.dtx, F=ltxref.dtx,
 G=ltmiscen.dtx, H=ltmath.dtx, I=ltlists.dtx, J=ltboxes.dtx, K=lttab.dtx, L=lt pictur.dtx,
 M=ltthm.dtx, N=ltsect.dtx, O=ltfloat.dtx, P=ltidxglo.dtx, Q=ltbibl.dtx, R=ltpage.dtx,
 S=ltclass.dtx, T=ltfilehook.dtx, U=ltshipout.dtx, V=ltoutput.dtx, W=lthyphen.dtx,
 X=ltfinal.dtx

D591, D592, D593, D594, D595, D596

`\DeclareTextSymbolDefault`

 [r185](#), [r240](#), [r241](#),
[r242](#), [r243](#), [r244](#), [r245](#), [r246](#), [r247](#),
[r248](#), [r249](#), [r250](#), [r251](#), [r252](#), [r253](#),
[r254](#), [r255](#), [r256](#), [r257](#), [r258](#), [r259](#),
[r260](#), [r261](#), [r262](#), [r263](#), [r264](#), [r265](#),
[r266](#), [r267](#), [r268](#), [r269](#), [r270](#), [r271](#),
[r272](#), [r273](#), [r274](#), [r275](#), [r276](#), [r277](#),
[r278](#), [r280](#), [r281](#), [r291](#), [D114](#), [D116](#),
[D118](#), [D120](#), [D121](#), [D122](#), [D123](#),
[D124](#), [D125](#), [D126](#), [D127](#), [D128](#),
[D129](#), [D130](#), [D131](#), [D132](#), [D133](#),
[D134](#), [D135](#), [D136](#), [D137](#), [D138](#),
[D139](#), [D140](#), [D141](#), [D142](#), [D143](#),
[D144](#), [D145](#), [D146](#), [D147](#), [D148](#),
[D149](#), [D150](#), [D151](#), [D152](#), [D153](#),
[D154](#), [D577](#), [D578](#), [D579](#), [D580](#),
[D581](#), [D582](#), [D583](#), [D584](#), [D597](#),
[D598](#), [D599](#), [D600](#), [D601](#), [D602](#),
[D603](#), [D604](#), [D623](#), [D624](#), [D642](#),
[D643](#), [D644](#), [D645](#), [D646](#), [D647](#),
[D648](#), [D650](#), [D953](#), [D954](#), [D955](#),
[D956](#), [D957](#), [D958](#), [D959](#), [D960](#),
[D961](#), [D962](#), [D963](#), [D964](#), [D965](#),
[D966](#), [D967](#), [D968](#), [D969](#), [D970](#),
[D971](#), [D972](#), [D973](#), [D974](#), [D975](#),
[D976](#), [D977](#), [D978](#), [D979](#), [D980](#),
[D981](#), [D982](#), [D983](#), [D984](#), [D985](#),
[D986](#), [D987](#), [D988](#), [D989](#), [D990](#),
[D991](#), [D992](#), [D993](#), [D994](#), [D995](#),
[D996](#), [D997](#), [D998](#), [D999](#), [D1000](#),
[D1001](#), [D1002](#), [D1003](#), [D1004](#), [D1005](#)

`\DeclareUnicodeAccent`

[r1043](#), [r1229](#), [r1230](#), [r1231](#), [r1232](#),
[r1233](#), [r1234](#), [r1235](#), [r1236](#), [r1237](#),
[r1238](#), [r1239](#), [r1240](#), [r1241](#), [r1242](#), [r1243](#)

`\DeclareUnicodeCharacter` [X370](#)

`\DeclareUnicodeComposite` [r1048](#), [r1248](#),
[r1249](#), [r1250](#), [r1251](#), [r1252](#), [r1253](#),
[r1254](#), [r1255](#), [r1256](#), [r1257](#), [r1258](#),
[r1259](#), [r1260](#), [r1261](#), [r1262](#), [r1263](#),
[r1264](#), [r1265](#), [r1266](#), [r1267](#), [r1268](#),
[r1269](#), [r1270](#), [r1271](#), [r1272](#), [r1273](#),
[r1274](#), [r1275](#), [r1276](#), [r1277](#), [r1278](#),
[r1279](#), [r1280](#), [r1281](#), [r1282](#), [r1283](#),
[r1284](#), [r1285](#), [r1286](#), [r1287](#), [r1288](#),
[r1289](#), [r1290](#), [r1291](#), [r1292](#), [r1293](#),
[r1294](#), [r1295](#), [r1296](#), [r1297](#), [r1298](#),
[r1299](#), [r1300](#), [r1301](#), [r1302](#), [r1303](#),
[r1304](#), [r1305](#), [r1306](#), [r1307](#), [r1308](#),
[r1309](#), [r1310](#), [r1311](#), [r1312](#), [r1313](#),
[r1314](#), [r1315](#), [r1316](#), [r1317](#), [r1318](#),
[r1319](#), [r1320](#), [r1321](#), [r1322](#), [r1323](#),
[r1324](#), [r1325](#), [r1326](#), [r1327](#), [r1328](#),
[r1329](#), [r1330](#), [r1331](#), [r1332](#), [r1333](#),
[r1334](#), [r1335](#), [r1336](#), [r1337](#), [r1338](#),
[r1339](#), [r1340](#), [r1341](#), [r1342](#), [r1343](#),
[r1344](#), [r1345](#), [r1346](#), [r1347](#), [r1348](#),
[r1349](#), [r1350](#), [r1351](#), [r1352](#), [r1353](#),
[r1354](#), [r1355](#), [r1356](#), [r1357](#), [r1358](#),
[r1359](#), [r1360](#), [r1361](#), [r1362](#), [r1363](#),
[r1364](#), [r1365](#), [r1366](#), [r1367](#), [r1368](#),
[r1369](#), [r1370](#), [r1371](#), [r1372](#), [r1373](#),
[r1374](#), [r1375](#), [r1376](#), [r1377](#), [r1378](#),
[r1379](#), [r1380](#), [r1381](#), [r1382](#), [r1383](#),
[r1384](#), [r1385](#), [r1386](#), [r1387](#), [r1388](#),
[r1389](#), [r1390](#), [r1391](#), [r1392](#), [r1393](#),
[r1394](#), [r1395](#), [r1396](#), [r1397](#), [r1398](#),
[r1399](#), [r1400](#), [r1401](#), [r1402](#), [r1403](#),
[r1404](#), [r1405](#), [r1406](#), [r1407](#), [r1408](#),
[r1409](#), [r1410](#), [r1411](#), [r1412](#), [r1413](#),
[r1414](#), [r1415](#), [r1416](#), [r1417](#), [r1418](#),
[r1419](#), [r1420](#), [r1421](#), [r1422](#), [r1423](#),
[r1424](#), [r1425](#), [r1426](#), [r1427](#), [r1428](#),
[r1429](#), [r1430](#), [r1431](#), [r1432](#), [r1433](#),
[r1434](#), [r1435](#), [r1436](#), [r1437](#), [r1438](#),
[r1439](#), [r1440](#), [r1441](#), [r1442](#), [r1443](#),
[r1444](#), [r1445](#), [r1446](#), [r1447](#), [r1448](#),
[r1449](#), [r1450](#), [r1451](#), [r1452](#), [r1453](#),
[r1454](#), [r1455](#), [r1456](#), [r1457](#), [r1458](#),
[r1459](#), [r1460](#), [r1461](#), [r1462](#), [r1463](#),
[r1464](#), [r1465](#), [r1466](#), [r1467](#), [r1468](#),
[r1469](#), [r1470](#), [r1471](#), [r1472](#), [r1473](#),
[r1474](#), [r1475](#), [r1476](#), [r1477](#), [r1478](#),
[r1479](#), [r1480](#), [r1481](#), [r1482](#), [r1483](#),
[r1484](#), [r1485](#), [r1486](#), [r1487](#), [r1488](#),
[r1489](#), [r1490](#), [r1491](#), [r1492](#), [r1493](#), [r1494](#)

`\def` [229](#), [377](#), [483](#), [489](#), [511](#), [622](#)

`\defaultshyphenchar` [b356](#), [f738](#), [f753](#)

`\defaultscriptratio` [u624](#), [u631](#)

`\defaultscriptscriptratio` [u625](#), [u631](#)

`\defaultskewchar` [b357](#)

`\deg` [H34](#)

`\delcode` [y915](#)

`\delimiter` [y828](#), [y898](#), [y909](#)

`\delimiterfactor` [b358](#)

`\delimitershortfall` [b368](#)

`\Delta` [A298](#)

`\delta` [A271](#)

`\depth` [J32](#), [J35](#)

`\det` [H30](#)

`\detokenize` [99](#), [307](#),
[812](#), [f527](#), [f545](#), [f609](#), [q253](#), [r998](#),
[r1041](#), [v434](#), [z416](#), [D44](#), [G130](#), [S785](#),
[S1106](#), [S1110](#), [S1238](#), [S1239](#), [S1243](#)

`\DH` [r528](#), [r1120](#), [X571](#)

`\dh` [r538](#), [r1126](#), [X571](#)

File Key: a=ltldirchk.dtx, b=ltplain.dtx, c=ltvers.dtx, d=ltluatex.dtx, e=ltexpl.dtx,
f=ltdefs.dtx, g=ltcmd.dtx, h=lt hooks.dtx, i=ltcmdhooks.dtx, j=ltalloc.dtx, k=ltcntrl.dtx,
l=lterror.dtx, m=ltpar.dtx, n=lt para.dtx, o=lt space.dtx, p=lt logos.dtx, q=lt files.dtx,
r=ltoutenc.dtx, s=lt counts.dtx, t=lt length.dtx, u=ltfssbas.dtx, v=ltfssaxes.dtx,
w=ltfssstrc.dtx, x=ltfsscmp.dtx, y=ltfssdcl.dtx, z=ltfssini.dtx, A=fontdef.dtx,
B=preload.dtx, C=ltfntcmd.dtx, D=lttextcomp.dtx, E=ltpageno.dtx, F=ltxref.dtx,
G=ltmiscen.dtx, H=ltmath.dtx, I=ltlists.dtx, J=ltboxes.dtx, K=lttab.dtx, L=lt pictur.dtx,
M=ltthm.dtx, N=ltsect.dtx, O=ltfloat.dtx, P=ltidxglo.dtx, Q=ltbibl.dtx, R=lt page.dtx,
S=ltclass.dtx, T=ltfilehook.dtx, U=ltshipout.dtx, V=ltoutput.dtx, W=lt hyphen.dtx,
X=ltfinal.dtx

- \Diamond z580
 - \diamond A380
 - \diamondsuit A332
 - \dim H28
 - dim commands:
 - \dim_new:N ... U199, U200, U201, U202
 - \dim_set:Nn .. U193, U194, U195, U196
 - \dim_use:N U502, U503, U504
 - \c_max_dim ... U213, U239, U264, U291
 - \c_zero_dim n63, U219, U220, U221, U227, U245, U246, U247, U270, U271, U272, U299, U300, U301, U329, U331, U332
 - \dimendef .. b42, b43, b44, b52, b91, d218
 - \dimenzero d218
 - \dimexpr r503, r519, r869, r872, r1211, r1214, L13
 - \directlua a9, a12, a17, a20, a25, b65, b81, b105, b256, b338, b348, d2, d14, d29, d205, d223, d248, d253, d257, f34, f649, f734, r1011, S1091, S1222, X84, X86, X94, X99, X101
 - disable commands:
 - disable_callback d808
 - \disable_callback 45
 - \DisableHook . 166, 191, 233, h1150, h1272
 - \DiscardShipoutBox 826–828, 831, U82, U402, U435, U491
 - \discretionary f736, f751, f761, H238
 - \displaylines H190
 - displaymath (environment) H332
 - \displaymath H334
 - \displaystyle A532, A535, A539, A541, H62, H197, H358, H361, H414, H439, H451, H479, H503, H506
 - \displaywidowpenalty b327
 - \displaywidth ... H197, H357, H426, H482
 - \div A383
 - \DJ r529, r1130, X571
 - \dj r539, r1131, X571
 - \do a74, a75, a126, b13, b14, f69, g1360, k3, k7, k16, k26, q66, q69, q134, q137, q189, q192, q232, q303, q350, q489, q506, q649, q655, C90, G431, G452, G560, G570, G576, G582, J57, J76, K244, K269, L104, L116, L123, L203, L337, L339, L362, L365, L394, L396, L417, L422, L478, L506, L535, L731, L787, O65, O134, Q16, Q45, Q62, S237, S254, S464, S483, S500, S518, S523, S537, S542, S578, S588, S1023, S1060, S1138, S1191, S1271, S1360, S1637
 - \DocInput w8, A5, B5, W4
 - \document 299
 - document (environment) G8
 - \document 68, 183, 423, 488, 585, q9, G183, Q44, Q61
 - \documentclass 169, 170, 228, 755, d16, w2, A2, B2, S595, S602, S645, S648, S805, S932, S1043, W2
 - \documentstyle S600, S1043
 - \dospecials a74, a126, b13, g1361, G431, G452, G560, G570, S1138, S1271, S1360
 - \dot A525
 - \doteq A469
 - \dotfill b469, f776, f797
 - \dots r320, r322
 - \doublehyphendemerits b330
 - \doublerulesep K311, K338, K362
 - \Downarrow A586
 - \downarrow A580
 - \downbracefill A538, A557
 - \dump X636
- E**
- \E S1145, S1148, S1176, S1278, S1281, S1308, S1367, S1370, S1398
 - \edef 415, 416, 461, 488
 - \egroup b406
 - \eject b436
 - \ell A311
 - else commands:
 - \else: h526, h533, h541, h683, h954, h1040, n29, T239
 - \em 497, 498, z403, z430, C42
 - \emergencystretch R59, R65
 - \emforce 496
 - \emforce 497, 498, z399, z426, z435
 - \emminershape .. 496, 497, z407, z413, z430
 - \emph 497, C42
 - \empty b404
 - \emptyset A318
 - \emreset 497
 - \emreset 497, z399, z402, z433
 - \emrest z402
 - \encodingdefault ... 487, d252, d267, d275, r990, r1528, r1561, r1562, y269, z511, z529, z545, z562, A50
 - \end 131, 165, 182, 184, 185, 202, 229, 585–587, 623, a69, f23, f671, g969, g984, l200, w9, A6, B6, G190, G197, G199, G227, G253, G410, G411, H460, H469, N15, N17, S1153, S1157, S1164, S1286, S1290, S1296, S1375, S1379, S1385, U387, W5, I112

File Key: a=ltldirchk.dtx, b=ltplain.dtx, c=ltvers.dtx, d=ltluatex.dtx, e=ltexpl.dtx, f=ltdefs.dtx, g=ltcmd.dtx, h=lt hooks.dtx, i=ltcmdhooks.dtx, j=ltalloc.dtx, k=ltcntrl.dtx, l=lterror.dtx, m=ltpar.dtx, n=lt para.dtx, o=lt space.dtx, p=ltlogos.dtx, q=ltfiles.dtx, r=ltoutenc.dtx, s=ltcounts.dtx, t=ltlength.dtx, u=ltfssbas.dtx, v=ltfssaxes.dtx, w=ltfssstrc.dtx, x=ltfsscmp.dtx, y=ltfssdcl.dtx, z=ltfssini.dtx, A=fontdef.dtx, B=preload.dtx, C=ltfntcmd.dtx, D=lttextcomp.dtx, E=ltpageno.dtx, F=ltxref.dtx, G=ltmiscen.dtx, H=ltmath.dtx, I=ltlists.dtx, J=ltboxes.dtx, K=lttab.dtx, L=lt pictur.dtx, M=ltthm.dtx, N=ltsect.dtx, O=ltfloat.dtx, P=ltidxglo.dtx, Q=ltbibl.dtx, R=lt page.dtx, S=ltclass.dtx, T=ltfilehook.dtx, U=ltshipout.dtx, V=ltoutput.dtx, W=lt hyphen.dtx, X=ltfinal.dtx

- `\endarray` [K171](#)
`\endcenter` [G352](#)
`\endcsname` [169](#), [815](#), [817](#)
`\enddisplaymath` [H335](#)
`\enddocument` [180](#), [184](#), [579](#),
[581](#), [824](#), [845](#), [846](#), [G8](#), [G66](#), [G68](#), [U352](#)
`\endenumerate` [I240](#)
`\endeqnarray` [H366](#), [H413](#)
`\endequation` [H338](#)
`\endfilecontents` [S1047](#)
`\endflushleft` [G402](#)
`\endflushright` [G404](#)
`\endgraf` [268](#), [275](#), [277](#), [b401](#), [n101](#), [n139](#), [J94](#)
`\endgroup` [301](#), [585](#), [817](#)
`\EndIncludeInRelease` [a22](#), [a50](#),
[a306](#), [a316](#), [b87](#), [b101](#), [b118](#), [b123](#),
[b133](#), [b137](#), [b147](#), [b150](#), [b167](#), [b181](#),
[b185](#), [b219](#), [b224](#), [b232](#), [b240](#), [b288](#),
[b300](#), [b344](#), [b352](#), [b502](#), [b536](#), [b544](#),
[b569](#), [b609](#), [b614](#), [b624](#), [b629](#), [c68](#),
[d226](#), [d249](#), [d272](#), [d276](#), [e10](#), [e18](#),
[e27](#), [e110](#), [e131](#), [e143](#), [e152](#), [f12](#), [f18](#),
[f60](#), [f64](#), [f312](#), [f340](#), [f368](#), [f373](#), [f392](#),
[f406](#), [f447](#), [f457](#), [f487](#), [f496](#), [f510](#),
[f515](#), [f613](#), [f628](#), [f660](#), [f669](#), [f719](#),
[f726](#), [f748](#), [f759](#), [f762](#), [f790](#), [f811](#),
[h139](#), [h145](#), [h171](#), [h179](#), [h370](#), [h395](#),
[h1151](#), [h1159](#), [o20](#), [o30](#), [o64](#), [o77](#),
[o85](#), [o90](#), [o103](#), [o109](#), [o151](#), [o165](#),
[o177](#), [o188](#), [o205](#), [o217](#), [o251](#), [o268](#),
[o306](#), [o328](#), [o365](#), [o398](#), [o431](#), [o435](#),
[o444](#), [o450](#), [o464](#), [o471](#), [q82](#), [q139](#),
[q194](#), [q255](#), [q276](#), [q288](#), [q337](#), [q371](#),
[q409](#), [q432](#), [q455](#), [q473](#), [q480](#), [q499](#),
[q516](#), [q529](#), [q533](#), [q551](#), [q562](#), [q572](#),
[q611](#), [q638](#), [r103](#), [r123](#), [r168](#), [r175](#),
[r330](#), [r352](#), [r367](#), [r375](#), [s28](#), [s33](#), [s89](#),
[s100](#), [s142](#), [s148](#), [s168](#), [s171](#), [t10](#),
[t14](#), [u52](#), [u73](#), [u231](#), [u248](#), [u294](#),
[u304](#), [u314](#), [u430](#), [u435](#), [u470](#), [u475](#),
[u492](#), [u510](#), [u549](#), [u582](#), [u698](#), [u710](#),
[v384](#), [v390](#), [v403](#), [v415](#), [v422](#), [v505](#),
[v520](#), [v599](#), [v611](#), [v622](#), [v629](#), [v636](#),
[v709](#), [v727](#), [v734](#), [v738](#), [v741](#), [w157](#),
[w160](#), [w176](#), [w549](#), [w555](#), [x21](#), [x143](#),
[y77](#), [y105](#), [y168](#), [y198](#), [y230](#), [y654](#),
[y696](#), [y709](#), [y716](#), [y904](#), [y912](#), [z162](#),
[z184](#), [z266](#), [z321](#), [z347](#), [z383](#), [z392](#),
[z429](#), [z441](#), [z448](#), [z482](#), [z488](#), [z523](#),
[z539](#), [z556](#), [z571](#), [A80](#), [A90](#), [A109](#),
[A118](#), [A633](#), [A640](#), [C33](#), [C40](#), [F50](#),
[F61](#), [F72](#), [G64](#), [G103](#), [G111](#), [G117](#),
[G150](#), [G163](#), [G225](#), [G275](#), [G294](#),
[G308](#), [G317](#), [G327](#), [G334](#), [G344](#),
[G349](#), [G373](#), [G389](#), [G398](#), [G436](#),
[G457](#), [G488](#), [G498](#), [G516](#), [G531](#),
[G543](#), [G565](#), [G573](#), [H86](#), [H95](#),
[H117](#), [H124](#), [H143](#), [H147](#), [H162](#),
[H170](#), [H219](#), [H236](#), [H266](#), [H274](#),
[H303](#), [H330](#), [H393](#), [H402](#), [H442](#),
[H454](#), [H463](#), [H472](#), [J13](#), [J22](#), [J69](#),
[J86](#), [J101](#), [J113](#), [J124](#), [J131](#), [J188](#),
[J195](#), [J243](#), [J251](#), [J302](#), [J320](#), [J387](#),
[J404](#), [J413](#), [J418](#), [J441](#), [J448](#), [K64](#),
[K69](#), [K156](#), [K164](#), [K226](#), [K231](#),
[L15](#), [L19](#), [L38](#), [L47](#), [L68](#), [L76](#), [L88](#),
[L96](#), [L111](#), [L121](#), [L150](#), [L155](#), [L171](#),
[L182](#), [L249](#), [L263](#), [L368](#), [L425](#),
[L463](#), [L469](#), [L500](#), [L530](#), [L555](#),
[L571](#), [L582](#), [L595](#), [L603](#), [L624](#),
[L641](#), [L651](#), [L655](#), [L745](#), [L800](#),
[L819](#), [L836](#), [N19](#), [N29](#), [N167](#), [N173](#),
[N178](#), [N209](#), [N231](#), [O104](#), [O172](#),
[O231](#), [O246](#), [O293](#), [O306](#), [O351](#),
[O368](#), [O411](#), [O417](#), [O426](#), [O430](#),
[O439](#), [O445](#), [O449](#), [O480](#), [O498](#),
[O540](#), [O546](#), [Q56](#), [Q72](#), [R39](#), [R46](#),
[S21](#), [S26](#), [S51](#), [S63](#), [S86](#), [S95](#), [S101](#),
[S113](#), [S136](#), [S143](#), [S175](#), [S181](#), [S200](#),
[S211](#), [S249](#), [S266](#), [S284](#), [S296](#), [S317](#),
[S330](#), [S343](#), [S353](#), [S387](#), [S403](#), [S412](#),
[S444](#), [S454](#), [S494](#), [S510](#), [S532](#), [S547](#),
[S561](#), [S568](#), [S583](#), [S592](#), [S634](#), [S641](#),
[S720](#), [S747](#), [S775](#), [S907](#), [S964](#), [S994](#),
[S1001](#), [S1179](#), [S1310](#), [S1400](#), [T14](#),
[T23](#), [T88](#), [T140](#), [T177](#), [T189](#), [T198](#),
[T243](#), [T254](#), [T261](#), [T289](#), [T309](#),
[T322](#), [T326](#), [T349](#), [T369](#), [T394](#),
[T439](#), [T455](#), [T462](#), [T492](#), [T497](#),
[U420](#), [U457](#), [U474](#), [U478](#), [V53](#), [V62](#),
[V178](#), [V196](#), [V365](#), [V370](#), [V418](#),
[V464](#), [V650](#), [V708](#), [V811](#), [V830](#),
[V893](#), [V911](#), [V953](#), [V974](#), [V1216](#),
[V1385](#), [V1467](#), [V1561](#), [V1683](#),
[V1810](#), [V1929](#), [V1957](#), [V2177](#),
[V2195](#), [V2242](#), [V2286](#), [X15](#), [X19](#),
[X37](#), [X55](#), [X65](#), [X72](#), [X80](#), [X127](#),
[X132](#), [X184](#), [X208](#), [X278](#), [X283](#),
[X324](#), [X330](#), [X426](#), [X473](#), [I132](#), [I137](#)
`\EndIncludeRelease` [c75](#)
`\enditemize` [I251](#)
`\endline` [b401](#), [H175](#)
`\endlinechar` [582](#),
[a92](#), [a93](#), [a94](#), [a204](#), [f37](#), [f39](#), [f44](#),
[q586](#), [q620](#), [S358](#), [S359](#), [S360](#), [X315](#)
`\endlist` [I98](#), [I240](#), [I251](#)
`\endlrbox` [J155](#)
`\endmath` [H333](#)

File Key: a=ltldirchk.dtx, b=ltplain.dtx, c=ltvers.dtx, d=ltluatex.dtx, e=ltexpl.dtx,
f=ltdefns.dtx, g=ltcmd.dtx, h=lt hooks.dtx, i=ltcmdhooks.dtx, j=ltalloc.dtx, k=ltcntrl.dtx,
l=ltterror.dtx, m=ltpar.dtx, n=lt para.dtx, o=lt space.dtx, p=ltlogos.dtx, q=ltfiles.dtx,
r=ltoutenc.dtx, s=ltcounts.dtx, t=ltlength.dtx, u=ltfssbas.dtx, v=ltfssaxes.dtx,
w=ltfssstrc.dtx, x=ltfsscmp.dtx, y=ltfssdcl.dtx, z=ltfssini.dtx, A=fontdef.dtx,
B=preload.dtx, C=ltfntcmd.dtx, D=lttextcomp.dtx, E=ltpageno.dtx, F=ltxref.dtx,
G=ltmiscen.dtx, H=ltmath.dtx, I=ltlists.dtx, J=ltboxes.dtx, K=lttab.dtx, L=lt pictur.dtx,
M=ltthm.dtx, N=ltsect.dtx, O=ltfloat.dtx, P=ltidnglo.dtx, Q=ltbibl.dtx, R=ltpage.dtx,
S=ltclass.dtx, T=ltfilehook.dtx, U=ltshipout.dtx, V=ltoutput.dtx, W=lthyphen.dtx,
X=ltfinal.dtx

- `\endminipage` [J354](#)
- `\EndModuleRelease` [c99](#), [c100](#),
[c140](#), [g2267](#), [h1294](#), [i315](#), [n141](#), [D809](#)
- `\endpicture` [L49](#)
- `\endsloppypar` [R63](#)
- `\endtabbing` [K84](#)
- `\endtabular` [K171](#)
- `\endtabular*` [K171](#)
- `\endtrivlist` [G352](#), [G402](#), [G404](#),
[G460](#), [H484](#), [K85](#), [M39](#), [I100](#), [I101](#)
- `\endverbatim` [G459](#), [G486](#)
- `\enlargethispage` [V1855](#)
- `\enlargethispage*` [V1855](#)
- `\enskip` [o473](#)
- `\enspace` [o461](#), [o469](#)
- `\ensuremath`
[s144](#), [H415](#), [O409](#), [O416](#), [O437](#), [O444](#)
- `enumerate` (environment) [I231](#)
- `\enumerate` [I231](#)
- environments:
- `array` [K168](#)
- `center` [G351](#)
- `displaymath` [H332](#)
- `document` [G8](#)
- `enumerate` [I231](#)
- `eqnarray` [H344](#), [H485](#)
- `eqnarray*` [H411](#)
- `equation` [H336](#), [H473](#)
- `filecontents` [S1047](#), [755](#)
- `flushleft` [G401](#)
- `flushright` [G403](#)
- `itemize` [I242](#)
- `list` [I34](#)
- `lrbox` [629](#)
- `math` [H332](#)
- `minipage` [630](#)
- `picture` [L21](#)
- `sloppypar` [R62](#)
- `tabbing` [K71](#)
- `tabular` [K174](#)
- `thebibliography` [745](#)
- `trivlist` [I89](#)
- `verbatim` [G459](#)
- `verbatim*` [G483](#)
- `\epsilon` [A272](#)
- `eqnarray` (environment) [H344](#), [H485](#)
- `\eqnarray` [H349](#), [H412](#)
- `eqnarray*` (environment) [H411](#)
- `\eqno` [H338](#)
- `equation` (environment) [H336](#), [H473](#)
- `\equation` [H337](#)
- `\equiv` [A448](#)
- `\errhelp` [a217](#), [c31](#), [I39](#), [I66](#), [W12](#), [X293](#), [X626](#)
- `\errmessage` [a4](#),
[a58](#), [a222](#), [b164](#), [b178](#), [b304](#), [c32](#),
[d63](#), [e79](#), [e94](#), [I47](#), [I72](#), [u521](#), [u556](#),
[w425](#), [w525](#), [x65](#), [W16](#), [X49](#), [X295](#)
- `\ERROR` [143](#), [270](#),
[g1047](#), [g1057](#), [g1066](#), [g1419](#), [g1432](#),
[g1452](#), [g1471](#), [g1547](#), [g1574](#), [h669](#), [h670](#)
- `\errorcontextlines` . [b361](#), [b492](#), [b511](#),
[b527](#), [b542](#), [b556](#), [b580](#), [b597](#), [I163](#)
- `\errorstopmode` [173](#), [177](#), [b477](#), [X635](#)
- `\escapchar` [311](#)
- `\escapechar` [94](#),
[d204](#), [f126](#), [f169](#), [f173](#), [f181](#), [f275](#),
[f276](#), [f300](#), [f437](#), [f525](#), [f543](#), [q390](#),
[u367](#), [u591](#), [w229](#), [y58](#), [y86](#), [y147](#),
[y178](#), [y209](#), [y253](#), [T271](#), [T294](#), [T314](#)
- `\eta` [A274](#)
- `\etatcatcode` [d860](#)
- `\eTeXversion` [a57](#)
- `\evensidemargin` [V73](#), [V613](#), [V672](#)
- `\everycr` .. [b457](#), [H192](#), [H195](#), [H357](#), [H500](#)
- `\everydisplay` [u345](#), [u346](#), [u351](#)
- `\everyeof` [X315](#)
- `\everyjob` [c37](#), [c42](#), [c47](#),
[d208](#), [d254](#), [d255](#), [y273](#), [U29](#), [U30](#),
[X98](#), [X317](#), [X402](#), [X596](#), [X597](#), [X599](#)
- `\everymath` [u344](#), [u346](#), [u349](#)
- `\everypar` [262](#)
- `\everypar` [264–269](#), [272–274](#),
[277](#), [623](#), [n32](#), [n35](#), [n81](#), [n91](#), [n104](#),
[q43](#), [q112](#), [q170](#), [u644](#), [u657](#), [u704](#),
[G166](#), [G433](#), [G455](#), [J292](#), [J313](#),
[K81](#), [N48](#), [N96](#), [N107](#), [N127](#), [N136](#),
[O187](#), [V165](#), [V192](#), [V1151](#), [V1317](#),
[I129](#), [I131](#), [I135](#), [I136](#), [I180](#), [I197](#)
- `\EveryShipout` [831](#)
- `\ExecuteOptions` [w57](#), [w70](#), [D843](#), [D880](#), [S571](#)
- `\exhyphenpenalty` [b322](#), [b430](#)
- `\exists` [A324](#)
- `\exp` [H31](#)
- exp commands:
- `\exp_after:wN` [g251](#), [g267](#), [g321](#), [g327](#),
[g341](#), [g980](#), [g1016](#), [g1040](#), [g1056](#),
[g1074](#), [g1080](#), [g1084](#), [g1115](#), [g1149](#),
[g1156](#), [g1169](#), [g1175](#), [g1181](#), [g1187](#),
[g1193](#), [g1199](#), [g1205](#), [g1211](#), [g1242](#),
[g1370](#), [g1786](#), [g1787](#), [g1788](#), [g1789](#),
[g1790](#), [g1791](#), [g1792](#), [g1882](#), [h50](#),
[h57](#), [h62](#), [h228](#), [h229](#), [h573](#), [h682](#),
[h894](#), [h1175](#), [i167](#), [T73](#), [T234](#), [T235](#)
- `\exp_args:Nc` [g933](#), [g944](#), [g973](#),
[g1725](#), [g1738](#), [h929](#), [i23](#), [i54](#), [i112](#), [i113](#)
- `\exp_args:Ncc` [g131](#)

File Key: a=ltldirchk.dtx, b=ltplain.dtx, c=ltvers.dtx, d=ltluatex.dtx, e=ltexpl.dtx, f=ltdefns.dtx, g=ltcmd.dtx, h=lt hooks.dtx, i=ltcmdhooks.dtx, j=ltalloc.dtx, k=ltcntrl.dtx, l=lterror.dtx, m=ltpar.dtx, n=lt para.dtx, o=lt space.dtx, p=ltlogos.dtx, q=ltfiles.dtx, r=ltoutenc.dtx, s=ltcounts.dtx, t=ltlength.dtx, u=ltfssbas.dtx, v=ltfssaxes.dtx, w=ltfssstrc.dtx, x=ltfsscmp.dtx, y=ltfssdcl.dtx, z=ltfssini.dtx, A=fontdef.dtx, B=preload.dtx, C=ltfntcmd.dtx, D=lttextcomp.dtx, E=ltpageno.dtx, F=ltxref.dtx, G=ltmiscen.dtx, H=ltmath.dtx, I=ltlists.dtx, J=ltboxes.dtx, K=lttab.dtx, L=lt pictur.dtx, M=ltthm.dtx, N=ltsect.dtx, O=ltfloat.dtx, P=ltidnglo.dtx, Q=ltbibl.dtx, R=ltpage.dtx, S=ltclass.dtx, T=ltfilehook.dtx, U=ltshipout.dtx, V=ltoutput.dtx, W=lthyphen.dtx, X=ltfinal.dtx

- \exp_args:Ne T29, T92, T205, T249, T355, T375,
... h332, h333, h977, h978, T46, T97 T400, T469, T538, U5, U463, U500
- \exp_args:Nee T129 \extracolsep K167
- \exp_args:Nf \extrafloats b152, b189, b274
- g346, g1551, g1578, g1639, g1812,
g1814, g1881, i132, T32, T34, T359
- \exp_args:NNc i114, i119, i124
- \exp_args:NNNo g327, g1283
- \exp_args:Nnnv h116
- \exp_args:NNo h665
- \exp_args:NNV h639
- \exp_args:NNx h469, h586
- \exp_args:No
... g88, g338, g357, g366, g453,
g787, g1424, g1435, g1476, g1557,
g1584, g1754, g1780, h868, h912, i166
- \exp_args:Nof g1427
- \exp_args:NV g191, g290, T53
- \exp_args:Nv h811, h818, h892
- \exp_args:Nx g1753,
g1756, g1779, h1181, h1223, U29
- \exp_last_unbraced:Nf ... g1816, i90
- \exp_last_unbraced:NNNo .. h231, i96
- \exp_last_unbraced:NnNo g1651
- \exp_not:N g86, g115, g116,
g148, g150, g151, g157, g159, g198,
g200, g207, g228, g229, g230, g232,
g297, g796, g806, g866, g879, g903,
g921, g928, g1216, g1217, g1324,
g1326, g1327, g1392, h239, i88,
i90, i91, i92, i156, i157, i162, i165,
i169, i177, i190, i199, i251, i252,
i277, i278, n37, n38, U31, U95, U125
- \exp_not:n 112,
g110, g114, g118, g121, g125, g142,
g149, g161, g201, g203, g231, g297,
g300, g301, g535, g669, g817, g867,
g880, g891, g903, g920, g964, g966,
g979, g1025, g1026, g1069, g1129,
g1325, g1442, g1443, g1483, g1484,
g1666, g1675, h556, i162, i166, i200,
i202, i203, i205, i206, T209, T223, U30
- \exp_stop_f: g1429, h523, h531, h1038
- \expandafter 459, 586, 587, 815
- expandafter commands:
 \expandafter: C88, K269
- \expanded 197, S785, X97
- \ExplSyntaxOff 760, e142, g2268, h1295,
i195, i316, n142, T12, T86, T138,
T241, T252, T264, T367, T392,
T437, T490, T545, U418, U465, U505
- \ExplSyntaxOn
... e134, g8, h3, i3, i195, n3, T7,

File Key: a=ltldirchk.dtx, b=ltplain.dtx, c=ltvers.dtx, d=ltluatex.dtx, e=ltexpl.dtx, f=ltdefns.dtx, g=ltcmd.dtx, h=lt hooks.dtx, i=ltcmdhooks.dtx, j=ltalloc.dtx, k=ltcntrl.dtx, l=lterror.dtx, m=ltpar.dtx, n=lt para.dtx, o=lt space.dtx, p=ltlogos.dtx, q=ltfiles.dtx, r=ltoutenc.dtx, s=ltcounts.dtx, t=ltlength.dtx, u=ltfssbas.dtx, v=ltfssaxes.dtx, w=ltfssstrc.dtx, x=ltfsscmp.dtx, y=ltfssdcl.dtx, z=ltfssini.dtx, A=fontdef.dtx, B=preload.dtx, C=ltfntcmd.dtx, D=lttextcomp.dtx, E=ltpageno.dtx, F=ltxref.dtx, G=ltmiscen.dtx, H=ltmath.dtx, I=ltlists.dtx, J=ltboxes.dtx, K=lttab.dtx, L=lt pictur.dtx, M=ltthm.dtx, N=ltsect.dtx, O=ltfloat.dtx, P=ltidxglo.dtx, Q=ltbibl.dtx, R=lt page.dtx, S=ltclass.dtx, T=ltfilehook.dtx, U=ltshipout.dtx, V=ltoutput.dtx, W=lthyphen.dtx, X=ltfinal.dtx

- _filehook_file_pop_assign:nnnn
..... [T59](#), [T489](#)
- _filehook_file_push: ... [T59](#), [T485](#)
- _filehook_file_subst_begin:nnn
..... [818](#), [T377](#), [T387](#)
- _filehook_file_subst_cycle_
error:NN [820](#), [T422](#), [T427](#)
- _filehook_file_subst_loop:NN ..
..... [819](#), [T396](#)
- _filehook_file_subst_tortoise_
hare:nn [819](#), [T389](#), [T396](#)
- _filehook_full_name:nn [T30](#)
- _filehook_if_file_replaced:TF .
..... [819](#), [T382](#), [T479](#)
- _filehook_if_no_extension:nTF .
..... [T44](#), [T471](#)
- \g_filehook_input_file_seq [809](#), [T59](#)
- \l_filehook_internal_tl [T59](#)
- _filehook_log_file_record:n . [T508](#)
- \g_filehook_nesting_level_int ..
..... [T505](#), [T510](#), [T513](#), [T514](#), [T520](#)
- _filehook_normalize_file_
name:w [T371](#), [T477](#)
- _filehook_resolve_file_subst:w
..... [T371](#), [T475](#)
- _filehook_set_curr_file:nnn ...
..... [T357](#), [T473](#)
- _filehook_set_curr_file_
assign:nnnNN [T360](#), [T362](#)
- _filehook_subst_add:nn
..... [814](#), [T204](#), [T206](#), [T250](#)
- _filehook_subst_empty_name_
chk:NN [T206](#)
- _filehook_subst_file_normalize:Nn
..... [T206](#)
- _filehook_subst_remove:n
..... [T206](#), [T251](#)
- \filename@parse [1](#), [6](#)
- \filesize [e72](#), [e118](#)
- \fill [o454](#)
- \finalhyphendemerits
..... [b331](#), [G360](#), [G364](#), [G370](#)
- \firstmark [R51](#), [V647](#), [V706](#), [V2217](#)
- flag internal commands:
 \flag_filehook_file_replaced . [819](#)
 flag_filehook_file_replaced . [T382](#)
- flag commands:
 \flag_clear:n [T386](#)
 \flag_if_raised:nTF [T384](#), [T405](#)
 \flag_new:n [T382](#)
 \flag_raise:n [T406](#)
- \flat [A328](#)
- \floatingpenalty [O469](#), [O488](#)
- \floatpagefraction [O278](#), [V2298](#)
- \floatsep [V726](#),
 [V744](#), [V751](#), [V2104](#), [V2154](#), [V2303](#)
- \flushbottom [R55](#)
- flushleft (environment) [G401](#)
- \flushleft [G401](#)
- flushright (environment) [G403](#)
- \flushright [G403](#)
- \fmtname [c1](#),
 [c38](#), [c40](#), [c43](#), [c45](#), [c48](#), [c57](#), [S654](#), [S658](#)
- \fmtversion [c1](#), [c19](#), [c38](#), [c40](#),
 [c43](#), [c45](#), [c48](#), [c57](#), [c93](#), [c106](#), [c154](#),
 [d268](#), [l2](#), [r1530](#), [z160](#), [K1](#), [L1](#), [S168](#),
 [S178](#), [S671](#), [S674](#), [V4](#), [X580](#), [X606](#)
- \fnsymbol [366](#)
- \fnsymbol [s107](#)
- \font [b462](#),
 [b467](#), [f737](#), [f740](#), [f752](#), [f755](#), [r297](#),
 [r298](#), [r299](#), [r439](#), [r446](#), [r799](#), [r806](#),
 [r866](#), [r1003](#), [r1004](#), [r1051](#), [r1157](#),
 [r1159](#), [r1161](#), [r1208](#), [u81](#), [u87](#), [u89](#),
 [w84](#), [z406](#), [z439](#), [z445](#), [z471](#), [B8](#), [B9](#),
 [B10](#), [C85](#), [D6](#), [D612](#), [D626](#), [D633](#), [G454](#)
- \fontdimen [b462](#), [b467](#),
 [r297](#), [r298](#), [r299](#), [r439](#), [r446](#), [r799](#),
 [r806](#), [z406](#), [z439](#), [z445](#), [C85](#), [D6](#),
 [D612](#), [D626](#), [D633](#), [L126](#), [L129](#), [L679](#)
- \fontencoding [385](#), [d251](#), [d252](#),
 [d274](#), [d275](#), [r867](#), [r1561](#), [u251](#), [u286](#),
 [u298](#), [u308](#), [y269](#), [z511](#), [z529](#), [z545](#),
 [A16](#), [A24](#), [A67](#), [A74](#), [A83](#), [A85](#), [D36](#)
- \fontfamily
 . [385](#), [412](#), [492](#), [u279](#), [z6](#), [z9](#), [z12](#),
 [z80](#), [z91](#), [z335](#), [z338](#), [z341](#), [z591](#),
 [A58](#), [A69](#), [A76](#), [A87](#), [D28](#), [D30](#),
 [D32](#), [D34](#), [D85](#), [D87](#), [D89](#), [D91](#), [D916](#)
- \fontname [r1004](#), [u89](#)
- \fontseries
 [385](#), [412](#), [415](#), [420](#), [428](#), [489](#), [491](#),
 [492](#), [501](#), [503](#), [u279](#), [v393](#), [v394](#),
 [v405](#), [v407](#), [v417](#), [v419](#), [z15](#), [z18](#),
 [z210](#), [z211](#), [z212](#), [z213](#), [z229](#), [z230](#),
 [z231](#), [z232](#), [z282](#), [z283](#), [z284](#), [z285](#),
 [z293](#), [z294](#), [z295](#), [z296](#), [z329](#), [z332](#), [z592](#)
- \fontseriesforce .. [412](#), [v398](#), [v409](#), [v420](#)
- \fontshape [385](#), [420](#), [428](#), [503](#), [r449](#), [r809](#),
 [u279](#), [v528](#), [v533](#), [v538](#), [v614](#), [v615](#),
 [v624](#), [v626](#), [v631](#), [v633](#), [v684](#), [v688](#),
 [v691](#), [v694](#), [v697](#), [v700](#), [v703](#), [v706](#),
 [v713](#), [z21](#), [z24](#), [z27](#), [z30](#), [z593](#), [D636](#)
- \fontshapeforce .. [v618](#), [v627](#), [v634](#), [v714](#)
- \fontsize
 . [p6](#), [r302](#), [r328](#), [r360](#), [r868](#), [r1210](#),
 [r1246](#), [u79](#), [u318](#), [z494](#), [z594](#), [D104](#),
 [D672](#), [D933](#), [O409](#), [O416](#), [O437](#), [O444](#)

File Key: a=ltldirchk.dtx, b=ltplain.dtx, c=ltvers.dtx, d=ltluatex.dtx, e=ltexpl.dtx, f=ltdefns.dtx, g=ltcmd.dtx, h=lt hooks.dtx, i=ltcmdhooks.dtx, j=ltalloc.dtx, k=ltcntrl.dtx, l=lterror.dtx, m=ltpar.dtx, n=lt para.dtx, o=lt space.dtx, p=ltlogos.dtx, q=ltfiles.dtx, r=ltoutenc.dtx, s=ltcounts.dtx, t=ltlength.dtx, u=ltfssbas.dtx, v=ltfssaxes.dtx, w=ltfssstrc.dtx, x=ltfsscmp.dtx, y=ltfssdcl.dtx, z=ltfssini.dtx, A=fontdef.dtx, B=preload.dtx, C=ltfntcmd.dtx, D=lttextcomp.dtx, E=ltpageno.dtx, F=ltxref.dtx, G=ltmiscen.dtx, H=ltmath.dtx, I=ltlists.dtx, J=ltboxes.dtx, K=lttab.dtx, L=lt pictur.dtx, M=ltthm.dtx, N=ltsect.dtx, O=ltfloat.dtx, P=ltidxglo.dtx, Q=ltbibl.dtx, R=ltpage.dtx, S=ltclass.dtx, T=ltfilehook.dtx, U=ltshipout.dtx, V=ltoutput.dtx, W=lthyphen.dtx, X=ltfinal.dtx

- r397, r476, r477, r481, r484, r486,
- r489, r501, r506, r522, r760, r761,
- r764, r767, r834, r837, r1245, r1247
- \hline [K358](#), [K361](#)
- \hom [H29](#)
- hook commands:
- \hook_debug_off: [177](#), [h7](#), [h1249](#)
- \hook_debug_on: [177](#), [h7](#), [h1248](#)
- \hook_disable:n ... [175](#), [h122](#), [h1150](#)
- \hook_gput_code:n [192](#)
- \hook_gput_code:nnn
... [176](#), [198](#), [199](#), [h265](#), [h321](#), [h1161](#)
- \hook_gput_next_code:nn
..... [176](#), [199](#), [h326](#), [h914](#), [h1163](#)
- \hook_gremove_code:nn
..... [176](#), [203](#), [h428](#), [h1165](#)
- \hook_gset_rule:nnnn
... [176](#), [205](#), [h478](#), [h1251](#), [h1253](#), [h1256](#)
- \hook_if_empty:nTF [169](#),
[176](#), [h791](#), [h1002](#), [h1257](#), [U63](#), [U163](#)
- \hook_if_empty_p:n
... [176](#), [h836](#), [h1002](#), [U67](#), [U114](#), [U368](#)
- \hook_log:n [177](#), [h769](#), [h1247](#)
- \hook_new:n ... [175](#), [191](#), [192](#), [194](#),
[201](#), [223](#), [h69](#), [h109](#), [h382](#), [h1142](#),
[U153](#), [U154](#), [U155](#), [U156](#), [U157](#), [U158](#)
- \hook_new_pair:nn
..... [175](#), [h108](#), [h1144](#), [n6](#), [n7](#)
- \hook_new_reversed:n
..... [175](#), [h101](#), [h109](#), [h1143](#)
- \g_hook_patch_action_list_tl ...
..... [i6](#), [i71](#), [i104](#)
- \hook_provide:n .. [175](#), [194](#), [h146](#),
[h170](#), [h173](#), [h176](#), [h1146](#), [h1147](#), [h1153](#)
- \hook_provide_pair:nn
..... [175](#), [h169](#), [h177](#), [h1149](#)
- \hook_provide_reversed:n
... [175](#), [194](#), [h146](#), [h170](#), [h175](#), [h1148](#)
- \hook_show:n ... [177](#), [215](#), [h769](#), [h1246](#)
- \hook_use:n
... [169](#), [175](#), [179](#), [191](#), [192](#), [197](#),
[220](#), [221](#), [h559](#), [h942](#), [h997](#), [h1244](#),
[n20](#), [n28](#), [n59](#), [n66](#), [U62](#), [U65](#), [U71](#), [U75](#)
- \hook_use_once:n
... [169](#), [175](#), [179](#), [221](#), [h990](#), [h1245](#)
- hook internal commands:
- \g_hook_??_code_prop [h475](#)
- \g_hook_??_reversed_tl [h475](#)
- \g_hook_⟨hook⟩_code_prop
..... [189](#), [203](#), [222](#)
- \g_hook_⟨hook⟩_labels_clist ... [192](#)
- \g_hook_⟨hook⟩_reversed_tl ... [189](#)
- \g_hook_all_seq [h28](#), [h84](#), [h548](#)
- _hook_apply_rule_>:nnn ... [h744](#)
- _hook_apply_rule_<:nnn ... [h744](#)
- _hook_apply_rule_<:nnn [h744](#)
- _hook_apply_rule_>:nnn [h744](#)
- _hook_apply_rule_x:nnn [h744](#)
- _hook_apply_label_pair:nnn ...
..... [211](#), [213](#), [218](#), [h616](#), [h617](#), [h671](#)
- _hook_apply_rule:nnn [213](#), [h681](#), [h687](#)
- _hook_apply_rule:nnnN [213](#)
- _hook_apply_rule_>:nnn ... [h722](#)
- _hook_apply_rule_<:nnn ... [h722](#)
- _hook_apply_rule_<:nnn [h694](#)
- _hook_apply_rule_>:nnn [h694](#)
- _hook_apply_rule_xE:nnn ... [h708](#)
- _hook_apply_rule_xW:nnn ... [h708](#)
- _hook_clear_next:n [h916](#)
- _hook_clist_gput:Nn
..... [h580](#), [h582](#), [h640](#), [h669](#)
- _hook_cmd_begin_document_code: ...
..... [245](#), [i26](#), [i313](#)
- _hook_cmd_if_scanable:NnTF ...
..... [242](#), [244](#), [i220](#), [i242](#)
- _hook_cmd_patch_xparse:Nnn ...
..... [i108](#), [i122](#)
- _hook_cmd_try_patch:nn ... [i32](#), [i43](#)
- \l_hook_cur_hook_tl
..... [214](#), [h30](#), [h602](#), [h728](#), [h739](#)
- _hook_curr_name_pop:
..... [227](#), [h1166](#), [h1263](#)
- _hook_curr_name_push:n
..... [227](#), [228](#), [h1166](#)
- _hook_curr_name_push_aux:n [h1166](#)
- _hook_currname_or_default: ...
..... [195](#), [196](#), [h183](#), [h191](#),
[h195](#), [h211](#), [h212](#), [h297](#), [h1072](#), [h1102](#)
- _hook_debug:n [186](#),
[h7](#), [h291](#), [h547](#), [h552](#), [h564](#), [h586](#),
[h621](#), [h641](#), [h696](#), [h703](#), [h710](#), [h718](#),
[h724](#), [h735](#), [i13](#), [i20](#), [i28](#), [i29](#), [i45](#), [i49](#)
- \g_hook_debug_bool [h6](#), [h10](#), [h15](#), [h21](#)
- _hook_debug_gset: [h7](#)
- _hook_debug_label_data:N
..... [h621](#), [h662](#), [h757](#)
- _hook_debug_print_rules:n .. [h898](#)
- \g_hook_delayed_patches_prop ...
..... [i11](#), [i31](#), [i37](#), [i38](#)
- _hook_disable:n [h122](#)
- _hook_end_document_label_-
check: [h1166](#)
- \g_hook_execute_immediately_-
prop [221](#), [h32](#), [h269](#), [h1001](#)
- _hook_file_hook_normalize:n ...
..... [202](#), [h333](#), [h413](#), [h978](#)
- \l_hook_front_tl [h593](#), [h632](#), [h635](#),
[h638](#), [h640](#), [h641](#), [h642](#), [h655](#), [h656](#)

File Key: a=ltldirchk.dtx, b=ltplain.dtx, c=ltvers.dtx, d=ltluatex.dtx, e=ltexpl.dtx, f=ltdefns.dtx, g=ltcmd.dtx, h=lthooks.dtx, i=ltcmdhooks.dtx, j=ltalloc.dtx, k=ltcntrl.dtx, l=lterror.dtx, m=ltpar.dtx, n=ltpara.dtx, o=ltspace.dtx, p=ltlogos.dtx, q=ltfiles.dtx, r=ltoutenc.dtx, s=ltcounts.dtx, t=ltlength.dtx, u=ltfssbas.dtx, v=ltfssaxes.dtx, w=ltfssstrc.dtx, x=ltfsscmp.dtx, y=ltfssdcl.dtx, z=ltfssini.dtx, A=fontdef.dtx, B=preload.dtx, C=ltfntcmd.dtx, D=lttextcomp.dtx, E=ltpageno.dtx, F=ltxref.dtx, G=ltmiscen.dtx, H=ltmath.dtx, I=ltlists.dtx, J=ltboxes.dtx, K=lttab.dtx, L=ltpicture.dtx, M=ltthm.dtx, N=ltsect.dtx, O=ltfloat.dtx, P=ltidxglo.dtx, Q=ltbibl.dtx, R=ltpage.dtx, S=ltclass.dtx, T=ltfilehook.dtx, U=ltshipout.dtx, V=ltoutput.dtx, W=lthyphen.dtx, X=ltfinal.dtx

`\c_hook_generics_file_prop` [201](#), [h406](#), [h425](#)
`\c_hook_generics_prop` [h351](#), [h380](#), [h423](#)
`\c_hook_generics_reversed_ii_`
`prop` [h359](#), [h383](#), [h425](#)
`\c_hook_generics_reversed_iii_`
`prop` [h362](#), [h386](#), [h425](#)
`__hook_gput_code:nnn` [h265](#)
`__hook_gput_next_code:nn` [h915](#), [h916](#)
`__hook_gput_next_do:nn` [199](#), [h326](#), [h916](#)
`__hook_gput_next_do:Nnn` [h916](#)
`__hook_gput_undeclared_hook:nnn`
. [199](#), [h313](#), [h321](#)
`__hook_gremove_code:nn` [h428](#)
`__hook_gremove_code_do:nn` [203](#), [h445](#), [h455](#)
`__hook_gset_rule:nnnn` [h478](#)
`\g_hook_hook_curr_name_tl` [196](#), [226–228](#), [h34](#),
[h214](#), [h224](#), [h1166](#), [h1178](#), [h1193](#),
[h1194](#), [h1201](#), [h1211](#), [h1212](#), [h1232](#)
`__hook_hook_gput_code_do:nnn` [h116](#), [h265](#), [h316](#)
`__hook_if_declared:nTF` [190](#), [191](#), [h73](#), [h157](#), [h382](#), [h1030](#), [i47](#)
`__hook_if_declared_p:n` [h1030](#)
`__hook_if_disabled:nTF` [190](#), [191](#),
[194](#), [h122](#), [h154](#), [h281](#), [h789](#), [h862](#), [h918](#)
`__hook_if_disabled_p:n` [h122](#)
`__hook_if_file_hook:wTF` [200–202](#), [221](#), [h330](#), [h396](#), [h975](#)
`__hook_if_file_hook_p:w` [h396](#)
`__hook_if_label_case:nnnnn` [h537](#), [h614](#), [h876](#)
`__hook_if_marked_removal:nnTF` [h272](#), [h467](#)
`__hook_if_public_command:N` [237](#)
`__hook_if_public_command:NTF` [i57](#)
`__hook_if_public_command:w` [i57](#)
`__hook_if_reversed:nTF` [h578](#), [h805](#),
[h842](#), [h844](#), [h1036](#), [h1133](#), [h1137](#), [h1139](#)
`__hook_if_reversed_p:n` [h1036](#)
`__hook_if_structure_exist:nTF` [190](#), [191](#), [h94](#), [h432](#), [h1004](#), [h1024](#)
`__hook_if_structure_exist_p:n` [h1024](#)
`__hook_if_usable:nTF` . [190](#), [191](#),
[203](#), [h275](#), [h292](#), [h353](#), [h450](#), [h567](#),
[h787](#), [h804](#), [h860](#), [h922](#), [h1018](#), [h1258](#)
`__hook_if_usable_p:n` [h835](#), [h1018](#)
`__hook_if_usable_use:n` [221](#), [h967](#)
`__hook_include_legacy_code_`
`chunk:n` [h89](#), [h110](#), [h566](#)
`__hook_init_structure:n` [190](#), [192](#), [199](#),
[223](#), [h86](#), [h92](#), [h299](#), [h315](#), [h485](#), [h921](#)
`__hook_initialize_all:` [h545](#), [h1261](#)
`__hook_initialize_hook_code:n` [212](#), [h546](#), [h562](#), [h965](#)
`__hook_initialize_single:NNn` [207](#), [208](#), [210](#), [h584](#), [h598](#)
`\l_hook_label_0_tl` [h593](#)
`__hook_label_if_exist_apply:nnnTF`
. [213](#), [218](#), [h671](#)
`__hook_label_ordered:nn` [207](#)
`__hook_label_ordered:nnTF` [206](#), [h498](#), [h504](#), [h510](#), [h529](#)
`__hook_label_ordered_p:nn` [h529](#)
`__hook_label_pair:nn` [206](#), [207](#), [h497](#), [h503](#), [h509](#),
[h513](#), [h516](#), [h520](#), [h521](#), [h753](#), [h754](#)
`\l_hook_labels_int` [212](#), [h593](#), [h601](#), [h605](#), [h637](#), [h658](#)
`\l_hook_labels_seq` [h593](#), [h600](#), [h606](#), [h624](#), [h759](#)
`__hook_list_if_rule_exists:nnnTF`
. [h869](#)
`__hook_list_one_rule:nnn` [h869](#)
`__hook_list_rules:nn` [218](#), [h822](#), [h869](#), [h903](#)
`__hook_log:nN` [h769](#)
`__hook_log_cmd:n` [h771](#), [h776](#), [h780](#), [h782](#), [h786](#)
`__hook_log_line:n` [h769](#)
`__hook_log_line_indent:n` [h769](#)
`__hook_log_next_code:n` [h818](#), [h867](#)
`__hook_make_name:n` [189](#), [197](#),
[h205](#), [h211](#), [h220](#), [h226](#), [h1181](#), [h1223](#)
`__hook_make_name:w` [h226](#)
`__hook_make_prefixes:w` [i127](#)
`__hook_make_usable:n` [201](#), [h77](#), [h80](#), [h164](#), [h357](#)
`__hook_mark_removal:nn` [204](#), [h446](#), [h453](#), [h457](#)
`__hook_msg_pair_found:nnn` [h696](#),
[h703](#), [h710](#), [h718](#), [h726](#), [h737](#), [h750](#)
`\g_hook_name_stack_seq` [h34](#), [h1167](#), [h1168](#), [h1172](#),
[h1179](#), [h1193](#), [h1200](#), [h1208](#), [h1218](#)
`__hook_new:n` [h69](#), [h105](#)
`__hook_new_reversed:n` [h101](#)
`__hook_next_⟨hook⟩` [190](#), [203](#), [222](#)
`__hook_normalize_hook_args:Nn` [h70](#), [h102](#), [h125](#), [h149](#),
[h151](#), [h233](#), [h772](#), [h777](#), [h915](#), [h995](#)

File Key: a=ltldirchk.dtx, b=ltplain.dtx, c=ltvers.dtx, d=ltluatex.dtx, e=ltexpl.dtx,
f=ltdefs.dtx, g=ltcmd.dtx, h=lthooks.dtx, i=ltcmdhooks.dtx, j=ltalloc.dtx, k=ltcntrl.dtx,
l=lterror.dtx, m=ltpar.dtx, n=ltpara.dtx, o=ltspace.dtx, p=ltlogos.dtx, q=ltfiles.dtx,
r=ltoutenc.dtx, s=ltcounts.dtx, t=ltlength.dtx, u=ltfssbas.dtx, v=ltfssaxes.dtx,
w=ltfssstrc.dtx, x=ltfsscmp.dtx, y=ltfssdcl.dtx, z=ltfssini.dtx, A=fontdef.dtx,
B=preload.dtx, C=ltfntcmd.dtx, D=lttextcomp.dtx, E=ltpageno.dtx, F=ltxref.dtx,
G=ltmiscen.dtx, H=ltmath.dtx, I=ltlists.dtx, J=ltboxes.dtx, K=lttab.dtx, L=ltpicture.dtx,
M=ltthm.dtx, N=ltsect.dtx, O=ltfloat.dtx, P=ltidxglo.dtx, Q=ltbibl.dtx, R=ltpage.dtx,
S=ltclass.dtx, T=ltfilehook.dtx, U=ltshipout.dtx, V=ltoutput.dtx, W=lthyphen.dtx,
X=ltfinal.dtx

`__hook_normalize_hook_args:Nnn` .
 [h233](#), [h266](#), [h429](#)
`__hook_normalize_hook_args_-`
 aux:Nn [h233](#)
`__hook_normalize_hook_rule_-`
 args:Nnnnn [h233](#), [h480](#)
`__hook_parse_dot_label:n` [h184](#), [h186](#)
`__hook_parse_dot_label:w` [h186](#)
`__hook_parse_dot_label_aux:w` . [h186](#)
`__hook_parse_dot_label_cleanup:w`
 [h186](#)
`__hook_parse_label_default:n` . . .
 [h180](#),
 [h245](#), [h251](#), [h252](#), [h259](#), [h260](#), [h262](#)
`__hook_patch_check:NNnn` [i57](#)
`__hook_patch_cmd_or_delay:Nnn` . .
 [200](#), [236](#), [i23](#), [i26](#)
`__hook_patch_command:Nnn`
 [236](#), [237](#), [i36](#), [i54](#), [i57](#)
`__hook_patch_debug:n`
 [i12](#), [i59](#), [i60](#), [i63](#),
 [i66](#), [i69](#), [i129](#), [i183](#), [i219](#), [i222](#), [i223](#), [i228](#)
`__hook_patch_DeclareRobustCommand:Nnn`
 [238](#), [i106](#), [i110](#)
`__hook_patch_expand_redefine:NNnn`
 . [238](#), [241](#), [i114](#), [i119](#), [i124](#), [i127](#), [i185](#)
`__hook_patch_newcommand:Nnn` . . .
 [238](#), [i107](#), [i113](#), [i117](#)
`\l_hook_patch_num_args_int`
 [i7](#), [i130](#), [i135](#), [i141](#)
`\l_hook_patch_param_text_tl` . . .
 . . . [239](#), [240](#), [i8](#), [i137](#), [i143](#), [i148](#), [i162](#)
`\l_hook_patch_prefixes_tl`
 [240](#), [i8](#), [i156](#), [i161](#)
`\l_hook_patch_replacement_tl` . . .
 . . . [239](#), [240](#), [i8](#), [i139](#), [i144](#), [i149](#), [i167](#)
`__hook_patch_required_catcodes:`
 [245](#), [i233](#), [i254](#), [i280](#)
`__hook_patch_retokenize:Nnnn` . . .
 [242](#), [i224](#), [i259](#)
`__hook_preamble_hook:n`
 . . . [220](#), [221](#), [h560](#), [h785](#), [h942](#), [h986](#)
`__hook_provide:n` [h146](#)
`__hook_provide:nn` . [h149](#), [h151](#), [h152](#)
`\l_hook_rear_tl`
 . . . [h593](#), [h622](#), [h628](#), [h629](#), [h651](#), [h652](#)
`\g_hook_removal_list_prop` [h29](#)
`\g_hook_removal_list_tl`
 [204](#), [h29](#), [h459](#), [h464](#), [h469](#)
`__hook_removal_tl:nn`
 [h460](#), [h465](#), [h470](#), [h473](#)
`__hook_retokenize_patch:Nnn` [i72](#), [i181](#)
`\l_hook_return_tl`
 [h25](#), [h305](#), [h308](#), [h444](#), [h638](#), [h639](#),
 [h994](#), [h996](#), [h1200](#), [h1201](#), [h1208](#), [h1212](#)
`__hook_rule<_gset:nnn` [h495](#)
`__hook_rule>_gset:nnn` [h495](#)
`__hook_rule_after_gset:nnn` . . . [h495](#)
`__hook_rule_before_gset:nnn` . . .
 [211](#), [h495](#)
`__hook_rule_gclear:nnn`
 [207](#), [h486](#), [h518](#)
`__hook_rule_incompatible-error_-`
 gset:nnn [h512](#)
`__hook_rule_incompatible-warning_-`
 gset:nnn [h512](#)
`__hook_rule_unrelated_gset:nnn` .
 [207](#), [h518](#)
`__hook_rule_voids_gset:nnn` . . . [h507](#)
`__hook_seq_csname:n`
 . . . [h591](#), [h608](#), [h642](#), [h699](#), [h706](#), [h764](#)
`__hook_set_default_label:n`
 [h1223](#), [h1225](#)
`__hook_str_compare:nn`
 [h23](#), [h523](#), [h531](#), [h540](#)
`__hook_strip_double_slash:n` . [h413](#)
`__hook_strip_double_slash:w` . [h413](#)
`__hook_tl_csname:n`
 . . . [h591](#), [h597](#), [h607](#), [h623](#), [h626](#),
 [h628](#), [h632](#), [h644](#), [h646](#), [h649](#), [h651](#),
 [h656](#), [h697](#), [h698](#), [h704](#), [h705](#), [h763](#)
`__hook_tl_gclear:N`
 . . . [h66](#), [h118](#), [h437](#), [h438](#), [h442](#), [h633](#)
`__hook_tl_gput:Nn`
 . . . [212](#), [h579](#), [h581](#), [h639](#), [h665](#), [h669](#)
`__hook_tl_gput_left:Nn` . . . [h59](#), [h579](#)
`__hook_tl_gput_right:Nn`
 [h56](#), [h300](#), [h581](#), [h666](#), [h938](#)
`__hook_tl_gset:Nn`
 [h47](#), [h57](#), [h61](#), [h497](#),
 [h503](#), [h509](#), [h513](#), [h516](#), [h571](#), [h937](#)
`__hook_tl_gset_eq:NN` [h65](#), [h67](#)
`__hook_tl_set:Nn` [188](#), [h41](#), [h607](#)
`__hook_tmp:w` [244](#), [h36](#), [h871](#),
 [h892](#), [h901](#), [h912](#), [h1169](#), [h1173](#),
 [h1175](#), [i245](#), [i248](#), [i252](#), [i262](#), [i265](#), [i278](#)
`\l_hook_tmpa_bool`
 . . . [217](#), [h24](#), [h821](#), [h824](#), [h832](#), [h841](#)
`\l_hook_tmpa_tl` [h25](#), [h1179](#),
 [i187](#), [i202](#), [i205](#), [i251](#), [i254](#), [i277](#), [i280](#)
`\l_hook_tmpb_tl` . [h25](#), [i192](#), [i203](#), [i206](#)
`__hook_toplevel_⟨hook⟩`
 [189](#), [199](#), [203](#), [222](#)
`__hook_try_declaring_generic_-`
 hook:nnn [199](#), [h283](#), [h318](#)

File Key: a=ltldirchk.dtx, b=ltplain.dtx, c=ltvers.dtx, d=ltluatex.dtx, e=ltexpl.dtx,
 f=ltdefs.dtx, g=ltcmd.dtx, h=lthooks.dtx, i=ltcmdhooks.dtx, j=ltalloc.dtx, k=ltcntrl.dtx,
 l=lterror.dtx, m=ltpar.dtx, n=ltpara.dtx, o=ltspace.dtx, p=ltlogos.dtx, q=ltfiles.dtx,
 r=ltoutenc.dtx, s=ltcounts.dtx, t=ltlength.dtx, u=ltfssbas.dtx, v=ltfssaxes.dtx,
 w=ltfssstrc.dtx, x=ltfsscmp.dtx, y=ltfssdcl.dtx, z=ltfssini.dtx, A=fontdef.dtx,
 B=preload.dtx, C=ltfntcmd.dtx, D=lttextcomp.dtx, E=ltpageno.dtx, F=ltxref.dtx,
 G=ltmiscen.dtx, H=ltmath.dtx, I=ltlists.dtx, J=ltboxes.dtx, K=lttab.dtx, L=ltpicture.dtx,
 M=ltthm.dtx, N=ltsect.dtx, O=ltfloat.dtx, P=ltidxglo.dtx, Q=ltbibl.dtx, R=ltpage.dtx,
 S=ltclass.dtx, T=ltfilehook.dtx, U=ltshipout.dtx, V=ltoutput.dtx, W=lthyphen.dtx,
 X=ltfinal.dtx

		I
__hook_try_declaring_generic_-hook:nNnn	199, 200, h320, h325, h328	\I b393, S1172, S1304, S1394, S1414, X241, X541
__hook_try_declaring_generic_-hook:wnTF	h339, h343	\i ... r247, r402, r452, r453, r454, r455, r456, r457, r547, r587, r588, r680, r682, r684, r686, r778, r1132, r1287, r1289, r1291, r1293, r1344, r1347, r1350, r1353, r1423, X245, X545, X552
__hook_try_declaring_generic_-hook_split:nNnn	h328	\ialign b457, b459, A337, A459, A530, A533, A537, A540, H155, H157, H176, K191, L141
__hook_try_declaring_generic_-next_hook:nn	199, h318, h924	\IeC X343, X347, X454
__hook_try_file_hook:n	221, h967	\if 582
__hook_try_patch_with_catcodes:Nnnnw	241, i199, i216	if commands:
__hook_try_put_cmd_hook:n	h356, i15	\if_case:w h523, h540
__hook_try_put_cmd_hook:w	i15	\if_cs_exist:w h680, h891, h953
__hook_unmark_removal:nn	198, 204, h273, h462	\if_int_compare:w .. h531, h1038, n62
__hook_unpatchable_cases:n	i298, i300	\if_meaning:w T239
__hook_update_hook_code:n	198, 203, 208, h278, h451, h490, h544, h546, h550, h935	\if_mode... 275
__hook_use:wn	220, 221, h949, h962, h967	\if_mode_horizontal: n29
__hook_use_end:	h942	\IfBold 494
__hook_use_initialized:n	220, h559, h942	\IfBooleanF g2218
__hook_use_once_store:n	h995, h1000	\IfBooleanT g2218
__hook_use_undefined:w	h942	\IfBooleanTF 163, g2218
\g_hook_used_prop	h33, h547, h554, h587	\IfClassAtLeastTF S164
\l_hook_work_prop	211, h31, h583, h603, h610, h612, h621, h638, h662, h731, h742	\ifcsname 414, 421, f635, f652, v440, v443, v653, v656, w128, z39, z50, S1061, S1192, U424
hook_?? internal commands:		\ifdefined . b493, b557, e22, e71, e72, e73, e74, e117, e118, e119, z476, X305
__hook_??	205	\iff A500
hook_⟨hook⟩ internal commands:		\IfFileExists 299, 758
__hook_⟨hook⟩	189, 190, 194, 208	\IfFileExists 317, a178, e37, e68, e114, q405, q417, q519, q575, q600, S811, T149, T183, T193, X574
\hookleftarrow	A480	\iffontchar r866, r1051, r1157, r1159, r1161, r1208
\hookrightarrow	A478	\IfFontSeriesContextTF 494, z349, z385, z387
hook ?? internal commands:		\IfFormatAtLeastTF 588, 813, S164
__hook~??	h475	\IfHookEmptyTF 169, 173, 229, h1257, h1292, G210
\hphantom	H75	\IfHookExistsTF 229, h1258, h1291
\hrule	b425, b469, o350, o358, o384, o392, r289, r294, A340, A618, J163, J168, J216, J226, K359, K376, L590, L600, O395	\ifinner H264, H272, H292, H319, O57, O126, O315
\hrulefill	b469, f779, f800	\IfNoValueF g2237
\hspace	o437, o441, o446	\IfNoValueT g2237
\hss	844	\IfNoValueTF g2237
\Hwithstroke	339, r494, r1205	\ifnum 829
\hwithstroke	339, r510, r1206	\ifodd y973, L320, L344, L378, L400, O68, O137, V21, V138, V610, V668, V982, V985, V1018, V1021, V1132, V1135,
\hyphenation	r205	
\hyphenchar	100, f737, f740, f749, f752, f755, f760, G454	
\hyphenpenalty	b321, u661, u693	

File Key: a=ltldirchk.dtx, b=ltplain.dtx, c=ltvers.dtx, d=ltluatex.dtx, e=ltexpl.dtx, f=ltdefns.dtx, g=ltcmd.dtx, h=lthooks.dtx, i=ltcmdhooks.dtx, j=ltalloc.dtx, k=ltcntrl.dtx, l=lterror.dtx, m=ltpar.dtx, n=ltpara.dtx, o=ltspace.dtx, p=ltlogos.dtx, q=ltfiles.dtx, r=ltoutenc.dtx, s=ltcounts.dtx, t=ltlength.dtx, u=ltfssbas.dtx, v=ltfssaxes.dtx, w=ltfssstrc.dtx, x=ltfsscmp.dtx, y=ltfssdcl.dtx, z=ltfssini.dtx, A=fontdef.dtx, B=preload.dtx, C=ltfntcmd.dtx, D=lttextcomp.dtx, E=ltpageno.dtx, F=ltxref.dtx, G=ltmiscen.dtx, H=ltmath.dtx, I=ltlists.dtx, J=ltboxes.dtx, K=lttab.dtx, L=ltpicture.dtx, M=ltthm.dtx, N=ltsect.dtx, O=ltfloat.dtx, P=ltidxglo.dtx, Q=ltbibl.dtx, R=ltpage.dtx, S=ltclass.dtx, T=ltfilehook.dtx, U=ltshipout.dtx, V=ltoutput.dtx, W=lthyphen.dtx, X=ltfinal.dtx

- V1294, V1297, V1574, V1577,
V1695, V1698, V1818, V2039, V2047
\IfPackageAtLeastTF S164
\IfTargetDateBefore S1622
\IfValueF g2240
\IfValueT g2240
\IfValueTF g2240
\ifvbox ... V319, V376, V423, V502, V518
\ifx 99, 484–486, 498, 815
\ignorespaces 183, 327, o49, o143, o162,
o174, o185, o201, o214, o483, q71,
q138, q193, r72, u292, u302, u312,
G216, G223, G274, G289, G337,
G342, G348, H300, H327, J154,
J383, J401, K57, K62, K68, K83,
K92, K105, K109, K116, K123,
K125, K134, K154, K237, K301,
K303, K305, K332, L36, L46, L66,
L75, L109, L120, L131, L143, L148,
L154, M30, M32, N110, O17, O24,
O476, O495, Q7, Q9, U330, I55, I217
\ignorespacesafterend G7
\IJ r250, r436, r550, r1133
\ij r249, r434, r549, r1134
\Im A314
\imath A309
\immediate 184
\in A429, A461
in commands:
in_callback d792
\in_callback 44
\include 299
\include 183, 309, 803, 805,
806, 809, q217, q265, q267, q283, q285
include/after 806
include/after/... 806
include/before 806
include/before/... 806
include/end 806
include/end/... 806
\IncludeInRelease . 71, 103, a18, a23,
a290, a307, b49, b88, b103, b119,
b125, b134, b139, b148, b154, b168,
b182, b186, b220, b228, b233, b244,
b289, b336, b346, b480, b504, b538,
b545, b571, b611, b616, b626, c68,
c168, d3, d227, d250, d273, e2, e11,
e20, e58, e112, e132, e144, f5, f13,
f56, f62, f272, f314, f342, f370, f377,
f394, f410, f448, f460, f488, f499,
f511, f518, f614, f631, f661, f702,
f721, f732, f749, f760, f771, f791,
g2264, h122, h140, h146, h172,
h343, h371, h1145, h1152, h1265,
i310, n128, o5, o21, o54, o65, o81,
o86, o98, o104, o132, o152, o166,
o178, o191, o206, o235, o252, o271,
o307, o333, o366, o426, o432, o440,
o445, o459, o465, q10, q83, q141,
q219, q256, q277, q292, q339, q400,
q410, q436, q456, q474, q484, q500,
q525, q530, q538, q552, q563, q579,
q612, r77, r104, r148, r169, r324,
r332, r353, r369, s24, s30, s46, s90,
s127, s143, s151, s169, t5, t11, u24,
u53, u210, u232, u284, u295, u305,
u420, u431, u439, u471, u479, u493,
u514, u550, u637, u699, v2, v385,
v392, v404, v416, v424, v506, v522,
v600, v613, v623, v630, v638, v710,
v729, v735, v739, w113, w158,
w161, w541, w550, x2, x22, y49,
y78, y138, y169, y200, y607, y655,
y701, y710, y894, y905, z33, z163,
z188, z267, z322, z351, z384, z396,
z430, z443, z474, z483, z508, z525,
z541, z558, A63, A81, A100, A110,
A622, A634, C27, C34, D607, F38,
F51, F62, G10, G65, G107, G112,
G124, G151, G170, G226, G276,
G301, G309, G322, G328, G340,
G345, G355, G374, G391, G415,
G437, G464, G489, G502, G517,
G533, G556, G566, H79, H87,
H109, H118, H139, H144, H152,
H163, H203, H220, H259, H267,
H277, H304, H385, H394, H431,
H443, H455, H464, J4, J14, J50,
J70, J91, J102, J117, J125, J181,
J189, J235, J244, J281, J303, J371,
J388, J408, J414, J434, J442, K60,
K65, K137, K157, K222, K227,
L10, L16, L26, L39, L55, L69, L80,
L89, L100, L112, L146, L151, L160,
L172, L235, L250, L311, L369,
L455, L464, L473, L501, L531,
L558, L572, L585, L596, L608,
L625, L645, L652, L686, L746,
L804, L820, N5, N20, N161, N168,
N174, N187, N210, O35, O105,
O206, O232, O280, O294, O335,
O352, O406, O412, O420, O427,
O434, O440, O446, O463, O481,
O531, O541, Q41, Q57, R20, R40,
S18, S23, S38, S53, S65, S89, S97,
S103, S128, S138, S166, S176, S188,
S201, S231, S250, S270, S286, S300,

File Key: a=ltldirchk.dtx, b=ltplain.dtx, c=ltvers.dtx, d=ltluatex.dtx, e=ltexpl.dtx,
f=ltdefs.dtx, g=ltcmd.dtx, h=lthooks.dtx, i=ltcmdhooks.dtx, j=ltalloc.dtx, k=ltcntrl.dtx,
l=lterror.dtx, m=ltpar.dtx, n=ltpara.dtx, o=ltspace.dtx, p=ltlogos.dtx, q=ltfiles.dtx,
r=ltoutenc.dtx, s=ltcounts.dtx, t=ltlength.dtx, u=ltfssbas.dtx, v=ltfssaxes.dtx,
w=ltfssstrc.dtx, x=ltfsscmp.dtx, y=ltfssdcl.dtx, z=ltfssini.dtx, A=fontdef.dtx,
B=preload.dtx, C=ltfntcmd.dtx, D=lttextcomp.dtx, E=ltpageno.dtx, F=ltxref.dtx,
G=ltmiscen.dtx, H=ltmath.dtx, I=ltlists.dtx, J=ltboxes.dtx, K=lttab.dtx, L=ltpicture.dtx,
M=ltthm.dtx, N=ltsect.dtx, O=ltfloat.dtx, P=ltidxglo.dtx, Q=ltbibl.dtx, R=ltpage.dtx,
S=ltclass.dtx, T=ltfilehook.dtx, U=ltshipout.dtx, V=ltoutput.dtx, W=lthyphen.dtx,
X=ltfinal.dtx

- S318, S331, S347, S367, S388, S404,
S433, S445, S477, S495, S515, S533,
S552, S562, S572, S584, S625, S635,
S691, S721, S748, S780, S908, S988,
S995, S1049, S1180, S1311, T5,
T15, T27, T89, T144, T179, T190,
T203, T247, T255, T267, T290,
T310, T323, T330, T353, T373,
T398, T444, T456, T467, T494,
U3, U421, U461, U475, V24, V54,
V151, V179, V345, V366, V371,
V419, V591, V651, V794, V812,
V873, V894, V930, V954, V1066,
V1217, V1386, V1468, V1562,
V1684, V1903, V1930, V2160,
V2178, V2197, V2243, X8, X16,
X23, X38, X57, X66, X73, X92,
X129, X152, X185, X274, X279,
X300, X325, X334, X427, I125, I133
\includeonly 299
\includeonly 307,
805, 806, q217, q257, q259, q278, q279
\indent 269, 276, o462, K81, I161
\IndentBox 267, 269, n49, n132
\index 742
\index N182, P6, P18, R24, R32, V620, V679
\indexentry P15
\inf H25
\infty A316
\initcatcodetable d91
\input 299, 758
\input 170, 183,
228, 755, 803, 804, 808, 809, 816,
a68, a174, a177, a234, d18, e108,
e130, f22, q535, w16, x106, z604,
z614, z624, A10, A11, A12, A13,
A14, A23, A41, A42, A46, A47,
A136, A137, A138, A139, A654,
A655, A656, D1129, S601, X150,
X164, X189, X267, X311, X391, X579
\input@path 1, 6
input@path commands:
 \input@path: a239
\inputencodingname ... X368, X390, X472
\InputIfFileExists 299, 758
\InputIfFileExists ... 315, 317, 780,
781, 809, 811, 812, 822, q518, q541,
q556, q566, q576, q630, r1538,
u391, z596, z606, z616, D845,
D1212, S877, S939, T143, W8, X261
\inputlineno a327, I165
\insert .. 265, b254, b279, b281, b284,
b299, O465, O484, V517, V518, V1883
\int A348
int commands:
 \int_add:Nn g517
 \int_compare:nNnTF g385,
 g512, g793, g1342, g1645, g1647,
 h626, h648, h658, h1051, i135,
 i189, U55, U112, U142, U353, U359
 \int_decr:N
 g464, g478, g492, g1341, h637
 \int_eval:n
 g1659, h645, h698, h705, U355
 \int_gdecr:N T520
 \int_gincr:N .. g943, T510, U87, U103
 \int_gset:Nn T506
 \int_incr:N g289,
 g408, g765, g825, g832, g1351, h605
 \int_new:N g15, g27,
 g35, g1321, h594, i7, T505, U346, U348
 \int_set:Nn .. g1325, g1752, g1778,
 i130, i152, i239, i240, T210, T224
 \int_step_inline:nnn i141
 \int_use:N 829, g25,
 g796, g947, T514, U105, U117, U363
 \int_value:w U48, U137
 \int_zero:N
 .. g280, g374, g687, g690, g800, h601
 \c_max_int ... U214, U240, U265, U292
 \c_zero_int i135, U91
\interdisplaylinepenalty
 o13, H55, H194, H379
\interfootlinepenalty b383
\interfootnotelinepenalty
 b383, o18, O467, O486
\interlinepenalty o11, u663,
 G426, G429, G447, G450, N67,
 N118, N194, N217, O467, O486,
 V338, V1153, V1157, V1319, V1323
\intextsep V1136, V1140,
 V1155, V1158, V1165, V1298,
 V1304, V1321, V1324, V1333, V2303
\intop A347, A348
\iota A276
iow commands:
 \iow_char:N
 g396, g460, g545, g614, g626, g657,
 g1326, g2042, g2123, h728, h739,
 h1066, h1096, h1108, h1113, h1118,
 i289, n114, n121, n122, n123, n124
 \iow_log:n h771
 \iow_newline: T518
 \iow_now:Nn U362
 \iow_term:n h291,
 h553, h555, h564, h641, h660, h661,
 h663, h727, h738, h752, h758, h759,

File Key: a=ltldirchk.dtx, b=ltplain.dtx, c=ltvers.dtx, d=ltluatex.dtx, e=ltexpl.dtx,
f=ltdefs.dtx, g=ltcmd.dtx, h=lthooks.dtx, i=ltcmdhooks.dtx, j=ltalloc.dtx, k=ltcntrl.dtx,
l=lterror.dtx, m=ltpar.dtx, n=ltpara.dtx, o=ltspace.dtx, p=ltlogos.dtx, q=ltfiles.dtx,
r=ltoutenc.dtx, s=ltcounts.dtx, t=ltlength.dtx, u=ltfssbas.dtx, v=ltfssaxes.dtx,
w=ltfssstrc.dtx, x=ltfsscmp.dtx, y=ltfssdcl.dtx, z=ltfssini.dtx, A=fontdef.dtx,
B=preload.dtx, C=ltfntcmd.dtx, D=lttextcomp.dtx, E=ltpageno.dtx, F=ltxref.dtx,
G=ltmiscen.dtx, H=ltmath.dtx, I=ltlists.dtx, J=ltboxes.dtx, K=lttab.dtx, L=ltpicture.dtx,
M=ltthm.dtx, N=ltsect.dtx, O=ltfloat.dtx, P=ltidxglo.dtx, Q=ltbibl.dtx, R=ltpage.dtx,
S=ltclass.dtx, T=ltfilehook.dtx, U=ltshipout.dtx, V=ltoutput.dtx, W=lthyphen.dtx,
X=ltfinal.dtx

h760, h763, h767, h776, h900, h905,
i13, i20, i28, i30, i46, i50, T511, T544
\ishortstack [L132](#)
\itdefault v697, z30, [A94](#)
\item 264, 615, i230, G351, G401, G403,
G417, G439, H438, H450, H477,
K78, M36, M38, Q4, Q8, [I141](#), I219
\itemindent 614, 615, [I9](#), I42, I95, I187, I208
itemize (environment) [I242](#)
\itemize [I242](#)
\itemsep 615, [I1](#), I176
\iterate a81, a82, [b413](#)
\itshape 422, 496, r447, r807,
v695, v696, z28, z29, z407, z440,
z446, C21, D634, M36, M38, O399

J

\J X243, X543
\j r248, r403,
r548, r779, r1142, r1357, r1443, X552
\jmath A310
\jobname 787, 812
\Join z578
\joinrel A471, A478, A480, A482, A484,
A486, A488, A490, A492, A496, A498
\jot [H53](#), H191, H390, H400

K

\k ... 345, r485, r590, r595, r617, r622,
r698, r699, r757, r758, r812, r814,
r819, r821, r1243, r1311, r1312,
r1329, r1330, r1352, r1353, r1354,
r1407, r1408, r1441, r1442, D181, D204
\kanjiskip e74
\kappa A277
\ker H27
\kern 841
kernel internal commands:
 _kernel_chk_if_free_cs:N
 151, 164, g1715, g1716, g2261
 _kernel_cmd_if_xparse:NTF
 238, g1708, [g1810](#), i108
 _kernel_exp_not:w
 h42, h48, h50, h57, h62
 l_kernel_expl_bool i194
 _kernel_file_name_sanitiz:n . T98
 _kernel_msg_expandable_-
 error:nnnn T429
 _kernel_msg_new:nnn T435
\kerneltmpDoNotUse 243,
 i232, i244, i250, i255, i261, i268, i281
\kill K154, K162

L

\L r242, r424, r530, r771, r1135,
S1169, S1301, S1391, S1413, X571
\l r251, r426, r551, r780, r1136, X571
\label 740, [F32](#), N182, R24, R32, V619, V678
\labelenumi 626
\labelenumiv 626
\labelformat 573
\labelformat [F46](#), F52, F58, F63, F69
\labelitemi 626
\labelitemii 626
\labelitemiii 626
\labelitemiv 626
\labelsep 615, 626, M36, M38, [I9](#), I210, I216
\labelwidth 613, 615, [I9](#), I93, I209, I211, I214
\Lambda A300
\lambda A278
\land A368, A370
\langle A594
\language b35, b82, b84, b99,
q52, q121, G422, G562, V597, W10
\lastbox 271, u681, H180, H181,
N99, N132, V305, I130, I136, I185
\LastDeclaredEncoding . u137, [u140](#), X468
\lastnamedcs f653
\lastnodetype u674, u675, u676, u680
\lastpenalty u677, C112, C115
\lastskip b437, b438, b440,
b442, o44, o128, o140, o159, o220,
o221, o225, o227, o228, o240, o258,
o280, o283, o315, o318, o319, C102,
C105, L123, I115, I116, I150, I151
\LaTeX f732, p3,
p15, S1131, S1264, S1353, U388, U394
\LaTeXe p13
\latexrelease 822
\latexreleaseversion c1
\lbrace r308, A598
\lbrack b397
\lccode i19, i20,
i21, i22, i23, i24, r140, r1056, G507,
G522, G537, G580, X210, X227,
X235, X242, X244, X245, X247,
X249, X250, X251, X252, X257,
X535, X542, X544, X545, X547, X549
\lceil A602
\ldotp A501, A504, A619
\ldots r322, A505
\le A416, A418
\leaders b469, A340,
A558, A559, A561, A562, K376,
L565, L578, L590, L600, N199, N222
\leadsto z581

File Key: a=ltldirchk.dtx, b=ltplain.dtx, c=ltvers.dtx, d=ltluatex.dtx, e=ltexpl.dtx,
f=ltdefs.dtx, g=ltcmd.dtx, h=lt hooks.dtx, i=ltcmdhooks.dtx, j=ltalloc.dtx, k=ltcntrl.dtx,
l=lterror.dtx, m=ltpar.dtx, n=lt para.dtx, o=lt space.dtx, p=ltlogos.dtx, q=ltfiles.dtx,
r=ltoutenc.dtx, s=ltcounts.dtx, t=ltlength.dtx, u=ltfssbas.dtx, v=ltfssaxes.dtx,
w=ltfssstrc.dtx, x=ltfsscmp.dtx, y=ltfssdcl.dtx, z=ltfssini.dtx, A=fontdef.dtx,
B=preload.dtx, C=ltfntcmd.dtx, D=lttextcomp.dtx, E=ltpageno.dtx, F=ltxref.dtx,
G=ltmiscen.dtx, H=ltmath.dtx, I=ltlists.dtx, J=ltboxes.dtx, K=lttab.dtx, L=lt pictur.dtx,
M=ltthm.dtx, N=ltsect.dtx, O=ltfloat.dtx, P=ltidxglo.dtx, Q=ltbibl.dtx, R=ltpage.dtx,
S=ltclass.dtx, T=ltfilehook.dtx, U=ltshipout.dtx, V=ltoutput.dtx, W=lthyphen.dtx,
X=ltfinal.dtx

- `\leavevmode` [b428](#), [b455](#), [b458](#),
[b469](#), [b471](#), [o407](#), [o421](#), [r75](#), [r184](#),
[r289](#), [r290](#), [r393](#), [r425](#), [r429](#), [r432](#),
[r479](#), [r763](#), [r796](#), [C123](#), [D98](#), [D927](#),
[G426](#), [G447](#), [G460](#), [G471](#), [G479](#),
[G558](#), [G568](#), [G581](#), [H438](#), [H450](#),
[H477](#), [J8](#), [J17](#), [J24](#), [J156](#), [J158](#),
[J174](#), [J202](#), [J263](#), [J339](#), [J421](#), [J438](#),
[J445](#), [K178](#), [L134](#), [L314](#), [L373](#), [N40](#),
[N195](#), [N207](#), [N218](#), [O512](#), [Q14](#),
[V157](#), [V162](#), [V184](#), [V189](#), [I58](#), [I103](#)
- `\left` .. [A625](#), [A627](#), [A629](#), [A631](#), [A636](#),
[A637](#), [A638](#), [A639](#), [H154](#), [H160](#), [H182](#)
- `\Leftarrow` [A410](#), [A492](#), [A498](#)
- `\leftarrow`
[A437](#), [A439](#), [A480](#), [A490](#), [A496](#), [A550](#)
- `\leftarrowfill` [A534](#), [A550](#)
- `\lefteqn` [H414](#)
- `\leftharpoondown` [A453](#), [A467](#)
- `\leftharpoonup` [A452](#)
- `\lefthyphenmin` [W11](#)
- `\leftline` [J472](#)
- `\leftmargin`
[613](#), [615](#), [I9](#), [I52](#), [I53](#), [I94](#), [I146](#), [I148](#)
- `\leftmargini` [615](#), [H430](#), [I17](#)
- `\leftmarginii` [I17](#)
- `\leftmarginiii` [I17](#)
- `\leftmarginiv` [I17](#)
- `\leftmarginv` [I17](#)
- `\leftmarginvi` [615](#), [I17](#)
- `\leftmark` [R48](#)
- `\Leftrightarrow` [A409](#)
- `\leftrightharpoonarrow` [A436](#)
- `\leftskip` [b450](#),
[u658](#), [G359](#), [G365](#), [G369](#), [G379](#),
[G383](#), [G387](#), [G419](#), [G441](#), [J295](#),
[J316](#), [N192](#), [N197](#), [N215](#), [N220](#), [I74](#)
- `\legacyoldstylenums` [D4](#), [D617](#)
- `\leq` [A414](#), [A416](#)
- `\let` ... [86](#), [89](#), [229](#), [377](#), [519](#), [523](#), [822](#), [848](#)
- `\LetLtxMacro` [93](#)
- `\lfloor` [A606](#)
- `\lg` [H4](#)
- `\lgroup` [A608](#)
- `\lhd` [z584](#)
- `\lhook` [A477](#), [A478](#)
- `\lim` [H6](#)
- `\liminf` [H8](#)
- `\limits` [A539](#), [A543](#), [H149](#), [H340](#)
- `\limsup` [H7](#)
- `\line` [I219](#), [L158](#), [L450](#), [L809](#), [L826](#)
- `\linebreak` [279](#)
- `\linebreak` [o9](#), [o26](#)
- `\linepenalty` [b320](#)
- `\lineskip` ... [b391](#), [b423](#), [b458](#), [A458](#),
[H187](#), [J297](#), [J317](#), [K71](#), [K198](#), [L136](#),
[L315](#), [L374](#), [U223](#), [U274](#), [V622](#), [V681](#)
- `\lineskiplimit` [b392](#), [b423](#), [b460](#), [b461](#),
[A458](#), [A510](#), [H189](#), [H193](#), [J283](#),
[J298](#), [J305](#), [U224](#), [U275](#), [V622](#), [V681](#)
- `\linespread` [u316](#)
- `\linethickness` [L130](#), [L810](#), [L827](#)
- `\linewidth` [614](#), [q30](#), [q99](#), [q157](#),
[H285](#), [H311](#), [H439](#), [H451](#), [H478](#),
[H482](#), [H500](#), [J293](#), [J314](#), [K36](#),
[O266](#), [V146](#), [V205](#), [I15](#), [I51](#), [I52](#), [I54](#)
- `list` (environment) [I34](#)
- `\list` [I34](#), [I236](#), [I247](#)
- `\listfiles` [758](#)
- `\listfiles` [184](#), [185](#), [q645](#)
- `\listparindent` [615](#), [I9](#), [I41](#), [I50](#)
- `\ll` [A434](#)
- `\llap` [J476](#), [I238](#), [I249](#)
- `\lmoustache` [A563](#)
- `\ln` [H5](#)
- `\lnot` [A326](#), [A327](#)
- `\LoadClass` [756](#)
- `\LoadClass` [S607](#),
[S621](#), [S845](#), [S956](#), [S1030](#), [S1038](#), [S1039](#)
- `\LoadClassWithOptions` [756](#)
- `\LoadClassWithOptions` [S620](#)
- `\LoadFontDefinitionFile` [u418](#),
[u444](#), [u445](#), [A21](#), [A27](#), [A28](#), [A29](#), [A33](#)
- `\LoadPackageWithOptions` [804](#)
- `\loccount` [d17](#)
- `\log` [H3](#)
- `\loggingall` [b480](#)
- `\loggingoutput` [b476](#), [b494](#), [b512](#), [b528](#), [b542](#)
- `\LogHook` [173](#), [h1246](#)
- `\long` [85](#)
- `\Longleftarrow` [A492](#)
- `\longleftarrow` [A489](#)
- `\Longleftrightarrow` [A498](#), [A500](#)
- `\longleftrightharpoonarrow` [A496](#)
- `\longmapsto` [A494](#)
- `\Longrightarrow` [A486](#)
- `\longrightarrow` [A487](#), [A494](#)
- `\loop` [a81](#), [b413](#), [d150](#),
[d159](#), [u672](#), [K382](#), [S1079](#), [S1140](#),
[S1210](#), [S1273](#), [S1324](#), [S1362](#), [X351](#),
[X362](#), [X372](#), [X383](#), [X413](#), [X439](#), [X449](#)
- `\lor` [A369](#), [A371](#)
- `\lower` . [p2](#), [A458](#), [J214](#), [L35](#), [L45](#), [L196](#),
[L305](#), [L306](#), [L353](#), [L354](#), [L409](#), [L410](#)
- `\lowercase` [l26](#), [r141](#), [r1057](#), [r1536](#), [u332](#),
[u390](#), [G511](#), [G526](#), [G541](#), [G581](#), [X567](#)
- `\lq` [b395](#)
- `lrbox` (environment) [629](#)

File Key: a=ltldirchk.dtx, b=ltplain.dtx, c=ltvers.dtx, d=ltluatex.dtx, e=ltexpl.dtx,
f=ltdefns.dtx, g=ltcmd.dtx, h=lt hooks.dtx, i=ltcmdhooks.dtx, j=ltalloc.dtx, k=ltcntrl.dtx,
l=lterror.dtx, m=ltpar.dtx, n=lt para.dtx, o=lt space.dtx, p=ltlogos.dtx, q=ltfiles.dtx,
r=ltoutenc.dtx, s=ltcounts.dtx, t=ltlength.dtx, u=ltfssbas.dtx, v=ltfssaxes.dtx,
w=ltfssstrc.dtx, x=ltfsscmp.dtx, y=ltfssdcl.dtx, z=ltfssini.dtx, A=fontdef.dtx,
B=preload.dtx, C=ltfntcmd.dtx, D=lttextcomp.dtx, E=ltpageno.dtx, F=ltxref.dtx,
G=ltmiscen.dtx, H=ltmath.dtx, I=ltlists.dtx, J=ltboxes.dtx, K=lttab.dtx, L=lt pictur.dtx,
M=ltthm.dtx, N=ltsect.dtx, O=ltfloat.dtx, P=ltidxglo.dtx, Q=ltbibl.dtx, R=ltpage.dtx,
S=ltclass.dtx, T=ltfilehook.dtx, U=ltshipout.dtx, V=ltoutput.dtx, W=lt hyphen.dtx,
X=ltfinal.dtx

<code>\lrbox</code>	J144	A181 , A182 , A183 , A184 , A185 ,	
<code>\ltfilehookdate</code>	T503	A186 , A187 , A188 , A189 , A190 ,	
<code>\ltfilehookversion</code>	T503	A191 , A192 , A193 , A194 , A195 ,	
lua commands:		A196 , A197 , A198 , A199 , A200 ,	
<code>\lua_now:n</code>	U31	A201 , A202 , A203 , A204 , A205 ,	
<code>\luabytecode</code>	d194	A206 , A207 , A208 , A209 , A210 ,	
<code>\luachunk</code>	d202	A211 , A212 , A213 , A214 , A215 ,	
<code>\luaedef</code>	U42	A216 , A217 , A218 , A219 , A220 ,	
<code>\luafunction</code>	41 , d178	A221 , A222 , A223 , A224 , A225 ,	
<code>luatexbase</code>	d280	A226 , A227 , A228 , A229 , A230 ,	
<code>\luatexluafunction</code>	a18 , a23	A297 , A298 , A299 , A300 , A301 ,	
<code>\luatexversion</code> ...	a11 , d5 , e73 , e119 , r994	A302 , A303 , A304 , A305 , A306 ,	
		A307 , A516 , A517 , A518 , A519 ,	
		A520 , A521 , A522 , A523 , A525 , A528	
M			
<code>\M</code>	b393	<code>\mathbf</code>	z14 , z199 , z280 , z328 , A151
<code>\magstep</code>	b384	<code>\mathbin</code> y976 , A232 , A233 , A235 , A358 ,	
<code>\magstephalf</code>	b384	A359 , A360 , A361 , A364 , A365 ,	
<code>\makeatletter</code>		A366 , A367 , A370 , A371 , A372 ,	
f728 , i190 , q32 , q101 , q159 , u396 ,		A373 , A374 , A375 , A376 , A377 ,	
G26 , G79 , N151 , S601 , S809 , S917 , V2		A378 , A379 , A380 , A381 , A382 ,	
<code>\makeatother</code>	f728 , i190 , S601 , X634	A383 , A384 , A385 , A386 , A387 ,	
<code>\makebox</code>	629	A388 , A389 , A390 , A391 , A392 ,	
<code>\makebox</code>	H285 , H311 , J3	A393 , A394 , A395 , A396 , A397 , H37	
<code>\makeglossary</code>	742	<code>\mathcal</code>	A150
<code>\makeglossary</code>	q204 , P20	<code>\mathchar</code>	
<code>\makeindex</code>	742	b456 , y733 , y777 , A335 , A336 , A617	
<code>\makeindex</code>	q203 , P3	<code>\mathchardef</code> .	b21 , b22 , b23 , b24 , b107 ,
<code>\makelabel</code>		b110 , b111 , d219 , j3 , j4 , j5 , j6 , r70 , y768	
... 614 , I45 , I97 , I205 , I218 , I238 , I249		<code>\mathcharzero</code>	d219
<code>\MakeLowercase</code>	X561 , X573	<code>\mathchoice</code>	H61
<code>\MakeRobust</code>	476 , f271 , f773 ,	<code>\mathclose</code>	y979 , A231 ,
f774 , f775 , f776 , f777 , f778 , f779 ,		A240 , A242 , A245 , A250 , A256 ,	
f780 , f781 , f782 , f783 , f784 , f785 ,		A258 , A260 , A566 , A593 , A597 ,	
f786 , f787 , f788 , y705 , y900 , L806 ,		A601 , A605 , A611 , H43 , H46 , H49 , H52	
L807 , L808 , L809 , L810 , L811 ,		<code>\mathcode</code>	y765 , A252 , A253 , A254
L812 , L813 , L814 , L815 , L816 , L817		<code>\mathdollar</code>	r307 , A614
<code>\maketitle</code>	706	<code>\mathellipsis</code>	r321 , A619
<code>\MakeUppercase</code>	F48 , F60 , X551	<code>\mathgroup</code>	
<code>\mapsto</code>	A444	b79 , u14 , w303 , w309 , w315 , w316 ,	
<code>\mapstochar</code>	A443 , A444 , A494	w327 , A643 , D8 , D14 , D614 , D1139	
<code>\marginpar</code>	267 , O308	<code>\mathhexbox</code>	b456 , z505
<code>\marginparpush</code>	V85 , V1834	<code>\mathindent</code> H428 , H440 , H452 , H480 , H490	
<code>\marginparsep</code>	V84 , V1845 , V1847	<code>\mathinner</code>	A504 , A508 , A513 , A619
<code>\marginparwidth</code> .	O341 , O359 , V83 , V1847	<code>\mathit</code>	v696 , z29 , A153 , A156 , A617
<code>\mark</code>	R27 , R35 , R53	<code>\mathnormal</code>	A149
<code>\markboth</code>	R21 , R22 , R41 , R43	<code>\mathop</code>	y975 , A341 , A342 ,
<code>\markright</code>	R22 , R44	A343 , A344 , A345 , A346 , A347 ,	
<code>\marks</code>	d37 , X10 , X12	A349 , A350 , A351 , A352 , A353 ,	
<code>math</code> (environment)	H332	A354 , A356 , A357 , A537 , A540 , H3 ,	
<code>\math</code>	H332	H4 , H5 , H6 , H7 , H8 , H9 , H10 , H11 ,	
<code>\mathaccent</code>	y622 , y670 , y704 , y714	H12 , H13 , H14 , H15 , H16 , H17 ,	
<code>\mathalpha</code> ..	y792 , y971 , A169 , A170 ,	H18 , H19 , H20 , H21 , H22 , H23 ,	
A171 , A172 , A173 , A174 , A175 ,		H24 , H25 , H26 , H27 , H28 , H29 ,	
A176 , A177 , A178 , A179 , A180 ,		H30 , H31 , H32 , H33 , H34 , H149 , H340	

File Key: a=ltldirchk.dtx, b=ltplain.dtx, c=ltvers.dtx, d=ltluatex.dtx, e=ltexpl.dtx, f=ltdefns.dtx, g=ltcmd.dtx, h=lt hooks.dtx, i=ltcmdhooks.dtx, j=ltalloc.dtx, k=ltcntrl.dtx, l=lterror.dtx, m=ltpar.dtx, n=lt para.dtx, o=lt space.dtx, p=ltlogos.dtx, q=ltfiles.dtx, r=ltoutenc.dtx, s=ltcounts.dtx, t=ltlength.dtx, u=ltfssbas.dtx, v=ltfssaxes.dtx, w=ltfssstrc.dtx, x=ltfsscmp.dtx, y=ltfssdcl.dtx, z=ltfssini.dtx, A=fontdef.dtx, B=preload.dtx, C=ltfntcmd.dtx, D=lttextcomp.dtx, E=ltpageno.dtx, F=ltxref.dtx, G=ltmiscen.dtx, H=ltmath.dtx, I=ltlists.dtx, J=ltboxes.dtx, K=lttab.dtx, L=lt pictur.dtx, M=ltthm.dtx, N=ltsect.dtx, O=ltfloat.dtx, P=ltidnglo.dtx, Q=ltbibl.dtx, R=ltpage.dtx, S=ltclass.dtx, T=ltfilehook.dtx, U=ltshipout.dtx, V=ltoutput.dtx, W=lthyphen.dtx, X=ltfinal.dtx

<code>\mathopen</code>	y978, A241, A244, A249, A255, A257, A259, A564, A595, A599, A603, A607, A609, H41, H44, H47, H50
<code>\mathord</code>	y792, y974, A236, A243, A246, A251, A263, A264, A265, A267, A268, A269, A270, A271, A272, A273, A274, A275, A276, A277, A278, A279, A280, A281, A282, A283, A284, A285, A286, A287, A288, A289, A290, A291, A292, A293, A294, A295, A296, A308, A309, A310, A311, A312, A313, A314, A315, A316, A317, A318, A319, A320, A321, A322, A323, A324, A325, A327, A328, A329, A330, A331, A332, A333, A334, A524, A526, A527, A549, A550, A553, A554, A555, A556, A568, A570, A572, A575, A577, A591, A613, A614, A615, A616
<code>\mathpalette</code>	A457, A461, A464, <u>H60</u> , H69, H99, H129
<code>\mathparagraph</code>	r310, s134, s146, <u>A614</u>
<code>\mathpunct</code>	y980, A234, A238, A501, A502, A503
<code>\mathrel</code>	y977, A237, A239, A247, A248, A261, A262, A338, A398, A399, A400, A401, A402, A403, A404, A405, A406, A407, A408, A409, A410, A411, A414, A415, A418, A419, A420, A421, A422, A423, A424, A425, A426, A427, A428, A429, A430, A432, A433, A434, A435, A436, A437, A438, A441, A442, A443, A445, A446, A447, A448, A449, A450, A451, A452, A453, A454, A455, A457, A461, A464, A471, A473, A476, A477, A479, A482, A484, A579, A581, A583, A585, A587, A589, H42, H45, H48, H51, H149, H340
<code>\mathring</code>	A528
<code>\mathrm</code>	z5, z238, z302, z334, A148
<code>\mathsection</code>	r311, s133, s145, <u>A614</u>
<code>\mathsf</code>	z8, z243, z307, z337, A152, A155
<code>\mathsterling</code>	r319, <u>A614</u>
<code>\mathstrut</code>	<u>H84</u> , H93, H158, H159
<code>\mathsurround</code>	b444
<code>\mathsymbol</code>	y770
<code>\mathtt</code>	z11, z248, z312, z340, A154
<code>\mathunderscore</code>	<u>A614</u>
<code>\mathversion</code>	<u>u336</u> , z467, z469
<code>\matrix</code>	<u>H156</u> , H160, H167
<code>\max</code>	H22
<code>\maxdeadcycles</code>	V7
<code>\maxdepth</code>	b365, o287, q60, q128, q183, V92, V169, V170, V506, V514, V546, V715, V724, V764, V991, X138
<code>\maxdimen</code>	847, <u>b309</u> , b366, b367, b423, b461, b477, b492, b493, b511, b527, b542, u645, u655, u690, u705, w384, w437, A458, L475, L503, L533, L611, L628, S1434, S1475, S1484, U351, U353, U400, V291, V1853, V1873, V1878, V2165, V2205, V2206, V2208, X142
<code>\mbox</code>	629
<code>\mbox</code>	b456, p13, r293, r409, r568, r1157, z501, A506, J11, J20, <u>J24</u> , L52, O409, O416, O437, O444
<code>\mddefault</code>	482, 483, 485, 487, z18, z195, z221, z222, z223, z232, z275, z296, z332, <u>A92</u> , A105, A107, A121
<code>\mdseries</code>	186, 487, 494, z16, z17, z156, <u>z218</u> , z290, z291, z330, z331, z504, C20
<code>mdseries</code>	<u>z252</u>
<code>mdseries/defaults</code>	<u>z252</u>
<code>\meaning</code>	a219, a228, a323, f228, f290, f328, f356, f439, f699, y444, y457, y558, y623, y670, y734, y828, y924, y1028
<code>\medbreak</code>	<u>b438</u> , f780, f801
<code>\mediumseries</code>	487
<code>\medmuskip</code>	A645, H36, H38, H211, H214, H228
<code>\medskip</code>	b441, <u>o400</u>
<code>\medskipamount</code>	b440, <u>o401</u> , <u>o403</u>
<code>\medspace</code>	<u>H201</u>
<code>\MessageBreak</code>	e78, e81, e82, e83, e84, e85, e86, e99, e100, e101, e102, e103, f204, f279, f318, f346, <u>l3</u> , l6, l13, l33, l46, l60, l73, l171, l173, l179, l186, r161, r988, r1541, r1544, u34, u35, u536, u570, v456, w20, w21, w67, w88, w327, w478, w498, w530, w546, w561, w574, x31, x33, y399, y408, y546, z58, C144, D23, D79, D81, D100, D851, D853, D854, D855, D857, D859, D860, D861, D862, D863, D913, D915, D922, D929, D1144, G43, G83, S278, S292, S647, S658, S660, S662, S673, S797, S798, S800, S801, S802, S804, S806, S840, S841, S842, S843, S924, S925, S927, S928, S929, S931, S933, S951, S952, S953, S954, S1015, S1032, S1033, S1101, S1118, S1157, S1232, S1251, S1290, S1345,

File Key: a=ltldirchk.dtx, b=ltplain.dtx, c=ltvers.dtx, d=ltluatex.dtx, e=ltexpl.dtx, f=ltdefs.dtx, g=ltcmd.dtx, h=lthooks.dtx, i=ltcmdhooks.dtx, j=ltalloc.dtx, k=ltcntrl.dtx, l=lterror.dtx, m=ltpar.dtx, n=ltpara.dtx, o=ltspace.dtx, p=ltlogos.dtx, q=ltfiles.dtx, r=ltoutenc.dtx, s=ltcounts.dtx, t=ltlength.dtx, u=ltfssbas.dtx, v=ltfssaxes.dtx, w=ltfssstrc.dtx, x=ltfsscmp.dtx, y=ltfssdcl.dtx, z=ltfssini.dtx, A=fontdef.dtx, B=preload.dtx, C=ltfntcmd.dtx, D=lttextcomp.dtx, E=ltpageno.dtx, F=ltxref.dtx, G=ltmiscen.dtx, H=ltmath.dtx, I=ltlists.dtx, J=ltboxes.dtx, K=lttab.dtx, L=ltpictur.dtx, M=ltthm.dtx, N=ltsect.dtx, O=ltfloat.dtx, P=ltidxglo.dtx, Q=ltbibl.dtx, R=ltpage.dtx, S=ltclass.dtx, T=ltfilehook.dtx, U=ltshipout.dtx, V=ltoutput.dtx, W=lthyphen.dtx, X=ltfinal.dtx

- S1379, S1488, S1490, S1572, S1575,
S1588, S1590, U99, U174, U468,
U469, U470, U471, V578, V1960,
V1997, X286, X287, X288, X290
- \mho z577
- \mid A402
- \min H23
- minipage (environment) 630
- \minipage J326
- \mit z629
- \mkern A335,
A338, A340, A462, A471, A513,
A514, A515, A545, A546, A547,
A548, A549, A550, A551, A552,
H36, H37, H40, H73, H74, N200, N223
- mlist commands:
mlist_to_hlist d841
- mode commands:
\mode_if_horizontal:TF n56, n61
\mode_if_horizontal_p: n53
\mode_if_inner:TF n57
\mode_if_inner_p: n54
\mode_if_vertical:TF n75, n87
- \models A484
- module commands:
module_error d336
module_info d336
module_warning d336
- \module_error 44
- \module_info 44
- \module_warning 44
- modules d289
- \month . a185, c11, c17, S1135, S1268, S1357
- \moveright V625, V684
- \mp A389
- \mscount K379
- msg commands:
\msg_error:nn . h1185, h1202, n83, n93
\msg_error:nnn g274, g656,
g1698, g1702, g2166, g2173, h74,
h160, h282, h302, h919, h1210, T75
\msg_error:nnnn . g387, g395, g419,
g423, g459, g473, g486, g569, g576,
g585, g613, g625, g644, g678, g1229,
g1766, g1770, g2133, g2147, g2185,
g2199, i24, i81, i211, n18, n30, n69
\msg_error:nnnnn g600, g1373,
g1380, g1658, h1189, h1220, h1229
\msg_error:nnnnnn h492, h711
\msg_expandable_error:nn h190
\msg_expandable_error:nnn
g345, g2231, h218
\msg_expandable_error:nnnn
g1550, g1577
- \msg_info:nnnn . . g65, g76, g182, g186
- \msg_line_context: . g2096, g2101,
g2106, g2111, h1071, h1076, h1109
- \g_msg_module_name_prop h1046
- \g_msg_module_type_prop
g50, h1044, h1045
- \msg_new:nnn
g2041, g2093, g2098, g2103,
g2108, g2113, g2119, h1069, h1074,
h1105, h1111, h1116, h1121, h1126
- \msg_new:nnnn g1886, g1893,
g1900, g1907, g1914, g1921, g1928,
g1938, g1944, g1952, g1959, g1969,
g1975, g1983, g1991, g2001, g2008,
g2015, g2022, g2028, g2034, g2043,
g2049, g2055, g2061, g2068, g2074,
g2084, h1047, h1057, h1062, h1079,
h1094, h1130, i283, i292, n105, n116
- \msg_redirect_module:nnn g49
- \msg_warning:nnn g73, h155
- \msg_warning:nnnn g637
- \msg_warning:nnnnnn h719
- \mskip H36,
H38, H206, H225, H228, H230, H231
- \mu A279
- \mubyte X338
- \multicolumn 586, K233
- \multipt L78, L811, L828
- \multispan K233, K379
- \muskip ... b29, b55, b93, d34, A545, A546
- \muskipdef b55, b93, d220
- \muskipzero d220
- N
- \n d323, d325, d332,
d334, d461, d568, d594, d620, d666,
d688, d707, d715, d716, d736, d749,
d756, d757, d764, d776, X107, X112
- \nabla A319
- \NAME 74
- \narrower b449
- \natural A329
- \ncallback d671
- \ndefault d676, d680
- \ne 519, A413
- \narrow A405
- \NeedsTeXFormat . w12, D819, S652, S1634
- \neg A325, A326
- \negmedspace H201
- \negthickspace H201
- \negthinspace o468, H201
- \neq 519, A412
- new commands:
new_attribute d402

File Key: a=ltldirchk.dtx, b=ltplain.dtx, c=ltvers.dtx, d=ltluatex.dtx, e=ltexpl.dtx,
f=ltdefns.dtx, g=ltcmd.dtx, h=lthooks.dtx, i=ltcmdhooks.dtx, j=ltalloc.dtx, k=ltcntrl.dtx,
l=lterror.dtx, m=ltpar.dtx, n=ltpara.dtx, o=ltspc.dtx, p=ltlogos.dtx, q=ltfiles.dtx,
r=ltoutenc.dtx, s=ltcounts.dtx, t=ltlength.dtx, u=ltfssbas.dtx, v=ltfssaxes.dtx,
w=ltfssstrc.dtx, x=ltfsscmp.dtx, y=ltfssdcl.dtx, z=ltfssini.dtx, A=fontdef.dtx,
B=preload.dtx, C=ltfntcmd.dtx, D=lttextcomp.dtx, E=ltpageno.dtx, F=ltxref.dtx,
G=ltmiscen.dtx, H=ltmath.dtx, I=ltlists.dtx, J=ltboxes.dtx, K=lttab.dtx, L=ltpicture.dtx,
M=ltthm.dtx, N=ltsect.dtx, O=ltfloat.dtx, P=ltidxglo.dtx, Q=ltbibl.dtx, R=ltpage.dtx,
S=ltclass.dtx, T=ltfilehook.dtx, U=ltshipout.dtx, V=ltoutput.dtx, W=lthyphen.dtx,
X=ltfinal.dtx

- new_bytecode [d436](#)
- new_chunkname [d449](#)
- new_luafunction [d465](#)
- new_whatsit [d424](#)
- \new_attribute [42](#)
- \new_bytecode [42](#)
- \new_chunkname [42](#)
- \new_luafunction [42](#)
- \new_whatsit [42](#)
- \newattribute [41](#)
- \newattribute [d74](#), [d230](#)
- \newbox [b47](#),
[b314](#), [b446](#), [j13](#), [G482](#), [H66](#), [J115](#),
[K16](#), [K17](#), [K18](#), [K343](#), [L6](#), [L670](#),
[L675](#), [V86](#), [V120](#), [V121](#), [V122](#), [I27](#)
- \newcatcodetable [41](#)
- \newcatcodetable
..... [d84](#), [d93](#), [d94](#), [d120](#), [d121](#), [d234](#)
- \newcommand [74](#)
- \newcommand [86](#), [89](#), [92](#), [95](#),
[f77](#), [r4](#), [v530](#), [v535](#), [v540](#), [z36](#), [A51](#),
[A52](#), [A53](#), [A54](#), [A56](#), [A57](#), [A59](#),
[A60](#), [A92](#), [A93](#), [A94](#), [A95](#), [A96](#),
[A97](#), [A120](#), [A121](#), [A122](#), [G303](#),
[G304](#), [G305](#), [G306](#), [L682](#), [T534](#),
[T535](#), [T536](#), [T537](#), [T539](#), [V2291](#),
[V2294](#), [V2297](#), [V2298](#), [V2301](#), [V2302](#)
- \NewCommandCopy ... [89](#), [91](#), [92](#), [f461](#), [f463](#)
- \newcount [b47](#), [b341](#), [b383](#), [j7](#), [j8](#),
[o124](#), [q7](#), [s36](#), [w25](#), [y27](#), [y286](#), [H55](#),
[H344](#), [H345](#), [J367](#), [K11](#), [K12](#), [K13](#),
[K14](#), [K15](#), [K335](#), [K336](#), [K337](#), [L664](#),
[L665](#), [L666](#), [L667](#), [L676](#), [N36](#), [N140](#),
[N141](#), [O3](#), [O267](#), [O268](#), [O269](#),
[O270](#), [S1431](#), [V103](#), [V105](#), [V107](#),
[V109](#), [V111](#), [V119](#), [V1986](#), [V2289](#),
[V2292](#), [V2295](#), [V2299](#), [X3](#), [X4](#), [X5](#),
[X77](#), [I23](#), [I24](#), [I25](#), [I26](#), [I56](#), [I226](#), [I241](#)
- \newcounter [366](#)
- \newcounter [s10](#)
- \newdimen
..... [b47](#), [b309](#), [b311](#), [b312](#), [b382](#), [j10](#),
[j11](#), [j12](#), [o123](#), [w398](#), [w399](#), [H53](#),
[J171](#), [J172](#), [K3](#), [K5](#), [K6](#), [K7](#), [K8](#),
[K166](#), [K338](#), [K339](#), [K340](#), [K341](#),
[L3](#), [L4](#), [L5](#), [L7](#), [L431](#), [L432](#), [L433](#),
[L434](#), [L435](#), [L436](#), [L668](#), [L669](#),
[L671](#), [L672](#), [L673](#), [L674](#), [O451](#),
[V71](#), [V72](#), [V73](#), [V75](#), [V76](#), [V77](#),
[V78](#), [V79](#), [V80](#), [V81](#), [V82](#), [V83](#),
[V84](#), [V85](#), [V91](#), [V93](#), [V94](#), [V106](#),
[V108](#), [V110](#), [V112](#), [V113](#), [V114](#),
[V115](#), [V116](#), [V117](#), [V118](#), [V1987](#),
[V1988](#), [I9](#), [I10](#), [I11](#), [I12](#), [I13](#), [I14](#),
[I15](#), [I16](#), [I17](#), [I18](#), [I19](#), [I20](#), [I21](#), [I22](#)
- \NewDocumentCommand
..... [89](#), [103](#), [152](#), [234](#), [g2127](#), [h1142](#),
[h1143](#), [h1144](#), [h1147](#), [h1148](#), [h1149](#),
[h1150](#), [h1160](#), [h1162](#), [h1164](#), [h1216](#),
[h1234](#), [h1236](#), [h1250](#), [h1252](#), [h1255](#)
- \NewDocumentEnvironment .. [g1917](#), [g2163](#)
- \newenvironment [75](#)
- \newenvironment
..... [182](#), [f146](#), [S1133](#), [S1266](#), [S1355](#)
- \NewExpandableDocumentCommand ... [g2179](#)
- \newfam [b47](#), [d38](#), [u16](#)
- \newfont [z471](#)
- \newgroup [y47](#)
- \newhelp [b306](#)
- \NewHook [165](#), [166](#), [173](#),
[178](#), [182](#), [190](#), [191](#), [233](#), [234](#), [h1142](#),
[h1268](#), [q72](#), [q73](#), [q74](#), [w150](#), [z252](#),
[z253](#), [z254](#), [z255](#), [z256](#), [z257](#), [z258](#),
[z259](#), [z260](#), [G33](#), [G34](#), [G35](#), [G36](#), [G37](#)
- \NewHookPair [232](#)
- \newif [c70](#), [f168](#),
[f767](#), [j9](#), [q5](#), [q6](#), [u204](#), [v401](#), [v413](#),
[y15](#), [z381](#), [C82](#), [D871](#), [F3](#), [H75](#),
[H76](#), [H190](#), [H346](#), [J406](#), [K19](#), [K251](#),
[L157](#), [L427](#), [L428](#), [L429](#), [L430](#),
[L459](#), [L460](#), [N38](#), [N124](#), [S2](#), [V95](#),
[V96](#), [V97](#), [V98](#), [V99](#), [V100](#), [V101](#),
[V102](#), [I28](#), [I29](#), [I30](#), [I31](#), [I32](#), [I33](#), [I138](#)
- \newinsert
..... [b193](#), [b242](#), [J368](#), [O390](#), [V27](#), [V1852](#)
- \newlabel [F22](#), [F34](#)
- \newlanguage [b47](#), [X270](#)
- \newlength [374](#)
- \newlength [t3](#)
- \newline [o92](#), [o99](#), [o105](#)
- \newlinechar [a72](#), [f20](#)
- \newluabytecode [41](#)
- \newluabytecode [d189](#), [d244](#)
- \newluachunkname [42](#)
- \newluachunkname [d197](#), [d246](#)
- \newluafunction [41](#)
- \newluafunction
..... [d4](#), [d173](#), [d228](#), [d240](#), [r1010](#), [U28](#)
- \newmarks [X6](#)
- \newmathalphabet [x13](#), [x109](#)
- \NewMirroredHookPair .. [166](#), [h1142](#), [h1270](#)
- \NewModuleRelease [c140](#), [g9](#), [h4](#), [i4](#), [n4](#), [D2](#)
- \newmuskip [b47](#)
- \newpage [V133](#), [V139](#), [V150](#)
- \newread [b47](#), [b307](#)
- \NewReversedHook
..... [166](#), [171](#), [178](#), [233](#), [h1142](#), [h1269](#)

File Key: a=ltldirchk.dtx, b=ltplain.dtx, c=ltvers.dtx, d=ltluatex.dtx, e=ltexpl.dtx, f=ltdefs.dtx, g=ltcmd.dtx, h=lthooks.dtx, i=ltcmdhooks.dtx, j=ltalloc.dtx, k=ltcntrl.dtx, l=terror.dtx, m=ltpar.dtx, n=ltpara.dtx, o=ltspace.dtx, p=ltlogos.dtx, q=ltfiles.dtx, r=ltoutenc.dtx, s=ltcounts.dtx, t=ltlength.dtx, u=ltfssbas.dtx, v=ltfssaxes.dtx, w=ltfssstrc.dtx, x=ltfsscmp.dtx, y=ltfssdcl.dtx, z=ltfssini.dtx, A=fontdef.dtx, B=preload.dtx, C=ltfntcmd.dtx, D=lttextcomp.dtx, E=ltpageno.dtx, F=ltxref.dtx, G=ltmiscen.dtx, H=ltmath.dtx, I=ltlists.dtx, J=ltboxes.dtx, K=lttab.dtx, L=ltpictur.dtx, M=ltthm.dtx, N=ltsect.dtx, O=ltfloat.dtx, P=ltidxglo.dtx, Q=ltbibl.dtx, R=ltpage.dtx, S=ltclass.dtx, T=ltfilehook.dtx, U=ltshipout.dtx, V=ltoutput.dtx, W=lthyphen.dtx, X=ltfinal.dtx

<code>\newrobustcmd</code>	89	<code>\nolinebreak</code>	o9 , o27
<code>\newsavebox</code>	629	<code>\nonfrenchspacing</code>	b387 , b632 , f782 , f803 , q48 , q117 , q175
<code>\newsavebox</code>	J115	<code>\nonscript</code>	H36, H38
<code>\newskip</code>	b47 , b310 , b313 , b380 , b381 , j14 , j15 , j17 , o403 , o404 , o405 , o454 , t3 , G400 , H347 , H429 , V2303 , V2304 , V2305 , V2309 , V2310 , V2313 , V2314 , V2315 , V2319 , V2320 , V2321 , I2 , I3 , I4 , I5 , I6 , I7 , I8	<code>\nonumber</code>	H373 , H412 , H413
<code>\newtheorem</code>	M1	<code>\nopagebreak</code>	279
<code>\newtie</code>	545, r851 , D188 , D189 , D208 , D705 , D1015 , D1016	<code>\nopagebreak</code>	o7 , o25
<code>\newtoks</code>	b47 , b306 , j16 , n35 , u346 , u347 , v499 , w247	<code>\noprotusion</code>	N207 , N230
<code>\newwhatsit</code>	41	<code>\normalbaselines</code>	b391 , H154 , H156
<code>\newwhatsit</code>	d181 , d242	<code>\normalbaselineskip</code>	b380 , b392 , w188 , J299 , J318
<code>\newwrite</code> b47 , b308 , q3 , q4 , N154 , P4 , P21		<code>\normalcolor</code>	920, H339 , H425 , J89 , J359 , N203 , N226 , O97 , O166 , V216 , V492 , V629 , V639 , V688 , V698 , V2226 , V2259
<code>\newXeTeXintercharclass</code>	X21	<code>\normalfont</code>	185, 327 , 487 , u646 , u706 , z506 , z526 , z528 , z537 , z542 , z544 , z552 , z559 , z561 , z567 , C18 , G461 , H339 , H425 , N203 , N226 , O401
<code>\NG</code>	r531 , r1137 , X571	<code>normalfont</code>	z252
<code>\ng</code>	r552 , r1138 , X571	<code>\normallineskip</code>	b380 , b391 , J297 , J317
<code>\ni</code>	A430 , A431	<code>\normallineskiplimit</code>	b380 , b392 , H193 , J282 , J298 , J304
<code>\noalign</code> 586 , A339 , A531 , A534 , A537 , A538 , A542 , A543 , H158 , H159 , H175 , H178 , H192 , H390 , H400 , K224 , K230 , K359 , K378 , L148 , L154		<code>\normalmarginpar</code>	O387
<code>\nobreak</code>	b426 , b429 , b431 , f781 , f802 , o59 , o71 , o115 , o141 , o147 , o160 , o173 , o199 , o351 , o359 , o385 , o393 , o414 , o421 , o452 , q202 , q214 , r409 , r435 , r437 , r568 , r1157 , G325 , G332 , J471 , N90 , N197 , N198 , N202 , N220 , N221 , N225 , O513 , R29 , R37 , V336 , V1149 , V1315 , X194 , X196 , X200 , X201 , X202 , X206	<code>\normalsfcodes</code>	q44 , q46 , q48 , q113 , q115 , q117 , q171 , q173 , q175 , q197 , V618 , V677
<code>\nobreakdashes</code>	o406	<code>\normalshape</code>	417 , 422 , v682 , v723
<code>\nobreakspace</code>	o420	<code>\normalsize</code> 168 , q42 , q111 , q169 , C142 , O23 , O176 , O372 , S5 , V617 , V676	
<code>\nobreakspace</code>	295	<code>\not</code>	A338 , A412 , A413 , A435
<code>\nocide</code>	748	<code>\notin</code>	A461
<code>\nocite</code>	745	<code>\nu</code>	A280
<code>\nocite</code>	Q39	<code>\null</code>	b405 , r327 , r363 , r486 , r489 , r834 , r837 , r1245 , F17 , G426 , G447 , G558 , G568 , H112 , H121 , H156 , H185 , N197 , N220 , U376
<code>\nocorr</code>	C43 , C58 , C62 , C65	<code>\nulldelimiterspace</code>	b369 , A642
<code>\nocorrlist</code>	C89 , C121	<code>\nullfont</code>	G167
<code>\nofiles</code>	299	<code>\number</code>	a86 , c49 , c58 , d105 , f2 , f114 , s108 , u596 , u599 , w439 , y64 , y93 , y113 , y128 , y153 , y184 , y216 , z498 , S1046 , S1135 , S1268 , S1357 , U351
<code>\nofiles</code>	580, 582 , 846 , q198	<code>\numberline</code>	N72 , N82 , N233 , O17
<code>\noindent</code>	264, 267 , 269 , 272 , 276 , 623 , u668 , u694 , N139	<code>\numexpr</code>	b189 , b205 , b215 , b246 , d82 , d105 , d157 , f642 , r1035 , V36
<code>\nointerlineskip</code>	b421 , A339 , A531 , A534 , A538 , A542 , H284 , H310 , L563 , L566 , L576 , L578 , V1842 , V1850	<code>\nunknown</code>	d693
<code>\nolimits</code>	A348 , A355 , H3 , H4 , H5 , H9 , H10 , H11 , H12 , H13 , H14 , H15 , H16 , H17 , H18 , H19 , H20 , H21 , H26 , H27 , H28 , H29 , H31 , H34	<code>\narrow</code>	A407
<code>\nolinebreak</code>	279		

O

File Key: a=ltldirchk.dtx, b=ltplain.dtx, c=ltvers.dtx, d=ltluatex.dtx, e=ltexpl.dtx, f=ltdefns.dtx, g=ltcmd.dtx, h=lt hooks.dtx, i=ltcmdhooks.dtx, j=ltalloc.dtx, k=ltcntrl.dtx, l=lterror.dtx, m=ltpar.dtx, n=lt para.dtx, o=lt space.dtx, p=ltlogos.dtx, q=ltfiles.dtx, r=ltoutenc.dtx, s=ltcounts.dtx, t=ltlength.dtx, u=ltfssbas.dtx, v=ltfssaxes.dtx, w=ltfssstrc.dtx, x=ltfsscmp.dtx, y=ltfssdcl.dtx, z=ltfssini.dtx, A=fontdef.dtx, B=preload.dtx, C=ltfntcmd.dtx, D=lttextcomp.dtx, E=ltpageno.dtx, F=ltxref.dtx, G=ltmiscen.dtx, H=ltmath.dtx, I=ltlists.dtx, J=ltboxes.dtx, K=lttab.dtx, L=lt pictur.dtx, M=ltthm.dtx, N=ltsect.dtx, O=ltfloat.dtx, P=ltidnglo.dtx, Q=ltbibl.dtx, R=lt page.dtx, S=ltclass.dtx, T=ltfilehook.dtx, U=ltshipout.dtx, V=ltoutput.dtx, W=lthyphen.dtx, X=ltfinal.dtx

- \obeylines [b408](#), [f783](#),
[f804](#), [G432](#), [G453](#), [G549](#), [G550](#), [V583](#)
- \obeyspaces [b408](#), [f784](#), [f805](#), [V583](#)
- \oddsidemargin ... [V72](#), [V74](#), [V611](#), [V670](#)
- \odot [A384](#)
- \OE ... [r243](#), [r399](#), [r532](#), [r769](#), [r1139](#), [X570](#)
- \oe ... [r252](#), [r404](#), [r553](#), [r782](#), [r1140](#), [X570](#)
- \of [H67](#), [H343](#)
- \offinterlineskip [b421](#)
- \oint [A355](#)
- \ointop [A354](#), [A355](#)
- \oldstylenums [330](#), [539](#), [550](#), [D4](#), [D376](#),
[D377](#), [D378](#), [D379](#), [D380](#), [D381](#),
[D382](#), [D383](#), [D384](#), [D385](#), [D610](#), [D1136](#)
- \Omega [A307](#)
- \omega [A290](#)
- \ominus [A387](#)
- \omit [H178](#), [H179](#), [K369](#), [K372](#), [K379](#), [K383](#)
- \OmitIndent [267](#), [269](#), [274](#), [n49](#), [n131](#)
- \onecolumn [V141](#)
- \OnlyDescription [w5](#), [B3](#)
- \oalign
[b458](#), [r327](#), [r357](#), [r394](#), [r480](#), [r486](#),
[r488](#), [r499](#), [r515](#), [r731](#), [r764](#), [r834](#),
[r837](#), [r894](#), [r1245](#), [z503](#), [A462](#), [A465](#)
- \openin [803](#)
- \openup [H186](#), [H191](#)
- \oplus [A388](#)
- \OptionNotUsed [S432](#), [S457](#), [S970](#)
- or commands:
- \or: [h525](#), [h541](#)
- \oslash [A385](#)
- \OT [r372](#)
- \otimes [A386](#)
- \outer [d21](#), [d38](#)
- \outerparskip [II](#)
- \output [V256](#)
- \outputpenalty
 . [V258](#), [V272](#), [V295](#), [V298](#), [V299](#),
 [V334](#), [V1159](#), [V1160](#), [V1325](#), [V1328](#)
- \oval [L450](#), [L453](#), [L812](#), [L829](#)
- \over [A469](#), [H149](#), [H341](#)
- \overbrace [A536](#)
- \overfullrule [b364](#), [R69](#)
- \overleftarrow [A533](#)
- \overrightarrow [A530](#)
- \owns [A431](#), [A432](#)
- P**
- \P [r310](#)
- package/after [805](#)
- package/after/... [805](#)
- package/before [805](#)
- package/before/... [805](#)
- \PackageError [c74](#), [c123](#),
[c135](#), [l84](#), [r1539](#), [D825](#), [D877](#), [D921](#)
- \PackageInfo [l84](#), [D829](#),
[D846](#), [D851](#), [D867](#), [D868](#), [D928](#), [D1213](#)
- \PackageWarning
 [l84](#), [D827](#), [D878](#), [D1142](#), [U467](#)
- \PackageWarningNoLine .. [l84](#), [r986](#), [V1959](#)
- \pagebreak [279](#)
- \pagebreak [o6](#), [o7](#), [o22](#), [o24](#)
- \pagegoal [V1880](#), [V1887](#)
- \pagenumbering [571](#)
- \pagenumbering [E5](#)
- \pageref [F10](#)
- \pageshrink [V538](#), [V542](#), [V558](#)
- \pagestyle [R2](#)
- \pagetotal [V128](#)
- \paperheight [V93](#)
- \paperwidth [V93](#)
- \par [265](#), [266](#), [268](#), [269](#),
[274–277](#), [581](#), [623](#), [a120](#), [b11](#), [b401](#),
[b409](#), [b410](#), [b425](#), [b434](#), [b435](#), [b436](#),
[b438](#), [b440](#), [b442](#), [d156](#), [f9](#), [f21](#), [m3](#),
[m4](#), [m5](#), [n101](#), [n137](#), [u671](#), [G109](#),
[G165](#), [G325](#), [G332](#), [G424](#), [G445](#),
[J288](#), [J309](#), [J355](#), [J384](#), [K195](#),
[K385](#), [N41](#), [N90](#), [N205](#), [N227](#), [O15](#),
[O24](#), [O249](#), [O344](#), [O477](#), [R62](#), [R63](#),
[U392](#), [V166](#), [V193](#), [V257](#), [V1886](#),
[I63](#), [I110](#), [I127](#), [I129](#), [I135](#), [I161](#), [I164](#)
- para commands:
- \para_end:
 . [268](#), [275](#), [276](#), [n51](#), [n101](#), [n102](#), [n103](#)
- \g_para_indent_box
 [269](#), [272](#), [n14](#), [n42](#), [n44](#), [n47](#), [n49](#), [n78](#)
- \para_omit_indent: [269](#), [274](#), [n46](#), [n50](#)
- \para_raw_end: ... [269](#), [276](#), [n74](#), [n99](#)
- \para_raw_indent: . [269](#), [276](#), [n74](#), [n97](#)
- \para_raw_noindent: [269](#), [276](#), [n74](#), [n98](#)
- para internal commands:
- _para_handle_indent: . [n31](#), [n43](#), [n80](#)
- \g_para_standard_everypar_tl ...
 ... [273](#), [274](#), [n12](#), [n34](#), [n36](#), [n79](#), [n90](#)
- para/after [267](#), [n6](#)
- para/before [267](#), [n6](#)
- para/begin [267](#), [n6](#)
- para/end [267](#), [n6](#)
- \paracntvalue [270](#)
- \paragraphmark [N143](#)
- \parallel [A401](#)
- \parbox [629](#)
- \parbox [265](#), [267](#), [J234](#)
- \parboxrestore [J322](#)
- \parfillskip
 [275](#), [276](#), [b379](#), [u645](#), [u660](#), [u705](#),

File Key: a=ltldirchk.dtx, b=ltplain.dtx, c=ltvers.dtx, d=ltluatex.dtx, e=ltexpl.dtx, f=ltdefns.dtx, g=ltcmd.dtx, h=lthooks.dtx, i=ltcmdhooks.dtx, j=ltalloc.dtx, k=ltcntrl.dtx, l=lterror.dtx, m=ltpar.dtx, n=ltpara.dtx, o=ltspace.dtx, p=ltlogos.dtx, q=ltfiles.dtx, r=ltoutenc.dtx, s=ltcounts.dtx, t=ltlength.dtx, u=ltfssbas.dtx, v=ltfssaxes.dtx, w=ltfssstrc.dtx, x=ltfsscmp.dtx, y=ltfssdcl.dtx, z=ltfssini.dtx, A=fontdef.dtx, B=preload.dtx, C=ltfntcmd.dtx, D=lttextcomp.dtx, E=ltpageno.dtx, F=ltxref.dtx, G=ltmiscen.dtx, H=ltmath.dtx, I=ltlists.dtx, J=ltboxes.dtx, K=lttab.dtx, L=ltpictur.dtx, M=ltthm.dtx, N=ltsect.dtx, O=ltfloat.dtx, P=ltidxglo.dtx, Q=ltbibl.dtx, R=ltpage.dtx, S=ltclass.dtx, T=ltfilehook.dtx, U=ltshipout.dtx, V=ltoutput.dtx, W=lthyphen.dtx, X=ltfinal.dtx

- G361, G371, G380, G388, G420,
G442, J296, J317, N192, N215, I76
- \parindent 264, b371, b450, b451, G361, G366,
G371, G380, G384, G388, G420,
G442, J291, J312, N193, N216, I50
- \parsep 615, I1, I49, I90
- \parseunicodedataI d123, d162
- \parseunicodedataII d124, d126
- \parseunicodedataIII d128, d134
- \parseunicodedataIV d130, d142
- \parseunicodedataV d146, d149
- \parshape 272, I54
- \parskip 264,
267, 271, 273, 615, b372, G336,
G418, G420, G440, G442, H496,
J291, J312, K79, V1159, V1327,
I49, I73, I88, I90, I117, I153, I172, I223
- \partial A315
- \partopsep 615, H494, I1, I61
- \PassOptionsToClass 756
- \PassOptionsToClass S366
- \PassOptionsToPackage 756
- \PassOptionsToPackage S366
- \patterns r205
- \pdffilesize e71, e117
- \pdfgentounicode X301,
X302, X306, X321, X326, X327, X328
- \pdfhorigin U314
- \pdftexrevision X307
- \pdftexversion X305, X306, X307
- \pdfvariable U313, U318
- \pdfvorigin U319
- peek commands:
- \peek_meaning:NTF 154, g1845
- \peek_meaning_remove:NTF
..... 140, 154, g1269, g1371, g1833
- \peek_N_type:TF g1275, g1299, g1331
- \peek_remove_spaces:n g1267
- \penalty b430, b431,
b432, b433, b434, b435, b439, b441,
b443, o34, o37, o46, o281, o291,
o316, o320, C118, G426, G429,
G447, G450, H37, H194, H390,
H400, K56, O195, O199, O201,
O217, O221, O223, Q17, V136,
V176, V195, V198, V1157, V1323, I190
- \perp A447
- \phantom H75
- \Phi A305
- \phi A287
- \Pi A302
- \pi A282
- picture (environment) L21
- \picture L21
- \pm A390
- \pmatrix H160, H168
- \pmod H39
- \PopDefaultHookLabel 170, h1166
- \poptabs I206, K142, K161
- \poptracing w148, w340
- \postdisplaypenalty o12, H437, H449, H475
- \pounds r318
- \Pr H32
- pre commands:
- pre_shipout_filter 829
- \prec A421
- \preceq A424
- \predisplaypenalty b329, H436, H448, H474
- \pretolerance b316, u647, u662, u707
- \prevdepth 291,
b421, b425, b426, o287, o288, o349,
o354, o383, o388, H192, O196,
O198, O218, O220, V167, V169, V172
- \PreviousTotalPages 829, 847, U398
- prg commands:
- \prg_break:n
.. g2221, g2223, g2225, g2227, g2228
- \prg_break_point: g2229
- \prg_do_nothing: 127, g336,
g827, g1212, i201, i313, T132, T134,
U165, U166, U171, U181, U185, U321
- \prg_new_conditional:Npnn
..... h131, h396, h529,
h1002, h1018, h1024, h1030, h1036
- \prg_new_protected_conditional:Npnn
..... h345, h374, h467, i87, i242
- \prg_replicate:nn
g798, g838, g893, g1665, g1672, T513
- \prg_return_false: h137,
h349, h367, h378, h391, h404,
h408, h411, h471, h534, h1014,
h1022, h1028, h1034, h1041, i102, i257
- \prg_return_true: h136, h365, h389,
h407, h471, h532, h1013, h1016,
h1021, h1027, h1033, h1039, i101, i256
- \prime A253, A317, H243
- \ProcessedArgument
..... 150, g317, g321, g328,
g1598, g1599, g1616, g1618, g1640,
g1649, g1655, g1663, g1680, g2243
- \ProcessList g2248
- \ProcessOptions r1560,
w71, D844, D881, S458, S549, S1034
- \ProcessOptions* S458
- \prod A349
- prop commands:
- \prop_clear:N g1120

File Key: a=ltldirchk.dtx, b=ltplain.dtx, c=ltvers.dtx, d=ltluatex.dtx, e=ltexpl.dtx,
f=ltdefns.dtx, g=ltcmd.dtx, h=lt hooks.dtx, i=ltcmdhooks.dtx, j=ltalloc.dtx, k=ltcntrl.dtx,
l=lterror.dtx, m=ltpar.dtx, n=lt para.dtx, o=lt space.dtx, p=ltlogos.dtx, q=ltfiles.dtx,
r=ltoutenc.dtx, s=ltcounts.dtx, t=ltlength.dtx, u=ltfssbas.dtx, v=ltfssaxes.dtx,
w=ltfssstrc.dtx, x=ltfsscmp.dtx, y=ltfssdcl.dtx, z=ltfssini.dtx, A=fontdef.dtx,
B=preload.dtx, C=ltfntcmd.dtx, D=lttextcomp.dtx, E=ltpageno.dtx, F=ltxref.dtx,
G=ltmiscen.dtx, H=ltmath.dtx, I=ltlists.dtx, J=ltboxes.dtx, K=lttab.dtx, L=lt pictur.dtx,
M=ltthm.dtx, N=ltsect.dtx, O=ltfloat.dtx, P=ltidxglo.dtx, Q=ltbibl.dtx, R=lt page.dtx,
S=ltclass.dtx, T=ltfilehook.dtx, U=ltshipout.dtx, V=ltoutput.dtx, W=lthyphen.dtx,
X=ltfinal.dtx

- `\prop_const_from_keyval:Nn`
 h423, h425, h426, h427
`\prop_gclear:N` h436, h547, i38
`\prop_get:NnN` h638
`\prop_get:NnNTF` ... g1126, h305, h444
`\prop_gput:Nnn` .. g50, h307, h310,
 h586, h1001, h1044, h1045, h1046, i31
`\prop_gremove:Nn` h456
`\prop_if_empty:NTF` h569, h795
`\prop_if_empty_p:N` h1007
`\prop_if_exist:NTF` h1026
`\prop_if_in:NnTF` h269, h351,
 h359, h362, h380, h383, h386, h406
`\prop_map_break:` h615, h877
`\prop_map_function:NN` i37
`\prop_map_inline:Nn` .. h554, h603,
 h610, h612, h761, h798, h872, h874
`\prop_new:N` g45,
 h31, h32, h33, h96, h475, h476, i11
`\prop_put:Nnn` g1117, h731, h742
`\prop_set_eq:NN` h583
`\prop_show:N` 218
`\propto` A398
`\protect` 587, 835, 837,
 838, f102, f220, f234, f243, f248,
 f251, f252, f254, f255, f260, f261,
 f266, f269, f270, f295, f333, f361,
 f533, f553, i197, i199, i200, i206,
 i212, i219, i227, i230, i236, q210,
 r26, r32, r51, r55, r209, r217, y507,
 y1055, z491, C143, F12, G193,
 G203, G238, G241, G256, G266,
 K264, N12, N72, N82, N164, N171,
 N177, O17, Q5, T272, U83, U95,
 U125, U132, U152, V596, V655, X344
`\protected` 89, 335, 422,
 476, f7, f443, o56, o462, r308, r309,
 s160, v682, v686, v689, v692, v695,
 v698, v701, v704, y897, z426, G126,
 G324, H374, H412, K56, K201,
 K208, L142, U42, U487, U488, U489
`\providecommand` f178, h1277,
 h1280, r6, r981, D811, D812, V1970
`\ProvideCommandCopy` 91
`\ProvideDocumentCommand` g2127
`\ProvideDocumentEnvironment` g2163
`\ProvideExpandableDocumentCommand` g2179
`\ProvideHook` 166, h1147, h1155
`\ProvideMirroredHookPair`
 166, h1147, h1157
`\ProvideReversedHook` .. 166, h1147, h1156
provides commands:
 `provides_module` d290
`\provides_module` 44
`\ProvidesClass` 756
`\ProvidesClass` S346
`\ProvidesExplPackage` T502
`\ProvidesFile`
 a89, A665, A667, A668, A669, S355
`\ProvidesPackage` 756
`\ProvidesPackage`
 766, 767, w13, D817, D849,
 S269, S348, S350, S1635, T531, U482
`\ProvideTextCommand` r3, r60
`\ProvideTextCommandDefault` r57
`\Psi` A306
`\psi` A289
`\PushDefaultHookLabel` .. 169, 170, h1166
`\pushtabs` .. i206, K138, K139, K158, K160
`\pushtracing` w117, w321
`\put` 826, 830, 832, 844,
 L56, L58, L70, L72, L325, L326,
 L327, L328, L336, L338, L351,
 L352, L353, L354, L361, L364,
 L384, L385, L386, L387, L393,
 L395, L407, L408, L409, L410,
 L415, L420, L729, L785, L813, L830
- Q**
- `\q bezier` 670
`\q bezier` L682, L814, L831
`\q beziermax` L681, L707, L708, L768
`\qqquad` o473
`\quad` o473, H155, H157, H177, N111
quark commands:
 `\q_mark` g1499, g1503,
 g1515, g1792, g1800, g1803, g1808
 `\q_nil` 143, 145, g511,
 g1026, g1047, g1051, g1057, g1066,
 g1068, g1419, g1443, g1452, g1484,
 g1499, g1503, g1515, g1521, g1547
 `\quark_if_nil:NTF` ... 145, g522, g1505
 `\quark_if_recursion_tail_stop:N` g707
 `\quark_if_recursion_tail_stop:n` .
 g407, g575, g1805, g1806, h1171
 `\quark_if_recursion_tail_stop_-`
 do:Nn g530
 `\quark_if_recursion_tail_stop_-`
 do:nn g433, g438, g447,
 g456, g470, g483, g497, g507, g551, i218
 `\quark_new:N` g1035
 `\q_recursion_stop` g261,
 g272, g384, g697, g1118, g1133,
 g1135, g1142, g1795, g1801, h1177, i209
 `\q_recursion_tail` 116, 117,
 g384, g697, g1133, g1137, g1791,
 g1794, g1800, g1801, h1176, h1177, i209

File Key: a=ltldirchk.dtx, b=ltplain.dtx, c=ltvers.dtx, d=ltluatex.dtx, e=ltexpl.dtx, f=ltdefns.dtx, g=ltcmd.dtx, h=lthooks.dtx, i=ltcmdhooks.dtx, j=ltalloc.dtx, k=ltcntrl.dtx, l=lterror.dtx, m=ltpar.dtx, n=ltpara.dtx, o=ltspace.dtx, p=ltlogos.dtx, q=ltfiles.dtx, r=ltoutenc.dtx, s=ltcounts.dtx, t=ltlength.dtx, u=ltfssbas.dtx, v=ltfssaxes.dtx, w=ltfssstrc.dtx, x=ltfsscmp.dtx, y=ltfssdcl.dtx, z=ltfssini.dtx, A=fontdef.dtx, B=preload.dtx, C=ltfntcmd.dtx, D=lttextcomp.dtx, E=ltpageno.dtx, F=ltxref.dtx, G=ltmiscen.dtx, H=ltmath.dtx, I=ltlists.dtx, J=ltboxes.dtx, K=lttab.dtx, L=ltpicture.dtx, M=ltthm.dtx, N=ltsect.dtx, O=ltfloat.dtx, P=ltidxglo.dtx, Q=ltbibl.dtx, R=ltpage.dtx, S=ltclass.dtx, T=ltfilehook.dtx, U=ltshipout.dtx, V=ltoutput.dtx, W=lthyphen.dtx, X=ltfinal.dtx

- `\q_stop` ... [g349](#), [g511](#), [g520](#), [g525](#),
[g1051](#), [g1068](#), [g1656](#), [g1817](#), [g1826](#)
 quark internal commands:
`\q_cmd` ... [114](#), [142](#),
[143](#), [145](#), [146](#), [g333](#), [g337](#), [g862](#),
[g1033](#), [g1403](#), [g1404](#), [g1407](#), [g1412](#),
[g1418](#), [g1419](#), [g1421](#), [g1425](#), [g1430](#),
[g1433](#), [g1436](#), [g1447](#), [g1451](#), [g1452](#),
[g1455](#), [g1456](#), [g1459](#), [g1470](#), [g1473](#),
[g1477](#), [g1488](#), [g1490](#), [g1491](#), [g1492](#),
[g1493](#), [g1494](#), [g1495](#), [g1496](#), [g1500](#),
[g1519](#), [g1520](#), [g1521](#), [g1522](#), [g1523](#),
[g1524](#), [g1525](#), [g1526](#), [g1527](#), [g1528](#),
[g1529](#), [g1530](#), [g1531](#), [g1532](#), [g1535](#),
[g1540](#), [g1546](#), [g1547](#), [g1553](#), [g1558](#),
[g1559](#), [g1562](#), [g1573](#), [g1580](#), [g1585](#),
[g1586](#), [g1587](#), [g1590](#), [g1591](#), [g1593](#)
`\quotedblbase` ... [r555](#), [r783](#), [r1168](#)
`\quotesinglbase` ... [r556](#), [r1165](#)
- R**
- `\r` ... [b399](#),
[b400](#), [r236](#), [r388](#), [r431](#), [r470](#), [r610](#),
[r637](#), [r647](#), [r673](#), [r756](#), [r795](#), [r1237](#),
[r1255](#), [r1281](#), [r1403](#), [r1404](#), [D183](#), [D205](#)
`\radical` ... [y921](#), [y924](#), [y954](#)
`\raggedbottom` ... [R53](#)
`\raggedleft` ... [G367](#), [G385](#), [G396](#), [G403](#)
`\raggedright` ... [G362](#), [G381](#), [G395](#), [G401](#)
`\raise` ... [r327](#), [r359](#),
[r430](#), [r433](#), [r732](#), [r797](#), [r895](#), [r1245](#),
[z504](#), [A465](#), [A513](#), [A515](#), [H73](#),
[J456](#), [J465](#), [L61](#), [L73](#), [L105](#), [L117](#),
[L195](#), [L305](#), [L452](#), [L495](#), [L526](#),
[L551](#), [L619](#), [L636](#), [L637](#), [L740](#), [L796](#)
`\raisebox` ... [630](#)
`\raisebox` ... [r872](#), [r1214](#), [J433](#)
`\rangle` ... [A592](#)
`\RawIndent` ... [269](#), [n97](#), [n133](#)
`\RawNoIndent` ... [n97](#)
`\RawNoindent` ... [269](#), [n98](#), [n134](#)
`\RawParEnd` ... [269](#), [n97](#), [n135](#)
`\RawShipout` ...
... [825–828](#), [831](#), [833–836](#), [U151](#), [U429](#)
`\rbrace` ... [r309](#), [A596](#)
`\rbrack` ... [b397](#)
`\rceil` ... [A600](#)
`\Re` ... [A313](#)
`\read` ... [317](#)
`\ReadOnlyShipoutCounter` ...
... [829](#), [831](#), [851](#), [U346](#), [U431](#)
`\Ref` ... [573](#)
`\ref` ... [F47](#), [F52](#), [F59](#), [F63](#), [F70](#)
`\ref` ... [626](#), [F10](#), [F47](#), [F59](#), [O535](#)
- `\refstepcounter` ... [366](#)
`\refstepcounter` ...
... [608](#), [F39](#), [F40](#), [F52](#), [F54](#), [F63](#),
[F65](#), [H337](#), [H476](#), [M27](#), [N59](#), [O9](#), [I202](#)
`\registernumber` ... [43](#)
`registernumber` ... [d381](#)
`\relax` ... [84](#),
[96](#), [265](#), [413](#), [586](#), [782](#), [815](#), [822](#), [848](#)
`\Relbar` ... [A476](#), [A484](#), [A486](#), [A492](#)
`\relbar` ... [A473](#), [A488](#), [A490](#)
`\relpenalty` ... [b324](#)
 remove commands:
... `remove_from_callback` ... [d747](#)
`\remove_from_callback` ... [44](#)
`\RemoveFromHook` .. [167](#), [173](#), [h1164](#), [h1279](#)
`\removelastskip` .. [b437](#), [b439](#), [b441](#), [b443](#)
`\renewcommand` ... [74](#)
`\renewcommand` ...
... [91](#), [483](#), [489](#), [511](#), [f124](#), [A66](#), [A68](#),
[A70](#), [A71](#), [A73](#), [A75](#), [A77](#), [A78](#),
[A84](#), [A86](#), [A88](#), [A89](#), [A103](#), [A104](#),
[A105](#), [A113](#), [A114](#), [H424](#), [H444](#), [H465](#)
`\RenewCommandCopy` ... [91](#), [f461](#), [f463](#)
`\RenewDocumentCommand` ... [g2127](#)
`\RenewDocumentEnvironment` . [g1924](#), [g2163](#)
`\renewenvironment` ... [75](#)
`\renewenvironment` ... [f152](#), [H473](#), [H485](#)
`\RenewExpandableDocumentCommand` ...
... [162](#), [g2179](#)
`\repeat` ... [a81](#), [a83](#), [b413](#), [d154](#),
[d164](#), [u688](#), [K382](#), [S1083](#), [S1144](#),
[S1214](#), [S1277](#), [S1328](#), [S1366](#), [X355](#),
[X366](#), [X376](#), [X387](#), [X417](#), [X443](#), [X453](#)
`\requestedLaTeXdate` ...
... [S1429](#), [S1462](#), [S1482](#), [S1568](#)
`\requestedpatchdate` ... [S1492](#), [S1569](#)
`\RequirePackage` ... [756](#)
`\RequirePackage` [170](#), [183](#), [804](#), [830](#), [d24](#),
[S597](#), [S604](#), [S631](#), [S640](#), [S1030](#), [V1967](#)
`\RequirePackageWithOptions` ... [756](#)
`\RequirePackageWithOptions` ... [S623](#)
 reserved@a commands:
... `\reserved@a:` ...
... [q232](#), [q303](#), [q350](#), [S1060](#), [S1191](#)
 reserved@b commands:
... `\reserved@b:` ... [S237](#), [S254](#)
 reserved@c commands:
... `\reserved@c:` ... [q649](#)
`\reservedb` ... [377](#)
`\restorecr` ... [o480](#)
`\ReverseBoolean` ... [g2244](#)
`\reversemarginpar` ... [O387](#)
`\rfloor` ... [A604](#)

File Key: a=ltldirchk.dtx, b=ltplain.dtx, c=ltvers.dtx, d=ltluatex.dtx, e=ltexpl.dtx, f=ltdefns.dtx, g=ltcmd.dtx, h=lthooks.dtx, i=ltcmdhooks.dtx, j=ltalloc.dtx, k=ltcntrl.dtx, l=lterror.dtx, m=ltpar.dtx, n=ltpara.dtx, o=ltspace.dtx, p=ltlogos.dtx, q=ltfiles.dtx, r=ltoutenc.dtx, s=ltcounts.dtx, t=ltlength.dtx, u=ltfssbas.dtx, v=ltfssaxes.dtx, w=ltfssstrc.dtx, x=ltfsscmp.dtx, y=ltfssdcl.dtx, z=ltfssini.dtx, A=fontdef.dtx, B=preload.dtx, C=ltfntcmd.dtx, D=lttextcomp.dtx, E=ltpageno.dtx, F=ltxref.dtx, G=ltmiscen.dtx, H=ltmath.dtx, I=ltlists.dtx, J=ltboxes.dtx, K=lttab.dtx, L=ltpicture.dtx, M=ltthm.dtx, N=ltsect.dtx, O=ltfloat.dtx, P=ltidxglo.dtx, Q=ltbibl.dtx, R=ltpage.dtx, S=ltclass.dtx, T=ltfilehook.dtx, U=ltshipout.dtx, V=ltoutput.dtx, W=lthyphen.dtx, X=ltfinal.dtx

- `\rgroup` [A608](#)
`\rhd` [z586](#)
`\rho` [A283](#)
`\rhohook` [A479, A480](#)
`\right` . [A625, A627, A629, A631, A636,](#)
 [A637, A638, A639, H155, H160, H184](#)
`\Rightarrow` [A411, A486, A498](#)
`\rightarrow` [A438,](#)
 [A440, A444, A478, A488, A496, A549](#)
`\rightarrowfill` [A531, A547](#)
`\rightharpoondown` [A455](#)
`\rightharpoonup` [A454, A466](#)
`\righthypphenmin` [W11](#)
`\rightleftharpoons` [A464](#)
`\rightline` [J472](#)
`\rightmargin` [615, I9, I40, I51](#)
`\rightmark` [R48](#)
`\rightskip` .. [b451, u659, G359, G363,](#)
 [G369, G379, G382, G387, G419,](#)
 [G441, J295, J316, N192, N215, I75](#)
`\rlap` [r430, r433, r797, H414, H425, J476, K81](#)
`\rmdefault` [364,](#)
 [487, 491, 492, z6, z143, z191, z239,](#)
 [z271, z303, z335, A50, A120, D7, D613](#)
`\rmfamily` [185, 482, 486, 495, z4,](#)
 [z5, z237, z301, z302, z333, z334, C15](#)
`rmfamily` [z252](#)
`\rmoustache` [A565](#)
`\rmsubstdefault` [A18, A30, D28, D39, D85](#)
`\Roman` [366](#)
`\Roman` [829, s104](#)
`\roman` [366](#)
`\roman` [s103](#)
`\romannumeral` [587, s109,](#)
 [s110, G192, G209, G255, I43, I234, I245](#)
`\root` [H66, H343](#)
`\rootbox` [H66](#)
`\rq` [b395](#)
`\rule` [630](#)
`\rule` [J383, J401, J407, O476, O495](#)
- S**
- `\S` [r311](#)
`\samepage` [279](#)
`\samepage` [o11, o28](#)
`\savebox` [629](#)
`\savebox` [J116](#)
`\savecatcodetable` [d117, d168, d170](#)
`\sb` [H199](#)
`\sbox` [629](#)
`\sbox` [b445, p4, r498,](#)
 [r514, J122, J129, J133, J138, J143, I205](#)
 scan commands:
 `\scan_new:N` [h40](#)
- `\scan_stop:` [110, 274, g1265,](#)
 [g1362, h339, h346, h375, i244, i245,](#)
 [i261, i262, U44, U325, U327, U328](#)
 scan internal commands:
 `\s__file_stop`
 [T103, T105, T108, T110, T112, T125](#)
 `\s__hook_mark` [h40, h196, h199, h202,](#)
 [h206, h209, h210, h330, h397, h416,](#)
 [h417, h421, h949, h962, h967, h975,](#)
 [i16, i18, i93, i98, i249, i252, i266, i278](#)
 `\scdefault` [v694, z27, A94](#)
 `\scriptfont` [w338](#)
 `\scriptscriptfont` [w339](#)
 `\scriptscriptstyle` [H65, H68](#)
 `\scriptspace` [b370](#)
 `\scriptstyle` [A337, H64](#)
 `\scshape` .. [r300, v692, v693, z25, z26, C23](#)
 `\searrow` [A406](#)
 `\sec` [H20](#)
 `\secdef` [N142](#)
 `\secondoftwo` [312](#)
 `\sectionmark` [N143](#)
 `\selectfont` [185, 186, 412, 414, 415, 421,](#)
 [429, 430, 491, 492, 501, 503, p7,](#)
 [r302, r329, r360, r449, r809, r871,](#)
 [r1213, r1247, r1561, u291, u301,](#)
 [u311, v528, v533, v538, v684, v688,](#)
 [v691, v694, v697, v700, v703, v706,](#)
 [w114, w115, w159, w162, w164, z6,](#)
 [z9, z12, z15, z18, z21, z24, z27,](#)
 [z30, z216, z235, z241, z246, z251,](#)
 [z287, z298, z305, z310, z315, z329,](#)
 [z332, z335, z338, z341, z417, z494,](#)
 [z518, z535, z550, D36, D94, D104,](#)
 [D636, D673, D916, D933, O403, O424](#)
 `selectfont` [w150](#)
 seq commands:
 `\seq_clear:N` [h600](#)
 `\seq_clear_new:N` [h608](#)
 `\seq_gpop:NNTF` ... [h1200, h1208, T72](#)
 `\seq_gpop_right:NN` [h1179](#)
 `\seq_gpush:Nn` [h1193, T64](#)
 `\seq_gput_right:Nn` . [h84, h1168, h1172](#)
 `\seq_if_empty:NNTF` [h1167, h1218](#)
 `\seq_if_exist:NNTF` [T60](#)
 `\seq_map_inline:Nn`
 [g1617, h548, h624, h642](#)
 `\seq_mapthread_function:NNN` ... [153](#)
 `\seq_new:N` . [g1601, h28, h35, h593, T61](#)
 `\seq_put_right:Nn` .. [h606, h699, h706](#)
 `\seq_set_split:Nnn` [g1615](#)
 `\seq_use:Nnnn` [h759, h764](#)

File Key: a=ltldirchk.dtx, b=ltplain.dtx, c=ltvers.dtx, d=ltluatex.dtx, e=ltexpl.dtx, f=ltdefns.dtx, g=ltcmd.dtx, h=lt hooks.dtx, i=ltcmdhooks.dtx, j=ltalloc.dtx, k=ltcntrl.dtx, l=lterror.dtx, m=ltpar.dtx, n=lt para.dtx, o=lt space.dtx, p=ltlogos.dtx, q=ltfiles.dtx, r=ltoutenc.dtx, s=ltcounts.dtx, t=ltlength.dtx, u=ltfssbas.dtx, v=ltfssaxes.dtx, w=ltfssstrc.dtx, x=ltfsscmp.dtx, y=ltfssdcl.dtx, z=ltfssini.dtx, A=fontdef.dtx, B=preload.dtx, C=ltfntcmd.dtx, D=lttextcomp.dtx, E=ltpageno.dtx, F=ltxref.dtx, G=ltmiscen.dtx, H=ltmath.dtx, I=ltlists.dtx, J=ltboxes.dtx, K=lttab.dtx, L=lt pictur.dtx, M=ltthm.dtx, N=ltsect.dtx, O=ltfloat.dtx, P=ltidxglo.dtx, Q=ltbibl.dtx, R=ltpage.dtx, S=ltclass.dtx, T=ltfilehook.dtx, U=ltshipout.dtx, V=ltoutput.dtx, W=lthyphen.dtx, X=ltfinal.dtx

- \seriesdefault
 487, 504, r1562, y271, z155, z157,
 z513, z531, z547, z564, z626, A120
- \setattribute 42
- \setattribute d82, d231
- \setcounter 366
- \setcounter q381, s2,
 s37, V2290, V2293, V2296, V2300, I225
- \SetDefaultHookLabel 169, 170, 226, h1166
- \setlength 374
- \setlength 284, 293, o83, o243,
 o442, t4, H492, H497, H498, H499,
 J43, J204, J265, J268, J341, J423,
 J424, J425, J454, J455, J462, J463,
 J464, K176, K384, V2306, V2307,
 V2308, V2311, V2312, V2316,
 V2317, V2318, V2322, V2323, V2324
- \SetMathAlphabet
 .. u11, x140, x141, y512, A155, A156
- \setminus A393
- \setrangeatcode .. d96, d104, d113, d114
- \SetSymbolFont .. y367, A145, A146, A147
- \settodepth 374
- \settodepth t17
- \settoheight 374
- \settoheight t17
- \settowidth 374
- \settowidth t17
- \sfcode . b387, b388, b389, b390, b474,
 o416, q45, q114, q172, X236, X536
- \sfdefault z9,
 z147, z192, z244, z272, z308, z338, A50
- \sffamily 185, 482, 492, 495, z7,
 z8, z242, z306, z307, z336, z337, C16
- sffamily z252
- \sfsubstdefault A19, A31, D30, D87
- \shapedefault
 . 417, 487, 497, 512, r1562, v684,
 y272, z514, z532, z548, z565, A120
- \sharp A330
- \shipout 185, 825–
 828, 831, 833, 835, 837, 838, 848,
 851, U4, U52, U422, U425, V602, V660
- shipout commands:
- \l_shipout_box 826,
 834, 836, 837, 839, 841, U23, U35,
 U37, U50, U61, U126, U149, U179,
 U184, U207, U208, U215, U226,
 U229, U230, U234, U241, U251,
 U259, U266, U276, U282, U283,
 U286, U293, U295, U296, U302, U304
- \l_shipout_box_dp_dim 826,
 U194, U197, U199, U230, U283, U503
- \l_shipout_box_ht_dim 826, U193,
 U197, U199, U229, U249, U282, U502
- \l_shipout_box_ht_plus_dp_dim ...
 826,
 U196, U199, U215, U266, U277, U279
- \l_shipout_box_wd_dim
 . 826, U195, U199, U241, U293, U504
- \shipout_debug_off: .. 830, U7, U406
- \shipout_debug_on: ... 830, U7, U405
- \shipout_discard: ... 828, U343, U402
- \g_shipout_readonly_int
 829, 835, U103, U105,
 U112, U117, U346, U355, U359, U363
- \g_shipout_totalpage_int 829
- \g_shipout_totalpages_int
 829, 835, U87, U348
- shipout internal commands:
- __shipout_add_background_box:n .
 U180, U206, U338, U411
- __shipout_add_background_-
 picture:n U69, U337, U415
- __shipout_add_firstpage_-
 material:Nn U172, U189, U404, U409
- __shipout_add_firstpage_-
 specials:
 ... 836, 838, 839, U111, U165, U178
- __shipout_add_foreground_box:n .
 U118, U257, U341, U413
- __shipout_add_foreground_-
 picture:n U64, U340, U417
- __shipout_debug:n
 832, U7, U104, U116, U148
- \g__shipout_debug_bool
 U6, U10, U15, U21
- __shipout_debug_gset: U7
- \g__shipout_discard_bool
 U81, U88, U90, U203, U344
- __shipout_drop_firstpage_-
 specials: 838, 839, U127, U166, U178
- __shipout_excuse_extra_page: ...
 U375, U383
- __shipout_execute: .. 834, U46, U52
- __shipout_execute_cont: .. U57, U59
- __shipout_execute_main_cont:Nnnn
 834, 837, U60, U77, U147
- __shipout_execute_nohooks_cont:
 U144, U146
- __shipout_execute_raw:
 837, U135, U151
- __shipout_execute_test_level: ..
 U49, U54
- __shipout_execute_test_level_-
 raw: U135
- __shipout_finalize_box: . U26, U124

File Key: a=ltldirchk.dtx, b=ltplain.dtx, c=ltvers.dtx, d=ltluatex.dtx, e=ltexpl.dtx, f=ltdefs.dtx, g=ltcmd.dtx, h=lt hooks.dtx, i=ltcmdhooks.dtx, j=ltalloc.dtx, k=ltcntrl.dtx, l=lterror.dtx, m=ltpar.dtx, n=lt para.dtx, o=lt space.dtx, p=ltlogos.dtx, q=ltfiles.dtx, r=ltoutenc.dtx, s=ltcounts.dtx, t=ltlength.dtx, u=ltfssbas.dtx, v=ltfssaxes.dtx, w=ltfssstrc.dtx, x=ltfsscmp.dtx, y=ltfssdcl.dtx, z=ltfssini.dtx, A=fontdef.dtx, B=preload.dtx, C=ltfntcmd.dtx, D=lttextcomp.dtx, E=ltpageno.dtx, F=ltxref.dtx, G=ltmiscen.dtx, H=ltmath.dtx, I=ltlists.dtx, J=ltboxes.dtx, K=lttab.dtx, L=lt pictur.dtx, M=ltthm.dtx, N=ltsect.dtx, O=ltfloat.dtx, P=ltidxglo.dtx, Q=ltbibl.dtx, R=lt page.dtx, S=ltclass.dtx, T=ltfilehook.dtx, U=ltshipout.dtx, V=ltoutput.dtx, W=lthyphen.dtx, X=ltfinal.dtx

- \l__shipout_firstpage_box [839](#), [U169](#), [U180](#), [U187](#)
- __shipout_get_box_size:N [840](#), [U85](#), [U108](#), [U192](#), [U207](#)
- \l__shipout_group_level_tl [U47](#), [U53](#), [U56](#), [U136](#), [U143](#)
- \c__shipout_horigin_tl .. [U310](#), [U325](#)
- __shipout_init_page_origins: ... [U310](#), [U324](#)
- \g__shipout_lastpage_handled_-bool [U122](#), [U188](#), [U365](#)
- __shipout_picture_overlay:n ... [U323](#), [U338](#), [U341](#)
- \l__shipout_raw_box [836](#), [U25](#), [U139](#), [U147](#), [U149](#), [U179](#), [U184](#)
- __shipout_run_firstpage_hook: .. [836](#), [838](#), [839](#), [U109](#), [U162](#)
- \l__shipout_saved_badness_tl ... [U204](#), [U210](#), [U218](#), [U231](#), [U236](#), [U244](#), [U253](#), [U261](#), [U269](#), [U281](#), [U288](#), [U298](#), [U306](#)
- __shipout_saved_protect: [U83](#), [U132](#), [U152](#)
- \l__shipout_tmp_box [U204](#), [U217](#), [U219](#), [U220](#), [U221](#), [U225](#), [U243](#), [U245](#), [U246](#), [U247](#), [U250](#), [U268](#), [U270](#), [U271](#), [U272](#), [U278](#), [U297](#), [U299](#), [U300](#), [U301](#), [U303](#), [U329](#), [U331](#), [U332](#), [U333](#)
- \c__shipout_vorigin_tl .. [U310](#), [U327](#)
- shipout/after [826](#), [U153](#)
- shipout/background [826](#), [U153](#)
- shipout/before [826](#), [U153](#)
- shipout/firstpage [826](#), [U153](#)
- shipout/foreground [826](#), [U153](#)
- shipout/lastpage [826](#), [U153](#)
- \ShipoutBox [825–827](#), [831](#), [839](#), [U23](#), [U430](#), [U485](#)
- \ShipoutBoxDepth [U450](#), [U502](#)
- \ShipoutBoxHeight [851](#), [U449](#), [U502](#)
- \ShipoutBoxWidth [U451](#), [U502](#)
- \shortstack . [L132](#), [L147](#), [L152](#), [L815](#), [L832](#)
- \show [92](#), [94](#), [95](#), [f505](#), [f575](#), [f576](#)
- show commands:
 - \show_hook:n [215](#)
- \showboxbreadth [b359](#), [b477](#), [b554](#), [b578](#), [b595](#), [b620](#)
- \showboxdepth [b360](#), [b477](#), [b553](#), [b577](#), [b594](#), [b621](#), [u647](#), [u691](#), [u708](#)
- \ShowCommand [89](#), [92](#), [94](#), [f498](#)
- \ShowDocumentCommandArgSpec [g2249](#)
- \ShowDocumentEnvironmentArgSpec .. [g2249](#)
- \ShowHook [173](#), [174](#), [177](#), [179](#), [h1246](#), [h1284](#)
- \showhyphens [u636](#)
- \showoutput [b476](#)
- \showoverfull [b475](#), [b478](#), [b500](#), [b535](#), [b543](#)
- \showtokens [96](#), [f611](#)
- \Sigma [A303](#)
- \sigma [A284](#)
- \sim [A445](#), [A457](#)
- \simeq [A446](#)
- \sin [H9](#)
- \sinh [H11](#)
- \skew [A544](#)
- \skip [b28](#), [b53](#), [b92](#), [b208](#), [b250](#), [b295](#), [d33](#), [J358](#), [O391](#), [V316](#), [V490](#)
- skip commands:
 - \skip_zero:N [n23](#), [U222](#), [U223](#), [U224](#), [U273](#), [U274](#), [U275](#)
- \skipdef [b45](#), [b53](#), [b92](#), [d221](#)
- \skipzero [d221](#)
- \slash [b430](#), [f785](#), [f806](#)
- \sldefault [v691](#), [z24](#), [A94](#)
- \sloppy [J300](#), [J319](#), [R57](#), [R62](#)
- sloppypar (environment) [R62](#)
- \sloppypar [R62](#)
- \slshape [r440](#), [r800](#), [v689](#), [v690](#), [z22](#), [z23](#), [C22](#), [D627](#)
- \small [168](#)
- \smallbreak [b438](#), [f786](#), [f807](#)
- \smallint [A357](#)
- \smallskip [b439](#), [o400](#)
- \smallskipamount [b438](#), [o400](#), [o403](#)
- \smash [A473](#), [A547](#), [A548](#), [A551](#), [A552](#), [H126](#)
- \smile [A450](#)
- \sourceLaTeXdate [c152](#), [S67](#), [S105](#)
- \sp [H199](#)
- \space [b403](#)
- \spacefactor [b428](#), [b429](#), [o129](#), [o138](#), [o157](#), [o171](#), [o183](#), [o197](#), [o211](#), [o416](#), [o429](#), [o434](#), [r70](#), [r73](#), [O513](#), [O515](#)
- \spaceskip [D6](#), [D612](#)
- \spadesuit [A334](#)
- \span [K383](#)
- \special [185](#), [827](#), [836](#), [838–840](#), [851](#)
- \SplitArgument [g2244](#)
- \splitfirstmark [V2211](#)
- \SplitList [164](#), [g2244](#)
- \splitmaxdepth . [b366](#), [O469](#), [O488](#), [V2205](#)
- \splittopskip [b378](#), [O468](#), [O487](#)
- \sqcap [A376](#)
- \sqcup [A377](#)
- \sqrt [H342](#)
- \sqrtsign [A529](#), [H71](#), [H342](#)
- \sqsubset [z582](#)
- \sqsubseteq [A399](#)
- \sqsupset [z583](#)
- \sqsupseteq [A400](#)

File Key: a=ltldirchk.dtx, b=ltplain.dtx, c=ltvers.dtx, d=ltluatex.dtx, e=ltexpl.dtx, f=ltdefns.dtx, g=ltcmd.dtx, h=lthooks.dtx, i=ltcmdhooks.dtx, j=ltalloc.dtx, k=ltcntrl.dtx, l=lterror.dtx, m=ltpar.dtx, n=ltpara.dtx, o=ltspace.dtx, p=ltlogos.dtx, q=ltfiles.dtx, r=ltoutenc.dtx, s=ltcounts.dtx, t=ltlength.dtx, u=ltfssbas.dtx, v=ltfssaxes.dtx, w=ltfssstrc.dtx, x=ltfsscmp.dtx, y=ltfssdcl.dtx, z=ltfssini.dtx, A=fontdef.dtx, B=preload.dtx, C=ltfntcmd.dtx, D=lttextcomp.dtx, E=ltpageno.dtx, F=ltxref.dtx, G=ltmiscen.dtx, H=ltmath.dtx, I=ltlists.dtx, J=ltboxes.dtx, K=lttab.dtx, L=ltpictur.dtx, M=ltthm.dtx, N=ltsect.dtx, O=ltfloat.dtx, P=ltidxglo.dtx, Q=ltbibl.dtx, R=ltpage.dtx, S=ltclass.dtx, T=ltfilehook.dtx, U=ltshipout.dtx, V=ltoutput.dtx, W=lthyphen.dtx, X=ltfinal.dtx

- `\SS` [r304](#), [r534](#), [r1149](#), [X571](#)
`\ss` ... [r254](#), [r406](#), [r557](#), [r784](#), [r1124](#), [X571](#)
`\sscdefault` [v536](#), [v609](#), [v706](#)
`\sscshape` ... [v536](#), [v608](#), [v704](#), [v705](#), [C31](#)
`\stackrel` [H340](#)
`\star` [A397](#)
`\stepcounter` [366](#)
`\stepcounter` [s17](#), [s27](#),
[u601](#), [y48](#), [F41](#), [F54](#), [F65](#), [H350](#),
[H409](#), [H486](#), [O452](#), [O502](#), [V646](#), [V705](#)
`\stop` [G165](#)
`\storedpar` [d156](#), [d161](#)
str commands:
`\c_backslash_str` [i120](#), [i212](#)
`\str_case:nn` [i302](#)
`\str_case:nnTF` [i21](#)
`\str_case_e:nnTF` [g1812](#)
`\str_count:n` [i132](#)
`\str_gset:Nn` [h1241](#)
`\str_head:n` [g1757](#)
`\str_if_eq:nn` [207](#)
`\str_if_eq:nnTF` [T516](#)
`\str_if_eq:nnTF` [142](#), [145](#), [e136](#), [g354](#),
[g1024](#), [g1396](#), [g1409](#), [g1411](#), [g1441](#),
[g1461](#), [g1463](#), [g1482](#), [g1509](#), [g1537](#),
[g1539](#), [g1564](#), [g1566](#), [g1589](#), [g1859](#),
[h159](#), [h194](#), [h295](#), [h297](#), [h355](#), [h399](#),
[h434](#), [h441](#), [h752](#), [h828](#), [h908](#), [h969](#),
[h1050](#), [h1082](#), [h1187](#), [h1227](#), [i164](#),
[i168](#), [i184](#), [i270](#), [i273](#), [T131](#), [T510](#), [T520](#)
`\str_if_eq_p:nn`
... [g413](#), [g414](#), [g415](#), [g416](#), [h403](#), [h635](#)
`\str_map_function:NN` [e140](#)
`\str_new:N` [g19](#)
`\str_set:Nn`
... [g177](#), [g178](#), [g219](#), [g1690](#), [T364](#), [T365](#)
`\str_tail:n` .. [142](#), [g1396](#), [g1754](#), [g1780](#)
`\str_uppercase:n` [g2056](#)
str internal commands:
`_str_if_eq:nn` [186](#), [h23](#)
`\strcmp` [S271](#), [S287](#)
`\stretch` [o456](#)
`\string` [812](#), [815](#)
`\strut` .. [b446](#), [f787](#), [f808](#), [H178](#), [H179](#), [K29](#)
`\strutbox` [b446](#), [w189](#), [J383](#), [J401](#),
[K186](#), [K187](#), [O469](#), [O476](#), [O488](#), [O495](#)
`\subparagraphmark` [N143](#)
`\subsectionmark` [N143](#)
`\subset` [A426](#)
`\subseteq` [A428](#)
`\subsubsectionmark` [N143](#)
`\succ` [A420](#)
`\succeq` [A423](#)
`\sum` [A350](#)
`\sup` [H24](#)
`\suppressfloats` [V1972](#)
`\supset` [A425](#)
`\supseteq` [A427](#)
`\surd` [A336](#)
`\swarrow` [A408](#)
`\swdefault` [v531](#), [v607](#), [v703](#)
`\swshape` [v531](#), [v606](#), [v701](#), [v702](#), [C30](#)
`\symbol` [r162](#), [z472](#)
`\symletters` [D8](#), [D14](#), [D614](#), [D1139](#)
`\symoperators` [A643](#)
sys commands:
`\sys_if_engine luatex:TF` [U26](#)

T
`\T` [l23](#), [r335](#),
[r337](#), [r339](#), [r341](#), [r343](#), [r345](#), [r347](#),
[r349](#), [r351](#), [r374](#), [S1409](#), [S1413](#), [S1414](#)
`\t` [r282](#), [r741](#), [r849](#), [D157](#), [D158](#),
[D185](#), [D189](#), [D191](#), [D203](#), [D206](#),
[D208](#), [D688](#), [D859](#), [D1126](#), [D1128](#)
tabbing (environment) [K71](#)
`\tabbing` [K71](#)
`\tabbingsep` [K130](#), [K132](#), [K166](#)
`\tabcolsep` [K259](#), [K338](#)
`\tableofcontents` [581](#)
`\tabskip` [b457](#), [H195](#),
[H196](#), [H355](#), [H358](#), [H361](#), [H363](#),
[H490](#), [H503](#), [H506](#), [H508](#), [K167](#), [K192](#)
tabular (environment) [K174](#)
`\tabular` [K174](#)
`\tabular*` [K175](#)
`\tabularnewline` [K194](#), [K207](#)
`\tan` [H15](#)
`\tanh` [H17](#)
`\tau` [A285](#)
`\tencirc` [B10](#), [L125](#), [L678](#)
`\tencircw` [B10](#), [L128](#)
`\tenln` [B9](#), [L124](#), [L126](#), [L677](#), [L679](#)
`\tenlnw` [B9](#), [L127](#), [L129](#)
`\TeX` [p1](#), [p12](#)
 \TeX and \LaTeX 2 ϵ commands:
`\...-h@k` [781](#)
`\...@without@substitution` [429](#)
`\@` [a65](#), [d20](#),
[d860](#), [f728](#), [f729](#), [i189](#), [i19](#), [o425](#),
[p2](#), [S45](#), [S60](#), [S73](#), [S82](#), [S120](#), [X503](#)
`\@...hook` [192](#), [193](#)
`\@@` [a331](#), [a332](#), [k15](#), [k19](#),
[k20](#), [k21](#), [k22](#), [k24](#), [k27](#), [k28](#), [k30](#),
[k31](#), [q647](#), [q663](#), [w510](#), [w512](#), [w513](#),
[K238](#), [K239](#), [K240](#), [K250](#), [V10](#), [V11](#)
`\@@defaults` [u585](#)
`\@enc@update` [r183](#), [u259](#), [u263](#)

File Key: a=ltldirchk.dtx, b=ltplain.dtx, c=ltvers.dtx, d=ltluatex.dtx, e=ltexpl.dtx,
f=ltdefns.dtx, g=ltcmd.dtx, h=lthooks.dtx, i=ltcmdhooks.dtx, j=ltalloc.dtx, k=ltcntrl.dtx,
l=lterror.dtx, m=ltpar.dtx, n=ltpara.dtx, o=ltspace.dtx, p=ltlogos.dtx, q=ltfiles.dtx,
r=ltoutenc.dtx, s=ltcounts.dtx, t=ltlength.dtx, u=ltfssbas.dtx, v=ltfssaxes.dtx,
w=ltfssstrc.dtx, x=ltfsscmp.dtx, y=ltfssdcl.dtx, z=ltfssini.dtx, A=fontdef.dtx,
B=preload.dtx, C=ltfntcmd.dtx, D=lttextcomp.dtx, E=ltpageno.dtx, F=ltxref.dtx,
G=ltmiscen.dtx, H=ltmath.dtx, I=ltlists.dtx, J=ltboxes.dtx, K=lttab.dtx, L=ltpictur.dtx,
M=ltthm.dtx, N=ltsect.dtx, O=ltfloat.dtx, P=ltidxglo.dtx, Q=ltbibl.dtx, R=ltpage.dtx,
S=ltclass.dtx, T=ltfilehook.dtx, U=ltshipout.dtx, V=ltoutput.dtx, W=lthyphen.dtx,
X=ltfinal.dtx

- \@@end 317, a69, a222, f23, q624, q625,
 G32, G99, G165, W18, X590, X611
 \@@endpbox K193, K236, K386
 \@@eqncr H367, H389, H399, H404, H509
 \@@fileswith@pti@ns S526, S545, S966
 \@@hyph f24, f745
 \@@hyphenation r205
 \@@if@newlist V599, V644, V657, V703
 \@@ifdefinable f132, r17
 \@@input
 a68, f22, q522, q535, q575, G26,
 G79, S1556, T172, T186, T194, X317
 \@@italiccorr f25, C113, C117
 \@@line J472
 \@@math@bgroup C131, C138
 \@@math@egroup C128
 \@@par 275, 277, f21, m4, n101,
 n138, G165, G421, G426, G429,
 G443, G447, G450, J266, J288,
 J309, K199, N67, N118, V257, I82, I85
 \@@patterns r205
 \@@protect f254, f260, f269
 \@@startpbox K193, K236, K386
 \@@sverb 595, G500
 \@@underline J428, J431, J432
 \@@unprocessedoptions
 S882, S943, S1020
 \@@warning l166
 \@Alpha 99, s106, s122
 \@DeclareEncodingSubset
 D45, D47, D48, D49, D50, D53, D60
 \@DeclareMathDelimiter ... y784, y803
 \@DeclareMathSizes . u206, u207, u209
 \@Esphack o190, O201, O223, O241
 \@IncludeInRele@se c68
 \@IncludeInRelease c68
 \@M b21,
 b431, b432, f37, f39, o11, o12, o13,
 o14, o15, o16, o17, o18, o119, u654,
 u661, w439, w452, H377, K56,
 N67, N100, N118, N130, N194,
 N217, V176, V195, V198, V258, I194
 \@MM 246, b21, O469, O488, V299
 \@Mi j3, V136
 \@Mii .. j3, O53, O122, O194, O216,
 O241, O311, V295, V1159, V1326
 \@Miii j3, O55, O124, O313, V298
 \@Miv j3, O195, O201, O217, O223, V272
 \@Roman s104, s110
 \@TeXversion 2, a326, l28
 \@abspage@last 845, 847, U106, U112,
 U351, U353, U355, U363, U400, U401
 \@acci z628, J290, J311
 \@accii z628, J290, J311
 \@acciii z628, J290, J311
 \@acol
 K168, K178, K260, K261, K273,
 K274, K277, K294, K309, K317, K327
 \@acolampacol K258,
 K275, K277, K284, K292, K326, K329
 \@activechar@info V575
 \@addamp K251,
 K260, K261, K276, K290, K327, K328
 \@addfield K43,
 K53, K86, K93, K125, K140, K142
 \@addmarginpar V331, V1811
 \@addtobot V975,
 V1062, V1129, V1181, V1290, V1349
 \@addtocurcol ... V328, V1066, V1965
 \@addtodblcol V854, V1562
 \@addtofilelist a101,
 a103, q64, q132, q187, q522, q644,
 z602, z605, z612, z615, z622, z625,
 T169, T186, T194, X265, X268, X631
 \@addtonextcol .. V853, V1386, V1966
 \@addtopreamble K311,
 K324, K330, K331, K332, K334, K346
 \@addtoreset ... s16, s39, s44, s79, s82
 \@addtotoporbot
 V1012, V1175, V1343, V1435, V1524
 \@afterheading N92, N125
 \@afterindentfalse N45
 \@afterindenttrue
 N43, N124, N193, N216
 \@alph 99, s105, s118, O399
 \@ampacol ... K258, K275, K286, K329
 \@arabic s43, s76, s87, s102, s108, O397
 \@argarraycr K203, K204
 \@argdef f80
 \@argrsbox J452
 \@argtabularcr K210, K211
 \@array K181, K182
 \@arrayacol K168, K258
 \@arrayclassiv K169, K331
 \@arrayclassz K168, K275
 \@arraycr K170, K201, K203
 \@arrayparboxrestore J280, J322, K384
 \@arrayrule
 K309, K311, K315, K317, K319, K346
 \@arstrut K192, K237, K343
 \@arstrutbox K185, K218, K343, K385
 \@author N8, N32
 \@auxout
 q216, q222, q266, q284, q307,
 q336, q354, q369, F33, N181, Q7,
 Q8, Q19, Q29, Q37, Q47, Q64, U362

File Key: a=ltldirchk.dtx, b=ltplain.dtx, c=ltvers.dtx, d=ltluatex.dtx, e=ltexpl.dtx,
 f=ltdefns.dtx, g=ltcmd.dtx, h=lt hooks.dtx, i=ltcmdhooks.dtx, j=ltalloc.dtx, k=ltcntrl.dtx,
 l=lterror.dtx, m=ltpar.dtx, n=lt para.dtx, o=lt space.dtx, p=ltlogos.dtx, q=ltfiles.dtx,
 r=ltoutenc.dtx, s=ltcounts.dtx, t=ltlength.dtx, u=ltfssbas.dtx, v=ltfssaxes.dtx,
 w=ltfssstrc.dtx, x=ltfsscmp.dtx, y=ltfssdcl.dtx, z=ltfssini.dtx, A=fontdef.dtx,
 B=preload.dtx, C=ltfntcmd.dtx, D=lttextcomp.dtx, E=ltpageno.dtx, F=ltxref.dtx,
 G=ltmiscen.dtx, H=ltmath.dtx, I=ltlists.dtx, J=ltboxes.dtx, K=lttab.dtx, L=lt pictur.dtx,
 M=ltthm.dtx, N=ltsect.dtx, O=ltfloat.dtx, P=ltidxglo.dtx, Q=ltbibl.dtx, R=ltpage.dtx,
 S=ltclass.dtx, T=ltfilehook.dtx, U=ltshipout.dtx, V=ltoutput.dtx, W=lthyphen.dtx,
 X=ltfinal.dtx

- \@backslashchar [f219](#),
[f469](#), [f582](#), [f594](#), [f598](#), [f599](#), [f604](#),
[l185](#), [l187](#), [A266](#), [S1145](#), [S1278](#), [S1367](#)
- \@badcrerr [l227](#)
- \@badend [l198](#), [G297](#)
- \@badlinearg
... [l217](#), [L165](#), [L177](#), [L188](#), [L189](#),
[L193](#), [L242](#), [L247](#), [L257](#), [L262](#), [L275](#)
- \@badmath . [l201](#), [H262](#), [H264](#), [H269](#),
[H272](#), [H281](#), [H293](#), [H298](#), [H307](#),
[H320](#), [H325](#), [H433](#), [H445](#), [H461](#), [H470](#)
- \@badpoptabs [l205](#), [K85](#), [K151](#)
- \@badrequireerror [S420](#), [S1028](#)
- \@badtab [l208](#),
[K22](#), [K87](#), [K108](#), [K114](#), [K121](#), [K148](#)
- \@begin@tempboxa
... [J27](#), [J42](#), [J203](#), [J266](#), [J453](#), [J461](#)
- \@begin@documenthook
... [181](#), [192](#), [193](#), [q61](#), [q126](#),
[q129](#), [q181](#), [q184](#), [Q33](#), [S977](#), [S998](#)
- \@begin@divi [832](#), [V623](#), [V682](#), [V710](#)
- \@begin@divbox
... [827](#), [U440](#), [U441](#), [V86](#), [V711](#)
- \@begin@parpenalty
[615](#), [o14](#), [H436](#), [H448](#), [H474](#), [l23](#), [l170](#)
- \@begin@theorem [M30](#), [M35](#)
- \@bezier [L683](#), [L684](#)
- \@bibitem [Q3](#), [Q8](#)
- \@biblabel [Q4](#), [Q77](#)
- \@bitor
.. [V15](#), [V881](#), [V901](#), [V937](#), [V960](#),
[V1027](#), [V1111](#), [V1121](#), [V1269](#),
[V1280](#), [V1422](#), [V1509](#), [V1627](#), [V1752](#)
- \@botlist [V65](#),
[V384](#), [V386](#), [V431](#), [V433](#), [V717](#),
[V738](#), [V747](#), [V748](#), [V989](#), [V992](#),
[V1027](#), [V1121](#), [V1280](#), [V1922](#), [V1949](#)
- \@botnum [O274](#), [V109](#), [V986](#),
[V987](#), [V992](#), [V996](#), [V1458](#), [V1463](#),
[V1551](#), [V1558](#), [V1914](#), [V1941](#), [V1983](#)
- \@botroom [O275](#),
[V110](#), [V989](#), [V992](#), [V1915](#), [V1942](#)
- \@box@f@psbit [V2031](#), [V2033](#), [V2038](#)
- \@break@tfor ... [k31](#), [q495](#), [q512](#), [C98](#)
- \@bsphack ... [o36](#), [o125](#), [o340](#), [o356](#),
[o374](#), [o390](#), [F32](#), [O52](#), [O121](#), [O310](#),
[P6](#), [P18](#), [P23](#), [P35](#), [Q43](#), [Q60](#), [V1882](#)
- \@caption [O12](#), [O14](#)
- \@c@p@t@y@e [O5](#), [O9](#),
[O12](#), [O40](#), [O88](#), [O109](#), [O157](#), [V1995](#)
- \@car [f53](#), [p14](#), [r89](#), [r110](#)
- \@carcube [f55](#), [f135](#), [f585](#)
- \@cc@l@v [831](#), [b16](#), [V300](#),
[V304](#), [V382](#), [V383](#), [V412](#), [V429](#),
[V430](#), [V459](#), [V483](#), [V487](#), [V488](#), [X53](#)
- \@cc@l@v@i
[b21](#), [b57](#), [b82](#), [b93](#), [b95](#), [b99](#), [b159](#),
[b173](#), [d30](#), [d58](#), [S1143](#), [S1276](#), [S1365](#)
- \@cdr [f53](#), [f671](#), [f672](#), [q250](#)
- \@center@cr [G320](#),
[G358](#), [G363](#), [G368](#), [G378](#), [G382](#), [G386](#)
- \@centering [H347](#),
[H348](#), [H355](#), [H358](#), [H361](#), [H502](#), [H506](#)
- \@cflb [V714](#)
- \@cflt [V714](#)
- \@changed@cmd
... [r3](#), [r63](#), [r223](#), [u131](#), [u267](#), [X462](#)
- \@changed@x [r3](#), [r211](#), [r219](#)
- \@changed@x@mouth [r211](#), [r219](#)
- \@charlb [q375](#), [q383](#)
- \@charrb [q377](#), [q383](#)
- \@chclass [K271](#), [K272](#), [K335](#), [K348](#), [K353](#)
- \@check@IncludeInRelease [c68](#)
- \@check@c [f189](#), [f191](#)
- \@check@eq [f195](#), [f196](#), [f200](#)
- \@checkend [G16](#),
[G71](#), [G213](#), [G220](#), [G272](#), [G287](#), [G296](#)
- \@chnum
[K279](#), [K298](#), [K335](#), [K350](#), [K351](#), [K352](#)
- \@circ .. [L622](#), [L640](#), [L649](#), [L654](#), [L657](#)
- \@circle [L605](#), [L606](#)
- \@circlefnt [L125](#), [L128](#), [L447](#), [L491](#),
[L520](#), [L545](#), [L615](#), [L631](#), [L663](#), [L678](#)
- \@cite [Q16](#), [Q75](#)
- \@cite@ofmt [Q24](#), [Q76](#)
- \@citea [Q15](#), [Q17](#)
- \@citeb
.. [Q18](#), [Q19](#), [Q20](#), [Q23](#), [Q24](#), [Q46](#),
[Q47](#), [Q48](#), [Q49](#), [Q63](#), [Q64](#), [Q65](#), [Q66](#)
- \@citex [Q13](#), [Q14](#)
- \@classi [K271](#), [K307](#)
- \@classii [K271](#), [K321](#)
- \@classiii [K271](#), [K326](#)
- \@classiv [K169](#), [K180](#), [K272](#)
- \@classoptionslist
[S9](#), [S467](#), [S482](#), [S483](#), [S500](#), [S696](#),
[S697](#), [S725](#), [S726](#), [S752](#), [S753](#), [S1637](#)
- \@classv [K272](#), [K332](#)
- \@classz [K168](#), [K179](#), [K271](#)
- \@cline [K367](#)
- \@clnht [L195](#), [L196](#), [L204](#),
[L206](#), [L208](#), [L218](#), [L225](#), [L273](#), [L672](#)
- \@clnwd
.. [L197](#), [L203](#), [L207](#), [L209](#), [L210](#), [L672](#)

File Key: a=lt@dirchk.dtx, b=lt@plain.dtx, c=lt@vers.dtx, d=lt@luatex.dtx, e=lt@expl.dtx, f=lt@defs.dtx, g=lt@cmd.dtx, h=lt@hooks.dtx, i=lt@cmdhooks.dtx, j=lt@alloc.dtx, k=lt@cntrl.dtx, l=lt@error.dtx, m=lt@par.dtx, n=lt@para.dtx, o=lt@space.dtx, p=lt@logos.dtx, q=lt@files.dtx, r=lt@outenc.dtx, s=lt@counts.dtx, t=lt@length.dtx, u=lt@fssbas.dtx, v=lt@fssaxes.dtx, w=lt@fssstrc.dtx, x=lt@fsscmp.dtx, y=lt@fssdcl.dtx, z=lt@fssini.dtx, A=fontdef.dtx, B=preload.dtx, C=lt@fntcmd.dtx, D=lt@textcomp.dtx, E=lt@pageno.dtx, F=lt@xref.dtx, G=lt@miscen.dtx, H=lt@math.dtx, I=lt@lists.dtx, J=lt@boxes.dtx, K=lt@tab.dtx, L=lt@pictur.dtx, M=lt@hbm.dtx, N=lt@sect.dtx, O=lt@float.dtx, P=lt@idxglo.dtx, Q=lt@bibl.dtx, R=lt@page.dtx, S=lt@class.dtx, T=lt@filehook.dtx, U=lt@shipout.dtx, V=lt@output.dtx, W=lt@hyphen.dtx, X=lt@final.dtx

`\@cls@pkg` h102, S278,
 S279, S292, S293, S795, S841, S879,
 S922, S952, S1004, S1013, S1015,
 S1032, S1490, S1565, S1587, S1617
`\@clsextension` 781, 812, S31,
 S150, S161, S224, S308, S324, S336,
 S416, S438, S449, S467, S481, S499,
 S598, S613, S621, S695, S724, S751,
 S845, S872, S899, S956, S969, S1005
`\@clubpenalty` q7,
 q25, q94, q151, N106, N135, I128, I196
`\@colht` q22, q91, q148, O273,
 O275, O278, O284, O285, O298,
 O299, V114, V231, V242, V251,
 V252, V387, V399, V434, V447,
 V474, V505, V535, V541, V545,
 V555, V560, V645, V704, V777,
 V815, V859, V884, V903, V943,
 V965, V1642, V1768, V2096, X141
`\@colnum` O276, V111, V995,
 V1040, V1109, V1110, V1138,
 V1146, V1267, V1268, V1300,
 V1312, V1420, V1421, V1458,
 V1463, V1507, V1508, V1550,
 V1557, V1910, V1937, V1976, V2151
`\@colroom` q23, q92, q149,
 V115, V252, V273, V274, V285,
 V288, V387, V434, V777, V994,
 V1039, V1105, V1108, V1137,
 V1262, V1266, V1299, V1416,
 V1419, V1502, V1506, V1911,
 V1938, V2106, V2111, V2156, X140
`\@combinedblfloats` V750, V2230, V2269
`\@combinefloats` V501, V714
`\@comdblflflt` V750
`\@comflelt` V720, V736, V750
`\@cons`
 b196, b213, f52, s44, O193, O215,
 O239, O379, V237, V888, V907,
 V923, V947, V949, V969, V971,
 V1141, V1209, V1305, V1378,
 V1451, V1541, V1644, V1667,
 V1770, V1795, V1812, V1813, V2157
`\@contfield` K50, K141, K153
`\@copy@...` 92
`\@copy@DeclareRobustCommand`
 f483, f520, f541, f616, f619
`\@copy@newcommand` 86, 93,
 f306, f484, f520, f562, f591, f616, f622
`\@ctrerr` .. I194, s121, s125, s139, s147
`\@curfield` K16,
 K41, K47, K51, K52, K54, K130, K131
`\@curline`
 . K16, K27, K39, K44, K53, K54,
 K55, K90, K91, K103, K128, K129
`\@curr@enc` r154, r156
`\@curr@file` 779, 780, 815–817, q226,
 q227, q236, q238, q262, q270, q388,
 q407, q542, q557, S826, S1090,
 S1095, S1101, S1107, S1111, S1122,
 S1131, S1158, S1221, S1226, S1232,
 S1255, S1264, S1290, T265, T339, T341
`\@curr@file@reqd`
 816, 817, T265, T341, T345
`\@currbox` b275, b276, b277,
 O60, O91, O95, O129, O160, O164,
 O193, O214, O215, O239, O257,
 O259, O261, O319, O322, O327,
 O331, V213, V214, V225, V226,
 V228, V229, V237, V311, V312,
 V853, V854, V1102, V1104, V1112,
 V1135, V1139, V1141, V1156,
 V1197, V1209, V1257, V1260,
 V1297, V1302, V1305, V1322,
 V1367, V1378, V1410, V1426,
 V1440, V1451, V1493, V1530,
 V1541, V1581, V1585, V1596,
 V1602, V1604, V1608, V1613,
 V1622, V1631, V1637, V1644,
 V1667, V1702, V1706, V1718,
 V1725, V1727, V1731, V1737,
 V1747, V1762, V1770, V1795,
 V1813, V1822, V2001, V2002,
 V2031, V2061, V2066, V2112,
 V2115, V2127, V2135, V2152, V2157
`\@currdir`
 .. 9, a108, a130, a132, a138, a140,
 a146, a148, a153, a155, a165, a178,
 a243, a256, a269, S1073, S1095,
 S1122, S1204, S1226, S1255, S1338
`\@current@cmd` r25, u271
`\@currentcounter` .. F39, F40, F42, F53
`\@currentlabel`
 ... 608, F34, F43, F55, F66, F74,
 H351, H487, J379, J397, O471, O490
`\@currentvir`
 270, I199, G3, G176, G231,
 G248, G281, G297, J149, S1133,
 S1145, S1153, S1157, S1164, S1266,
 S1278, S1286, S1290, S1296, S1355,
 S1367, S1375, S1379, S1385, I112
`\@currentvline` I199,
 G177, G232, G249, G282, G298, J150
`\@currentx`
 780, S30, S44, S59, S72, S81, S119,
 S304, S307, S308, S323, S324, S335,
 S336, S438, S449, S460, S467, S481,
 S499, S528, S608, S617, S629, S791,

File Key: a=ltldirchk.dtx, b=ltplain.dtx, c=ltvers.dtx, d=ltluatex.dtx, e=ltexpl.dtx,
 f=ltdefns.dtx, g=ltcmd.dtx, h=lthooks.dtx, i=ltcmdhooks.dtx, j=ltalloc.dtx, k=ltcntrl.dtx,
 l=lterror.dtx, m=ltpar.dtx, n=ltpara.dtx, o=ltspace.dtx, p=ltlogos.dtx, q=ltfiles.dtx,
 r=ltoutenc.dtx, s=ltcounts.dtx, t=ltlength.dtx, u=ltfssbas.dtx, v=ltfssaxes.dtx,
 w=ltfssstrc.dtx, x=ltfsscmp.dtx, y=ltfssdcl.dtx, z=ltfssini.dtx, A=fontdef.dtx,
 B=preload.dtx, C=ltfntcmd.dtx, D=lttextcomp.dtx, E=ltpageno.dtx, F=ltxref.dtx,
 G=ltmiscen.dtx, H=ltmath.dtx, I=ltlists.dtx, J=ltboxes.dtx, K=lttab.dtx, L=ltpictur.dtx,
 M=ltthm.dtx, N=ltsect.dtx, O=ltfloat.dtx, P=ltidxglo.dtx, Q=ltbibl.dtx, R=ltpage.dtx,
 S=ltclass.dtx, T=ltfilehook.dtx, U=ltshipout.dtx, V=ltoutput.dtx, W=lthyphen.dtx,
 X=ltfinal.dtx

- S792, S793, S798, S804, S811, S816,
S818, S823, S829, S837, S843, S845,
S852, S855, S860, S863, S865, S866,
S868, S872, S881, S883, S884, S889,
S892, S895, S899, S905, S914, S919,
S920, S925, S931, S935, S937, S938,
S940, S942, S944, S945, S948, S954,
S956, S969, S982, S1005, S1021, S1022
- `\@currlist` O193, O215, O379, V67,
V311, V388, V391, V435, V438, V1812
- `\@currname`
196, 226–228, 778–780, c68, c104,
c112, h216, h222, h1178, q656, S29,
S43, S58, S71, S80, S118, S276,
S278, S290, S292, S304, S307, S323,
S335, S460, S528, S617, S629, S789,
S792, S793, S795, S796, S798, S804,
S811, S813, S816, S818, S823, S828,
S837, S841, S843, S852, S854, S860,
S863, S865, S866, S870, S874, S881,
S883, S884, S889, S892, S896, S900,
S905, S913, S937, S938, S940, S942,
S944, S945, S982, S1013, S1015,
S1022, S1032, S1565, S1587, S1617
- `\@currnamestack`
227, 228, 760, 823, h1175, S35, T507
- `\@curroptions`
S460, S468, S518, S537, S1022, S1023
- `\@currpath` . 760, 765, 778, 779, 781,
S15, S48, S133, S276, S278, S790,
S811, S818, S827, S852, S853, S877
- `\@currpkg@reqd` .. 780, S306, S821,
S823, S832, S848, S862, S877, S879
- `\@currsize` z492
- `\@currtype` V119, V878, V879, V880,
V881, V898, V899, V900, V901,
V1027, V1111, V1121, V1269,
V1280, V1422, V1509, V1627,
V1752, V2001, V2003, V2004, V2007
- `\@curtab` K11,
K26, K86, K87, K88, K94, K95,
K98, K102, K103, K107, K146, K147
- `\@curtabmar` K11, K25, K26,
K38, K44, K89, K102, K106, K107
- `\@d@r` a161, a162
- `\@dashbox` L324, L325,
L326, L327, L328, L331, L337,
L339, L349, L351, L352, L353,
L354, L358, L362, L365, L382,
L384, L385, L386, L387, L390,
L394, L396, L405, L407, L408,
L409, L410, L413, L417, L422, L674
- `\@dashcnt` L317, L319,
L320, L321, L322, L323, L336,
L338, L341, L343, L344, L345,
L347, L348, L361, L364, L376,
L377, L378, L379, L380, L381,
L393, L395, L398, L399, L400,
L401, L403, L404, L416, L421, L674
- `\@dashdim` L316,
L317, L318, L319, L320, L322,
L325, L327, L328, L329, L336,
L338, L340, L341, L342, L343,
L344, L347, L351, L353, L354,
L355, L363, L366, L375, L376,
L377, L378, L380, L384, L386,
L387, L388, L393, L395, L397,
L398, L399, L400, L403, L407,
L409, L410, L411, L419, L424, L674
- `\@date` N9, N33
- `\@dbflt` O32, O264
- `\@dblarg` f693, N54, N142, O12
- `\@dbldeferlist` O239, V70,
V445, V450, V452, V816, V823,
V824, V1752, V1795, V1926, V1954
- `\@dblfloat` O31
- `\@dblfloatplacement`
... q31, q100, q158, O280, V401,
V449, V1907, V1934, V2235, V2275
- `\@dblflset` O26
- `\@dblpfbot` O290, O304, V2319
- `\@dblpfsep` O289, O303, V2319
- `\@dblpftop` O288, O302, V2319
- `\@dbltoplist` ... V69, V232, V235,
V237, V397, V398, V445, V446,
V755, V759, V761, V762, V1639,
V1644, V1764, V1770, V1925, V1952
- `\@dbltopnum` ... O283, O297, V107,
V127, V238, V240, V766, V1578,
V1579, V1643, V1646, V1674,
V1679, V1699, V1700, V1769,
V1773, V1802, V1807, V1918, V1945
- `\@dbltoproom` O284,
O286, O298, O300, V108, V1581,
V1584, V1585, V1594, V1595,
V1598, V1601, V1604, V1608,
V1612, V1616, V1621, V1641,
V1702, V1705, V1706, V1715,
V1716, V1717, V1720, V1724,
V1727, V1731, V1736, V1740,
V1745, V1746, V1767, V1919, V1946
- `\@dec@text@cmd` r3
- `\@declarecommandcopylisthook` ...
..... 91, 92, f480, f482, f494
- `\@declaredoptions` S8,
S423, S464, S502, S523, S542, S975
- `\@declareoption` ... S421, S422, S430

File Key: a=ltldirchk.dtx, b=ltplain.dtx, c=ltvers.dtx, d=ltluatex.dtx, e=ltexpl.dtx,
f=ltdefs.dtx, g=ltcmd.dtx, h=lthooks.dtx, i=ltcmdhooks.dtx, j=ltalloc.dtx, k=ltcntrl.dtx,
l=lterror.dtx, m=ltpar.dtx, n=ltpara.dtx, o=ltspace.dtx, p=ltlogos.dtx, q=ltfiles.dtx,
r=ltoutenc.dtx, s=ltcounts.dtx, t=ltlength.dtx, u=ltfssbas.dtx, v=ltfssaxes.dtx,
w=ltfssstrc.dtx, x=ltfsscmp.dtx, y=ltfssdcl.dtx, z=ltfssini.dtx, A=fontdef.dtx,
B=preload.dtx, C=ltfntcmd.dtx, D=lttextcomp.dtx, E=ltpageno.dtx, F=ltxref.dtx,
G=ltmiscen.dtx, H=ltmath.dtx, I=ltlists.dtx, J=ltboxes.dtx, K=lttab.dtx, L=ltpicture.dtx,
M=ltthm.dtx, N=ltsect.dtx, O=ltfloat.dtx, P=ltidglo.dtx, Q=ltbibl.dtx, R=ltpage.dtx,
S=ltclass.dtx, T=ltfilehook.dtx, U=ltshipout.dtx, V=ltoutput.dtx, W=lthyphen.dtx,
X=ltfinal.dtx

\@defaultfamilyhook
 ... [z261](#), [z517](#), [z534](#), [z549](#), [z554](#), [z569](#)
 \@defaultsubs
 ... [u539](#), [u573](#), [u585](#), [G46](#), [G86](#)
 \@defaultunits . [u214](#), [u218](#), [u219](#),
 [u220](#), [u235](#), [u328](#), [w179](#), [w181](#), [L13](#)
 \@defaultunitsset [J53](#), [J64](#),
 [L8](#), [L29](#), [L30](#), [L32](#), [L34](#), [L60](#), [L63](#),
 [L84](#), [L85](#), [L107](#), [L108](#), [L164](#), [L241](#),
 [L316](#), [L318](#), [L332](#), [L340](#), [L342](#),
 [L357](#), [L480](#), [L481](#), [L612](#), [L648](#),
 [L690](#), [L691](#), [L693](#), [L694](#), [L697](#),
 [L698](#), [L700](#), [L701](#), [L712](#), [L713](#),
 [L715](#), [L716](#), [L718](#), [L719](#), [L721](#), [L722](#)
 \@defdefault@ds ... [S421](#), [S426](#), [S431](#)
 \@deferlist
 .. [V68](#), [V384](#), [V393](#), [V394](#), [V397](#),
 [V402](#), [V404](#), [V410](#), [V431](#), [V440](#),
 [V442](#), [V778](#), [V786](#), [V787](#), [V798](#),
 [V803](#), [V804](#), [V1111](#), [V1209](#), [V1269](#),
 [V1378](#), [V1422](#), [V1451](#), [V1509](#),
 [V1541](#), [V1627](#), [V1667](#), [V1924](#), [V1951](#)
 \@definecounter
 ... [s12](#), [s36](#), [H336](#), [M8](#), [M16](#),
 [O396](#), [O398](#), [I227](#), [I228](#), [I229](#), [I230](#)
 \@depth ... [f26](#), [w191](#), [A558](#), [A559](#),
 [A561](#), [A562](#), [J427](#), [J471](#), [K187](#),
 [K219](#), [L227](#), [L300](#), [L303](#), [L324](#),
 [L333](#), [L383](#), [L391](#), [L727](#), [L783](#), [V1851](#)
 \@dir [a160](#), [a163](#), [a165](#), [a167](#), [a168](#)
 \@disable@packageload@do . [S816](#), [T442](#)
 \@dischyp [f730](#), [f764](#), [J289](#), [J310](#)
 \@docclearpage [V296](#), [V371](#)
 \@documentclasshook
 ... [S3](#), [S701](#), [S729](#), [S756](#)
 \@doendpe [G214](#), [G221](#), [G273](#), [G288](#), [I123](#)
 \@dofilelist ... [q653](#), [q669](#), [G38](#), [G81](#)
 \@donoparitem [I144](#), [I158](#)
 \@dot [L605](#), [L643](#)
 \@dotsep [N200](#), [N223](#)
 \@dottedtocline .. [N185](#), [N211](#), [N212](#)
 \@downline [L297](#), [L301](#), [L306](#)
 \@downvector [L268](#), [L306](#)
 \@eha [c171](#), [f280](#), [f319](#), [f347](#),
 [I170](#), [I188](#), [I190](#), [I192](#), [I200](#), [I202](#),
 [I232](#), [q223](#), [q267](#), [q285](#), [r52](#), [r84](#),
 [u28](#), [u58](#), [u102](#), [u144](#), [u187](#), [u253](#),
 [u339](#), [w106](#), [y25](#), [y70](#), [y99](#), [y161](#),
 [y192](#), [y224](#), [y325](#), [y346](#), [y378](#), [y419](#),
 [y464](#), [y469](#), [y524](#), [y642](#), [y646](#), [y650](#),
 [y685](#), [y689](#), [y693](#), [y750](#), [y760](#), [y845](#),
 [y850](#), [y853](#), [y885](#), [y888](#), [y961](#), [y964](#),
 [y967](#), [y1034](#), [y1040](#), [C146](#), [D24](#),
 [D81](#), [D915](#), [D924](#), [G175](#), [G230](#),
 [G247](#), [G280](#), [Q51](#), [Q68](#), [V1876](#), [V1892](#)
 \@ehb [I170](#),
 [I195](#), [I220](#), [I222](#), [I224](#), [V234](#), [V390](#), [V437](#)
 \@ehc [f128](#), [f155](#), [f469](#), [I170](#),
 [I227](#), [I230](#), [I236](#), [I238](#), [G509](#), [G524](#),
 [G539](#), [G552](#), [H408](#), [N31](#), [S1617](#), [I220](#)
 \@ehd [c147](#),
 [e88](#), [I170](#), [I197](#), [I204](#), [I207](#), [I209](#),
 [I215](#), [y118](#), [K100](#), [K109](#), [O6](#), [S662](#), [S879](#)
 \@elt [485](#),
 [486](#), [495](#), [f52](#), [q376](#), [r1564](#), [r1566](#),
 [s20](#), [s35](#), [s53](#), [s56](#), [z76](#), [z87](#), [z89](#), [z90](#),
 [z115](#), [z358](#), [z360](#), [z361](#), [z371](#), [z595](#),
 [A17](#), [A25](#), [V8](#), [V11](#), [V15](#), [V27](#), [V30](#),
 [V31](#), [V32](#), [V33](#), [V38](#), [V39](#), [V40](#),
 [V41](#), [V42](#), [V43](#), [V44](#), [V45](#), [V47](#),
 [V51](#), [V57](#), [V58](#), [V59](#), [V60](#), [V498](#),
 [V720](#), [V731](#), [V736](#), [V746](#), [V758](#),
 [V760](#), [V788](#), [V805](#), [V825](#), [V844](#),
 [V857](#), [V864](#), [V915](#), [V918](#), [V927](#), [V1898](#)
 \@empty [68](#), [84](#),
 [377](#), [416](#), [483](#), [489](#), [504](#), [511](#), [762](#), [k14](#)
 \@emptycol
 ... [V198](#), [V245](#), [V248](#), [V277](#), [V281](#)
 \@end@check@IncludeInRelease ...
 ... [c131](#), [c133](#)
 \@end@tempboxa
 ... [J36](#), [J45](#), [J208](#), [J279](#), [J459](#), [J469](#)
 \@enddocument@kernel@warnings ...
 ... [G39](#), [G41](#), [G101](#)
 \@enddocumenthook ... [G70](#), [S977](#), [S999](#)
 \@endfloatbox [O190](#), [O211](#), [O236](#), [O248](#)
 \@endparenv [623](#), [I120](#), [I123](#)
 \@endparpenalty
 [615](#), [o15](#), [H437](#), [H449](#), [H475](#), [I23](#), [I124](#)
 \@endpbox
 [K193](#), [K236](#), [K266](#), [K333](#), [K384](#), [K387](#)
 \@endpfalse .. [G181](#), [G235](#), [G252](#),
 [G285](#), [J152](#), [I129](#), [I131](#), [I135](#), [I136](#), [I138](#)
 \@endpeltrue [I138](#)
 \@endpetrue [I124](#), [I126](#), [I134](#)
 \@endpreamblehook [303](#)
 \@endtheorem ... [M13](#), [M19](#), [M25](#), [M35](#)
 \@enlargepage .. [V1861](#), [V1866](#), [V1868](#)
 \@ensuredmath [H419](#), [H421](#)
 \@enumctr [I234](#), [I237](#), [I238](#)
 \@enumdepth . [626](#), [I226](#), [I232](#), [I233](#), [I234](#)
 \@enumspacing [626](#)
 \@eqcnt [H344](#),
 [H405](#), [H410](#), [H489](#), [H504](#), [H505](#), [H507](#)
 \@eqncr [H356](#), [H374](#), [H411](#), [H412](#), [H491](#)
 \@eqnnum [H338](#), [H339](#), [H409](#), [H423](#), [H482](#)
 \@eqnset [H344](#), [H503](#)

File Key: a=ltldirchk.dtx, b=ltplain.dtx, c=ltvers.dtx, d=ltluatex.dtx, e=ltexpl.dtx,
 f=ltdefns.dtx, g=ltcmd.dtx, h=lthooks.dtx, i=ltcmdhooks.dtx, j=ltalloc.dtx, k=ltcntrl.dtx,
 l=ltterror.dtx, m=ltpar.dtx, n=ltpara.dtx, o=ltspace.dtx, p=ltlogos.dtx, q=ltfiles.dtx,
 r=ltoutenc.dtx, s=ltcounts.dtx, t=ltlength.dtx, u=ltfssbas.dtx, v=ltfssaxes.dtx,
 w=ltfssstrc.dtx, x=ltfsscmp.dtx, y=ltfssdcl.dtx, z=ltfssini.dtx, A=fontdef.dtx,
 B=preload.dtx, C=ltfntcmd.dtx, D=lttextcomp.dtx, E=ltpageno.dtx, F=ltxref.dtx,
 G=ltmiscen.dtx, H=ltmath.dtx, I=ltlists.dtx, J=ltboxes.dtx, K=lttab.dtx, L=ltpicture.dtx,
 M=ltthm.dtx, N=ltsect.dtx, O=ltfloat.dtx, P=ltidxglo.dtx, Q=ltbibl.dtx, R=ltpage.dtx,
 S=ltclass.dtx, T=ltfilehook.dtx, U=ltshipout.dtx, V=ltoutput.dtx, W=lthyphen.dtx,
 X=ltfinal.dtx

\@eqnswfalse H373
 \@eqnswtrue .. H346, H352, H410, H488
 \@eqpen [H344](#), H377, H379, H390, H400
 \@err@ [137](#), [141](#), [144](#), [152](#), [164](#), [168](#), [171](#), [179](#)
 \@esphack [o38](#),
 [o131](#), [o345](#), [o362](#), [o379](#), [o396](#), [F35](#),
 [O385](#), [P17](#), [P19](#), [P34](#), [Q54](#), [Q70](#), [V1884](#)
 \@evenfoot [R12](#), [R15](#), [V613](#), [V672](#)
 \@evenhead [R12](#), [R15](#), [V612](#), [V671](#)
 \@execute@begin@hook
 [585](#), [G178](#), [G182](#), [G185](#)
 \@expandtwoargs [f217](#),
 [S228](#), [S275](#), [S466](#), [S502](#), [S556](#), [S565](#)
 \@expast [K239](#), [K267](#)
 \@expl@@@filehook@clear@replacement@flag@@
 [S374](#), [S395](#), [T286](#), [T307](#), [T480](#)
 \@expl@@@filehook@drop@extension@@N
 [816](#), [T283](#), [T284](#), [T304](#), [T305](#), [T482](#)
 \@expl@@@filehook@file@pop@@
 [S834](#), [T157](#), [T486](#), [T496](#)
 \@expl@@@filehook@file@pop@assign@@nnnn
 [813](#), [T162](#), [T488](#)
 \@expl@@@filehook@file@push@@
 [S817](#), [T151](#), [T484](#)
 \@expl@@@filehook@if@file@replaced@@TF
 [816](#), [817](#), [T280](#), [T301](#), [T478](#)
 \@expl@@@filehook@if@no@extension@@nTF
 [T276](#), [T297](#), [T468](#), [T470](#), [T495](#)
 \@expl@@@filehook@normalize@file@name@@w
 [T282](#), [T303](#), [T476](#)
 \@expl@@@filehook@resolve@file@subst@@w
 [S372](#), [S393](#), [T279](#), [T300](#), [T474](#)
 \@expl@@@filehook@set@curr@file@@nNN
 [778](#), [815](#), [817](#),
 [S371](#), [S392](#), [S783](#), [T339](#), [T345](#), [T472](#)
 \@expl@@@hook@curr@name@pop@@
 [h1260](#), [S91](#)
 \@expl@@@initialize@all@@ [h1260](#), [G186](#)
 \@expl@@@shipout@add@background@box@@n
 [U408](#), [U493](#)
 \@expl@@@shipout@add@background@picture@@n
 [U408](#), [U497](#)
 \@expl@@@shipout@add@firstpage@material@@Nn
 [U408](#), [U490](#)
 \@expl@@@shipout@add@foreground@box@@n
 [U408](#), [U495](#)
 \@expl@@@shipout@add@foreground@picture@@n\@finalstrut
 [U408](#), [U499](#)
 \@expl@char@generate@@nn .. [e141](#), [f709](#)
 \@expl@cs@argument@spec@@N
 [e138](#), [e149](#), [f610](#)
 \@expl@cs@prefix@spec@@N
 [e137](#), [e148](#), [f607](#)
 \@expl@cs@replacement@spec@@N ...
 [e139](#), [e150](#), [f568](#), [f602](#), [f610](#)
 \@expl@cs@to@str@@N .. [e132](#), [e135](#),
 [e144](#), [e146](#), [f477](#), [f534](#), [f554](#), [f556](#),
 [f557](#), [f570](#), [f582](#), [f594](#), [f598](#), [f599](#), [f604](#)
 \@expl@finalise@setup@@
 [e30](#), [X254](#), [X255](#)
 \@expl@pop@filename@@
 [e26](#), [S90](#), [S94](#), [S100](#), [S111](#)
 \@expl@push@filename@@ .. [760](#), [e24](#),
 [S39](#), [S41](#), [S54](#), [S56](#), [S66](#), [S78](#), [S98](#), [S104](#)
 \@expl@push@filename@aux@@
 [228](#), [e25](#),
 [h1238](#), [S39](#), [S50](#), [S62](#), [S66](#), [S84](#), [S104](#)
 \@expl@str@if@eq@@nnTF
 [e136](#), [e147](#), [f441](#), [f443](#), [S275](#)
 \@expl@str@map@function@@NN
 [99](#), [e140](#), [e151](#), [f706](#)
 \@expl@sys@load@backend@@
 [e21](#), [e23](#), [q17](#)
 \@extra@page@added [G57](#), [U378](#)
 \@failedlist .. [V842](#), [V865](#), [V881](#),
 [V888](#), [V901](#), [V907](#), [V923](#), [V937](#), [V960](#)
 \@fcolmadefalse [V833](#)
 \@fcolmadetrue [V921](#)
 \@file-subst@{file} [814](#)
 \@filef@und
 [812](#), [q427](#), [q448](#), [q469](#), [q492](#), [q509](#),
 [q522](#), [q575](#), [T155](#), [T172](#), [T186](#), [T194](#)
 \@filehook@file@push [779](#)
 \@filehook@set@CurrentFile
 [q313](#), [S819](#), [T152](#), [T328](#)
 \@filelist
 [q63](#), [q131](#), [q186](#), [q643](#), [q644](#), [q655](#),
 [z602](#), [z612](#), [z622](#), [X265](#), [X615](#), [X631](#)
 \@fileswfalse
 [q199](#), [S1069](#), [S1070](#), [S1200](#), [S1201](#)
 \@fileswith@ptions [S420](#),
 [S526](#), [S545](#), [S687](#), [S688](#), [S692](#), [S694](#),
 [S722](#), [S723](#), [S749](#), [S750](#), [S777](#), [S966](#)
 \@fileswith@ptions
 [S682](#), [S683](#), [S685](#), [S689](#)
 \@fileswith@ptions
 [S598](#), [S605](#), [S613](#), [S680](#)
 \@fileswtrue
 [q5](#), [S1052](#), [S1055](#), [S1108](#),
 [S1112](#), [S1183](#), [S1186](#), [S1241](#), [S1245](#)
 \@finalstrut
 [J383](#), [J401](#), [J470](#), [K385](#), [O476](#), [O495](#)
 \@firstampfalse .. [K254](#), [K277](#), [K294](#)
 \@firstamptrue [K262](#)
 \@firstcolfirstmark
 [V2212](#), [V2213](#), [V2217](#)
 \@firstcoltopmark [V2210](#), [V2218](#)

File Key: a=ltldirchk.dtx, b=ltplain.dtx, c=ltvers.dtx, d=ltluatex.dtx, e=ltexpl.dtx,
 f=ltdefs.dtx, g=ltcmd.dtx, h=lthooks.dtx, i=ltcmdhooks.dtx, j=ltalloc.dtx, k=ltcntrl.dtx,
 l=lterror.dtx, m=ltpar.dtx, n=ltpara.dtx, o=ltspace.dtx, p=ltlogos.dtx, q=ltfiles.dtx,
 r=ltoutenc.dtx, s=ltcounts.dtx, t=ltlength.dtx, u=ltfssbas.dtx, v=ltfssaxes.dtx,
 w=ltfssstrc.dtx, x=ltfsscmp.dtx, y=ltfssdcl.dtx, z=ltfssini.dtx, A=fontdef.dtx,
 B=preload.dtx, C=ltfntcmd.dtx, D=lttextcomp.dtx, E=ltpageno.dtx, F=ltxref.dtx,
 G=ltmiscen.dtx, H=ltmath.dtx, I=ltlists.dtx, J=ltboxes.dtx, K=lttab.dtx, L=ltpictur.dtx,
 M=ltthm.dtx, N=ltsect.dtx, O=ltfloat.dtx, P=ltidxglo.dtx, Q=ltbibl.dtx, R=ltpage.dtx,
 S=ltclass.dtx, T=ltfilehook.dtx, U=ltshipout.dtx, V=ltoutput.dtx, W=lthyphen.dtx,
 X=ltfinal.dtx

`\@firstcolumnfalse` V2202, V2247
`\@firstcolumntrue` q28,
q97, q155, V98, V207, V2221, V2253
`\@firstofone` 91, 167, d12,
d100, d108, d166, e34, e61, e76,
f212, f465, f470, f471, f474, f475,
q125, q180, q395, r68, r153, w346,
y53, y81, y142, y173, y204, y795,
G69, H417, K372, O10, Q18, Q46,
Q63, S808, S835, T319, T334, X345
`\@firstoftwo` 312, 315, a87, c82, e35, e43, f212,
f442, f444, f466, f537, f587, f645,
f655, f665, f692, f716, q449, q493,
q510, r133, s157, s162, y799, z363,
D69, D836, D887, D903, F19, R16,
S157, S196, S208, S235, S253, S1626
`\@firsttab` K2, K74, K75, K76, K106, K118
`\@flcheckspace` . . V989, V1025, V2102
`\@flfail` V865, V916, V937, V947, V960, V969
`\@float` O26, O32
`\@floatboxreset` . . O101, O170, O174
`\@floatpenalty` O3, O53, O55, O58, O122, O124,
O127, O191, O194, O199, O201,
O212, O216, O221, O223, O237,
O241, O311, O313, O317, O321, O379
`\@floatplacement` q31, q100, q158, O271, V149,
V209, V253, V477, V1908, V1935
`\@flsetnum` V986, V1022, V1109, V1267,
V1420, V1507, V1578, V1699, V2070
`\@flsettextmin` V1085, V1237, V1406, V1489, V2086
`\@flstop` V1972
`\@flsucceed` V858, V866, V915, V949, V971
`\@fltovf` l223, O93, O162, O322
`\@flupdates` V992, V1037, V2148
`\@flushglue` j17,
G359, G363, G369, G379, G382,
G387, G420, G442, J296, J317, I76
`\@fnsymbol` s107, s126
`\@font@aliasinfo` w539
`\@font@info` u133, u171,
u177, u366, u383, u621, v478, w30,
w38, w46, w74, w87, w154, w200,
w214, w225, w239, w255, w261,
w274, w281, w288, w293, w303,
w315, w327, w491, w503, w508,
w515, w545, w558, w566, y234,
y249, y283, y329, y398, y404, y448,
y461, y544, y633, y676, y741, y835,
y1002, y1031, D55, D56, D99, X464
`\@font@series@contextfalse` z355, z391
`\@font@series@contexttrue` z374, z378, z390
`\@font@shape@subst@warning` . 421,
v445, v448, v453, v511, v658, v661
`\@font@warning` u3, u535, u540, u567, u574, v456,
w19, w33, w41, w49, w61, w77,
w476, w490, w502, w507, w514,
w557, w565, x30, G43, G83, X286
`\@fontenc@load@list` r1565, z595, A17, A25
`\@fontswitch` C126, C128
`\@footnotemark` O454, O460, O504, O510, O511, O537
`\@footnotetext` J349, O454, O460, O461, O520, O526
`\@for` k16, q232, q303, q350,
q655, Q16, Q45, Q62, S237, S254,
S464, S483, S500, S518, S523, S537,
S542, S578, S588, S1023, S1060, S1191
`\@forced@seriesfalse` v394, v407, w140
`\@forced@seriestrue` v398, v409
`\@forloop` k19, k20
`\@fornoop` k15, k23, k29
`\@fortmp` k17, k18, k26,
S576, S578, S1059, S1060, S1190, S1191
`\@fpbot` O290, O304, V863, V2313
`\@fpmin` O278, O287, O301, V113,
V920, V1916, V1943, V2165, V2182
`\@fps` O41, O42,
O44, O47, O64, O110, O111, O113,
O116, O133, V1993, V1995, V1998
`\@fpsadddefault` O45, O48, O114, O117, V1990
`\@fpsep` O289,
O303, V861, V870, V942, V964, V2313
`\@fpstype` V983, V1004, V1005, V1019,
V1050, V1051, V1075, V1077,
V1080, V1082, V1133, V1189,
V1190, V1225, V1228, V1231,
V1234, V1295, V1357, V1358,
V1396, V1398, V1401, V1403,
V1477, V1480, V1483, V1486,
V1575, V1590, V1592, V1610,
V1619, V1655, V1656, V1696,
V1711, V1713, V1733, V1743,
V1782, V1783, V1986, V2002,
V2004, V2006, V2009, V2010,
V2011, V2013, V2014, V2018,

File Key: a=ltldirchk.dtx, b=ltplain.dtx, c=ltvers.dtx, d=ltluatex.dtx, e=ltexpl.dtx,
f=ltdefs.dtx, g=ltcmd.dtx, h=lthooks.dtx, i=ltcmdhooks.dtx, j=ltalloc.dtx, k=ltcntrl.dtx,
l=lterror.dtx, m=ltpar.dtx, n=ltpara.dtx, o=ltspace.dtx, p=ltlogos.dtx, q=ltfiles.dtx,
r=ltoutenc.dtx, s=ltcounts.dtx, t=ltlength.dtx, u=ltfssbas.dtx, v=ltfssaxes.dtx,
w=ltfssstrc.dtx, x=ltfsscmp.dtx, y=ltfssdcl.dtx, z=ltfssini.dtx, A=fontdef.dtx,
B=preload.dtx, C=ltfntcmd.dtx, D=lttextcomp.dtx, E=ltpageno.dtx, F=ltxref.dtx,
G=ltmiscen.dtx, H=ltmath.dtx, I=ltlists.dtx, J=ltboxes.dtx, K=lttab.dtx, L=lt pictur.dtx,
M=ltthm.dtx, N=ltsect.dtx, O=ltfloat.dtx, P=ltidxglo.dtx, Q=ltbibl.dtx, R=ltpage.dtx,
S=ltclass.dtx, T=ltfilehook.dtx, U=ltshipout.dtx, V=ltoutput.dtx, W=lthyphen.dtx,
X=ltfinal.dtx

- V2019, V2021, V2022, V2056,
V2058, V2060, V2072, V2074,
V2088, V2090, V2120, V2123, V2134
\@fptop O288, O302, V860, [V2313](#)
\@framebox J179, J207, [J209](#)
\@framebox J186, J193, [J197](#)
\@framepicbox J186, J193, [J230](#)
\@freelist b196, b213,
b275, O60, O129, O319, O320, V29,
V34, V48, V56, V213, V499, V732,
V747, V761, V866, V1812, V1813
\@getcirc
. [L437](#), [L485](#), [L514](#), [L541](#), [L613](#), [L629](#)
\@getfipsbit
. [V980](#), [V1016](#), [V1572](#), [V1693](#), [V2029](#)
\@getlarrow [L266](#), [L274](#), [L276](#)
\@getlinechar [L190](#), [L229](#)
\@getpen o34, o37, o46, [o117](#)
\@getrarrow [L267](#), [L274](#), [L283](#)
\@glossaryfile P21, P22, P31
\@gnewline o95, o101, o108, [o111](#)
\@gobble [495](#), [779](#), c151,
d11, d98, e33, e65, e88, f111, f133,
[f208](#), f219, f236, f240, f277, f283,
f286, f296, f316, f322, f325, f334,
f344, f350, f353, f362, f380, f384,
f386, f387, f389, f397, f401, f403, k6,
k9, l101, l127, l153, l162, o75, o483,
q64, q132, q187, q402, q404, q643,
r29, u536, u569, w345, x26, y28,
y30, y287, y298, y362, y409, y410,
y439, y445, y453, y458, y476, y490,
y500, y509, y522, y539, y548, y622,
y624, y628, y636, y670, y679, y731,
y733, y744, y828, y838, y919, y924,
y993, y1024, z74, z115, z371, z605,
z615, z625, D864, G239, N143,
N144, N145, N146, N147, N182,
O7, Q11, Q25, Q26, S650, S814,
S1044, S1104, S1131, S1235, S1264,
S1348, S1353, S1427, S1571, S1583,
T259, V619, V620, V621, V678,
V679, V680, V927, V1900, V2166,
V2183, X268, X397, X554, X563, X631
\@gobble@AddToHook@args h1276, h1277
\@gobble@IncludeInRelease [c68](#)
\@gobble@RemoveFromHook@arg
. h1279, h1280
\@gobblecr o481, o482
\@gobblefour [f208](#), y24,
y284, y400, y402, y406, y408, y418,
y422, y546, y598, S1133, S1266, S1355
\@gobblethree [f208](#), [f596](#), [f609](#)
\@gobbletwo
. e123, e128, f175, f176, [f208](#),
k12, q32, q101, q159, u541, u575,
y132, G23, G44, G76, G84, R11,
R13, S1103, S1234, S1347, T258, X292
\@gtempa
. f126, f127, f181, f183, f528, f536,
q587, q588, q594, q595, q596, q621,
q622, q624, q625, q626, K3, K5, K6,
K7, K8, S274, S276, S289, S290,
S309, S311, S325, S327, S337, S339
\@halfwidth
. [L2](#), [L126](#), [L129](#), [L131](#), [L227](#),
[L299](#), [L302](#), [L324](#), [L333](#), [L349](#),
[L361](#), [L364](#), [L383](#), [L391](#), [L405](#),
[L416](#), [L421](#), [L680](#), [L706](#), [L725](#),
[L726](#), [L727](#), [L766](#), [L781](#), [L782](#), [L783](#)
\@halignto K170, K174, K177, K191
\@hangfrom N66, N117, [N138](#)
\@height b425, [f26](#),
o350, o358, o384, o392, r293, r295,
w190, A340, A558, A559, A561,
A562, J163, J168, J216, J226, J427,
J471, K186, K219, K359, K376,
L227, L300, L303, L324, L333,
L351, L359, L383, L391, L407,
L414, L590, L600, L726, L782, V1851
\@highpenalty o118, [X3](#)
\@hightab [K11](#), K21, K23, K74,
K86, K95, K96, K111, K146, K147
\@hline [L167](#), [L179](#), [L226](#), [L265](#)
\@holdpg V122,
V300, V302, V303, V308, V309, V310
\@hspace o437, [o438](#), [o453](#)
\@hspacer o437, [o452](#)
\@hvector [L244](#), [L259](#), [L265](#)
\@icentercr [G337](#), [G338](#)
\@iden [f215](#)
\@if f171, f172, [f174](#)
\@if@DeclareRobustCommand
. 93, f483, f507,
f518, f519, [f523](#), f614, f615, f618, i106
\@if@newcommand
. 86, 93, 95, 238, f304, f484,
f508, f519, f561, f573, [f578](#), i107, i112
\@if@pti@ns
. S228, S232, S234, S251, S252, S268
\@if@ptions
. S223, S224, S227, S229, S793, S920
\@ifatmargin [K55](#), K106
\@ifbothcounters
. s61, s69, s71, s79, s81, s93
\@ifclasslater [S160](#), S170, S180
\@ifclassloaded [S149](#)

File Key: a=ltldirchk.dtx, b=ltplain.dtx, c=ltvers.dtx, d=ltluatex.dtx, e=ltexpl.dtx,
f=ltdefs.dtx, g=ltcmd.dtx, h=lthooks.dtx, i=ltcmdhooks.dtx, j=ltalloc.dtx, k=ltcntrl.dtx,
l=lterror.dtx, m=ltpar.dtx, n=ltpara.dtx, o=ltspace.dtx, p=ltlogos.dtx, q=ltfiles.dtx,
r=ltoutenc.dtx, s=ltcounts.dtx, t=ltlength.dtx, u=ltfssbas.dtx, v=ltfssaxes.dtx,
w=ltfssstrc.dtx, x=ltfsscmp.dtx, y=ltfssdcl.dtx, z=ltfssini.dtx, A=fontdef.dtx,
B=preload.dtx, C=ltfntcmd.dtx, D=lttextcomp.dtx, E=ltpageno.dtx, F=ltxref.dtx,
G=ltmiscen.dtx, H=ltmath.dtx, I=ltlists.dtx, J=ltboxes.dtx, K=lttab.dtx, L=ltpicture.dtx,
M=ltthm.dtx, N=ltsect.dtx, O=ltfloat.dtx, P=ltidxglo.dtx, Q=ltbibl.dtx, R=ltpage.dtx,
S=ltclass.dtx, T=ltfilehook.dtx, U=ltshipout.dtx, V=ltoutput.dtx, W=lthyphen.dtx,
X=ltfinal.dtx

<code>\@ifclasswith</code>	S223	<code>\@iiiminipage</code>	J329 , J333 , J336 , J337 , J338
<code>\@ifdefinable</code>		<code>\@iiiparbox</code>	J241 , J249 , J256 , J259 , J260 , J261 , J366
... f84 , f86 , f130 , f132 , f238 , r14 , r17 , s11 , t3 , z471 , J115 , M7 , M15 , M22		<code>\@iiminipage</code>	J332 , J334
<code>\@iffileonpath</code>	q423 , q444 , q465 , q482 , q503	<code>\@iininput</code>	315 , q535 , q536
<code>\@ifl@aded</code>	779 , S149 , S150 , S153 , S159 , S792 , S905 , S919	<code>\@iiparbox</code>	J255 , J257
<code>\@ifl@t@r</code> .	r1530 , S168 , S178 , S184 , S189 , S191 , S202 , S203 , S213 , S671	<code>\@iirsbox</code>	J451 , J460
<code>\@ifl@ter</code>	r1569 , r1570 , S160 , S161 , S183 , S837 , S948	<code>\@imakebox</code>	J26 , J41 , J138
<code>\@ifl@ter@@</code>	r1569 , r1570	<code>\@imakepicbox</code> ...	J47 , J48 , J143 , J233
<code>\@ifnch</code>	e49 , e50 , f677 , f679 , f691	<code>\@iminipage</code>	J328 , J330
<code>\@ifnextchar</code>	a98 , e45 , f673 , f678 , f692 , o93 , o482 , q535 , s13 , w411 , G336 , H342 , J9 , J11 , J18 , J20 , J26 , J47 , J121 , J122 , J128 , J129 , J136 , J140 , J185 , J186 , J192 , J193 , J198 , J231 , J239 , J247 , J254 , J258 , J327 , J331 , J335 , J411 , J416 , J439 , J446 , J451 , K57 , K181 , K203 , K210 , L23 , L132 , L143 , L453 , M3 , M5 , M28 , O27 , O264 , O324 , O452 , O501 , O518 , Q3 , Q13 , S281 , S295 , S666 , S681 , S686 , S1053 , S1056 , S1184 , S1187 , V209 , V1974 , I143	<code>\@include</code> ..	307 , q227 , q270 , q286 , q290
<code>\@iforloop</code>	k21 , k22	<code>\@includeinreleasefalse</code>	c71 , c76 , c121 , c129 , f768
<code>\@ifpackagelater</code> ...	S160 , S169 , S179	<code>\@includeinreleasetrue</code>	c111
<code>\@ifpackageloaded</code>	S149 , V1958	<code>\@index</code>	P18 , P19 , P35
<code>\@ifpackagewith</code>	S223	<code>\@indexfile</code>	P4 , P5 , P14
<code>\@iframebox</code>	J199 , J200 , J201	<code>\@inlabel</code>	614
<code>\@iframepicbox</code>	J231 , J232	<code>\@inlabelfalse</code>	V163 , V190 , I28 , I104 , I184
<code>\@ifstar</code>	f73 , f692 , o59 , o71 , o330 , o437 , s67 , s77 , u206 , x121 , G325 , G332 , G563 , G572 , H376 , K56 , K202 , K209 , L142 , L605 , N52 , N142 , S421 , S461 , V1856	<code>\@inlabeltrue</code>	I28 , I178
<code>\@ifundefin@d@i</code> .	f635 , f636 , f653 , f656	<code>\@inmatherr</code>	I233 , L605 , I112 , I142
<code>\@ifundefin@d@ii</code>	f635 , f638 , f641	<code>\@inmathwarn</code>	r3
<code>\@ifundefined</code> ...	f127 , f134 , f154 , f161 , f183 , f194 , f277 , f283 , f316 , f322 , f344 , f350 , f380 , f397 , f478 , f630 , g2269 , s3 , s7 , s16 , s50 , s62 , s64 , u100 , u186 , w424 , y319 , D54 , D866 , F23 , G141 , G158 , G174 , G229 , G246 , G279 , M21 , Q20 , Q48 , Q65 , R3 , R7 , S147 , S376 , S382 , S397 , S409 , S485 , S519 , S538 , T449	<code>\@inpenc@test</code>	X342 , X409
<code>\@ignorefalse</code>	G4 , G180 , G216 , G223 , G234 , G251 , G274 , G284 , G289 , O384	<code>\@input</code>	803 , q34 , q103 , q161 , q297 , q344 , q574 , N152 , X633
<code>\@ignoretrue</code>	o200 , o213 , G4 , G7 , H335 , H338 , H370 , H512	<code>\@input@</code> .	q317 , q359 , q576 , u393 , Q31
		<code>\@input@file@exists@with@hooks</code>	T143
		<code>\@inputcheck</code>	788 , a70 , a191 , a192 , a195 , a203 , b307 , e38 , e39 , e42 , f38 , f45 , q418 , q419 , q426 , q439 , q440 , q447 , q460 , q461 , q468 , q490 , q491 , q494 , q507 , q508 , q511 , S1092 , S1093 , S1127 , S1223 , S1224 , S1260 , S1335 , S1336 , S1343
		<code>\@insertfalse</code>	V1073 , V1223 , V1394 , V1475 , V1570 , V1691
		<code>\@inserttrue</code>	V999 , V1044 , V1161 , V1329 , V1649 , V1776
		<code>\@invalidchar</code>	I238
		<code>\@iparbox</code>	J240 , J248 , J253
		<code>\@irsbox</code>	J439 , J446 , J451 , J452
		<code>\@isavebox</code>	J136 , J137
		<code>\@isavepicbox</code>	J141 , J142
		<code>\@ishortstack</code>	L133 , L141
		<code>\@istackcr</code>	L143 , L144
		<code>\@itabcr</code>	K57 , K58
		<code>\@item</code>	I143 , I156
		<code>\@itemdepth</code> .	626 , I241 , I243 , I244 , I245
		<code>\@itemfudge</code>	K38 , K44 , K82
		<code>\@itemitem</code>	I245 , I248
		<code>\@itemlabel</code>	I44 , I96 , I143
		<code>\@itempenalty</code>	615 , o16 , I23 , I175
		<code>\@itemspacing</code>	626

File Key: a=ltldirchk.dtx, b=ltplain.dtx, c=ltvers.dtx, d=ltluatex.dtx, e=ltexpl.dtx, f=ltdefns.dtx, g=ltcmd.dtx, h=lt hooks.dtx, i=ltcmdhooks.dtx, j=ltalloc.dtx, k=ltcntrl.dtx, l=lterror.dtx, m=ltpar.dtx, n=lt para.dtx, o=lt space.dtx, p=ltlogos.dtx, q=ltfiles.dtx, r=ltoutenc.dtx, s=ltcounts.dtx, t=ltlength.dtx, u=ltfssbas.dtx, v=ltfssaxes.dtx, w=ltfssstrc.dtx, x=ltfsscmp.dtx, y=ltfssdcl.dtx, z=ltfssini.dtx, A=fontdef.dtx, B=preload.dtx, C=ltfntcmd.dtx, D=lttextcomp.dtx, E=ltpageno.dtx, F=ltxref.dtx, G=ltmiscen.dtx, H=ltmath.dtx, I=ltlists.dtx, J=ltboxes.dtx, K=lttab.dtx, L=lt pictur.dtx, M=ltthm.dtx, N=ltsect.dtx, O=ltfloat.dtx, P=ltidxglo.dtx, Q=ltbibl.dtx, R=ltpage.dtx, S=ltclass.dtx, T=ltfilehook.dtx, U=ltshipout.dtx, V=ltoutput.dtx, W=lthyphen.dtx, X=ltfinal.dtx

<code>\@iwhiledim</code>	k7	f278 , f317 , f345 , f469 , l136 , l168 ,
<code>\@iwhilenum</code>	k3	l184 , l190 , l192 , l195 , l197 , l199 ,
<code>\@iwhilesw</code>	k10	l202 , l204 , l206 , l209 , l213 , l218 ,
<code>\@ixpt</code>	u717	l222 , l224 , l226 , l227 , l229 , l232 ,
<code>\@ixstackcr</code>	L142	l236 , l238 , q223 , q267 , q285 , r50 ,
<code>\@kernel@...</code>	760	r84 , u5 , u28 , u58 , u102 , u144 , u187 ,
<code>\@kernel@after@{hook}</code>	180	u253 , u339 , w105 , x100 , x111 , y23 ,
<code>\@kernel@after@begindocument</code> ...	e7 , i41 , q58 , q75	y68 , y97 , y117 , y159 , y190 , y222 ,
<code>\@kernel@after@begindocument@before</code>	303 , q16 , q75 , v731	y245 , y261 , y325 , y346 , y378 , y418 ,
<code>\@kernel@after@enddocument</code>	e1 , G15 , U352	y422 , y464 , y469 , y524 , y592 , y598 ,
<code>\@kernel@after@enddocument@afterlastpage</code>	e1 , h1204 , G19 , U358	y642 , y646 , y650 , y685 , y689 , y693 ,
<code>\@kernel@after@para@after</code> 268 , n8 , n67		y750 , y760 , y845 , y850 , y853 , y885 ,
<code>\@kernel@after@para@end</code>	267 , 268 , n8 , n60	y888 , y961 , y964 , y967 , y1034 ,
<code>\@kernel@after@shipout@background</code>	834 , U72 , U159	y1040 , z46 , z57 , z453 , z573 , C143 ,
<code>\@kernel@after@shipout@lastpage</code> .	846 , U115 , U120 , U159 , U369 , U374	G175 , G230 , G247 , G280 , G509 ,
<code>\@kernel@before@{hook}</code>	180	G524 , G539 , G551 , H408 , K100 ,
<code>\@kernel@before@begindocument</code> ...	e7 , q56 , q75 , U398	K109 , N31 , O6 , O83 , Q51 , Q68 ,
<code>\@kernel@before@enddocument</code>	581 , G13 , G105	S609 , S644 , S657 , S794 , S878 , S921 ,
<code>\@kernel@before@para@before</code>	267 , 268 , n8 , n19	S1012 , S1029 , S1037 , S1042 , S1064 ,
<code>\@kernel@before@para@begin</code>	267 , 268 , n8 , n27	S1117 , S1195 , S1250 , S1586 , S1616 ,
<code>\@kernel@before@shipout@background</code>	834 , U68 , U70 , U159	V234 , V390 , V1874 , V1891 , I219
<code>\@kernel@currrpathstack</code>	S47 , S49 , S93 , S125	<code>\@latex@info</code>
<code>\@kernel@make@file@csname</code>	T278 , T281 , T299 , T302 , T328	f224 ,
<code>\@kernel@rename@newcommand</code> .. 86 ,		f298 , f336 , f364 , f730 , l136 , r85 , D76
<code>\@killglue</code> . L59 , L73 , L103 , L115 , L123		<code>\@latex@info@no@line</code> l136 , U89 , V576
<code>\@kludgeins</code>	V319 , V320 , V321 , V323 , V376 ,	<code>\@latex@warning</code> .. l136 , l166 , r55 ,
V377 , V423 , V424 , V502 , V518 ,		u33 , F14 , L449 , O260 , Q22 , Q49 ,
V519 , V525 , V526 , V527 , V536 ,		Q66 , S1156 , S1163 , S1289 , S1295 ,
V552 , V556 , V566 , V1852 , V1883		S1378 , S1384 , U79 , U98 , U173 , V1996
<code>\@labels</code>	614 , I27 , I146 , I147 , I189 , I206 , I207	<code>\@latex@warning@no@line</code> e98 ,
<code>\@largefloatcheck</code>	O192 , O213 , O238 , O256	f202 , l136 , l167 , q19 , q88 , q145 ,
<code>\@lastchclass</code>	K262 , K272 , K273 , K275 , K283 ,	q641 , F8 , F26 , F27 , G51 , G58 ,
K308 , K322 , K326 , K335 , K348 , K349		G91 , N32 , S277 , S291 , S672 , S838 ,
<code>\@latex@error</code>	c147 , c171 , e80 , f128 , f155 ,	S949 , S1094 , S1100 , S1121 , S1225 ,
		S1231 , S1254 , S1337 , S1344 , S1411 ,
		S1488 , V243 , V275 , V1827 , V2062
		<code>\@latexbug</code>
		l225 , V333 , V1813
		<code>\@latexerr</code>
		l166 , V437
		<code>\@latexrelease@catcode@null</code> g6 , g2270
		<code>\@lbibitem</code>
		Q3 , Q4
		<code>\@ldots</code>
		A504 , A506
		<code>\@leftcolumn</code>
		V121 , V2203 , V2224 , V2248 , V2257
		<code>\@leftmark</code>
		R16 , R50
		<code>\@let@token</code>
		e49 , e51 ,
		f677 , f680 , f683 , f691 , o410 , o411 ,
		o418 , C83 , C96 , H243 , H245 , H248
		<code>\@lign</code>
		H195 , H197
		<code>\@linechar</code>
		L190 , L191 , L192 ,
		L196 , L197 , L199 , L204 , L206 ,
		L207 , L208 , L209 , L211 , L215 ,
		L216 , L219 , L220 , L225 , L272 , L670
		<code>\@linefmt</code>
		L124 , L127 ,
		L190 , L265 , L273 , L304 , L307 , L677

File Key: a=ltldirchk.dtx, b=ltplain.dtx, c=ltvers.dtx, d=ltluatex.dtx, e=ltexpl.dtx, f=ltdefns.dtx, g=ltcmd.dtx, h=lt hooks.dtx, i=ltcmdhooks.dtx, j=ltalloc.dtx, k=ltcntrl.dtx, l=lterror.dtx, m=ltpar.dtx, n=lt para.dtx, o=lt space.dtx, p=ltlogos.dtx, q=ltfiles.dtx, r=ltoutenc.dtx, s=ltcounts.dtx, t=ltlength.dtx, u=ltfssbas.dtx, v=ltfssaxes.dtx, w=ltfssstrc.dtx, x=ltfsscmp.dtx, y=ltfssdcl.dtx, z=ltfssini.dtx, A=fontdef.dtx, B=preload.dtx, C=ltfntcmd.dtx, D=lttextcomp.dtx, E=ltpageno.dtx, F=ltxref.dtx, G=ltmiscen.dtx, H=ltmath.dtx, I=ltlists.dtx, J=ltboxes.dtx, K=lttab.dtx, L=lt pictur.dtx, M=ltthm.dtx, N=ltsect.dtx, O=ltfloat.dtx, P=ltidxglo.dtx, Q=ltbibl.dtx, R=lt page.dtx, S=ltclass.dtx, T=ltfilehook.dtx, U=ltshipout.dtx, V=ltoutput.dtx, W=lthyphen.dtx, X=ltfinal.dtx

- \@linelen . L164, L165, L176, L177,
L203, L210, L219, L221, L226,
L227, L228, L241, L242, L256,
L257, L300, L303, L305, L306, [L671](#)
- \@list [615](#)
- \@listctr [614](#), [Q9](#), [I202](#), [I225](#)
- \@listdepth
..... [614](#), [J350](#), [I23](#), [I35](#), [I38](#), [I43](#), [I99](#)
- \@listfiles q62, q130, q185, q647, q662
- \@listi [615](#)
- \@listii [615](#)
- \@listvi [615](#)
- \@loadwithoptions
..... [S615](#), [S621](#), [S631](#), [S640](#)
- \@lowpenalty [o117](#), [X3](#)
- \@ltab [K71](#), [K106](#)
- \@m [246](#), [b21](#), [b385](#), [b387](#),
[b388](#), [b421](#), [b422](#), [o288](#), [o429](#), [o434](#),
[q45](#), [q114](#), [q172](#), [L213](#), [L217](#), [Q17](#), [I80](#)
- \@mainaux [q3](#),
[q37](#), [q38](#), [q106](#), [q107](#), [q164](#), [q165](#),
[q216](#), [q297](#), [q336](#), [q344](#), [q369](#), [G22](#), [G75](#)
- \@makebox [J11](#), [J20](#), [J25](#)
- \@makecaption [O24](#)
- \@makecol [V261](#), [V413](#), [V460](#), [V480](#)
- \@makefcolumn [V393](#), [V394](#),
[V402](#), [V404](#), [V440](#), [V442](#), [V450](#),
[V452](#), [V2161](#), [V2163](#), [V2179](#), [V2180](#)
- \@makefnmark [O400](#), [O514](#)
- \@makefntext . [J382](#), [J400](#), [O475](#), [O494](#)
- \@makeother [a76](#), [a97](#), [a126](#),
[f695](#), [f696](#), [u406](#), [u407](#), [u408](#), [u409](#),
[u410](#), [u411](#), [u412](#), [u413](#), [u414](#), [u415](#),
[u416](#), [G431](#), [G452](#), [G545](#), [G560](#),
[G570](#), [S363](#), [S364](#), [S1138](#), [S1271](#), [S1360](#)
- \@makepicbox [J10](#), [J19](#), [J46](#), [L366](#), [L424](#)
- \@makespecialcolbox [V503](#), [V522](#)
- \@marbox [O320](#), [O322](#),
[O326](#), [O330](#), [O331](#), [O379](#), [V1812](#),
[V1822](#), [V1825](#), [V1833](#), [V1835](#),
[V1836](#), [V1838](#), [V1839](#), [V1840](#), [V1849](#)
- \@marginparreset .. [O343](#), [O361](#), [O370](#)
- \@markright [R33](#), [R48](#)
- \@maxdepth [q60](#),
[q128](#), [q183](#), [V91](#), [V486](#), [V514](#), [X138](#)
- \@maxtab [K2](#), [K94](#)
- \@medpenalty [o118](#), [X3](#)
- \@meta@family@list [z76](#), [z88](#), [z178](#), [z359](#)
- \@midlist . [V66](#), [V499](#), [V500](#), [V1027](#),
[V1029](#), [V1141](#), [V1305](#), [V1923](#), [V1950](#)
- \@minipagefalse [J323](#), [J325](#),
[J363](#), [O187](#), [O250](#), [O345](#), [O363](#), [I181](#)
- \@minipagerestore [J351](#), [J353](#)
- \@minipagetrue [J324](#), [O186](#)
- \@minus [f26](#),
[V2306](#), [V2307](#), [V2308](#), [V2311](#), [V2312](#)
- \@missing@onefilewithoptions ...
..... [S812](#), [S851](#), [S962](#)
- \@missingfile@area
..... [q548](#), [q589](#), [q602](#), [S853](#)
- \@missingfile@base
..... [q548](#), [q590](#), [q603](#), [S854](#)
- \@missingfile@ext
..... [q548](#), [q591](#), [q604](#), [S855](#)
- \@missingfileerror [315](#), [316](#),
[779](#), [q543](#), [q558](#), [q568](#), [q577](#), [S852](#), [S942](#)
- \@mkboth [R11](#), [R13](#)
- \@mklab [I45](#), [I140](#)
- \@mkpream [K189](#), [K234](#), [K262](#)
- \@mparbottom
..... [O387](#), [O388](#), [V118](#), [V476](#),
[V1823](#), [V1831](#), [V1832](#), [V1833](#), [V1834](#)
- \@mpargs [J342](#), [J366](#)
- \@mparswitchfalse [V102](#)
- \@mpfn . [J348](#), [O452](#), [O457](#), [O523](#), [O527](#)
- \@mpfootins [J357](#), [J358](#),
[J361](#), [J367](#), [J374](#), [J375](#), [J392](#), [J393](#)
- \@mpfootnotetext [J349](#), [J369](#)
- \@mplistdepth [J350](#), [J367](#)
- \@multicnt [K370](#), [K372](#),
[K373](#), [K374](#), [K381](#), [K382](#), [K383](#),
[L103](#), [L104](#), [L106](#), [L115](#), [L116](#),
[L118](#), [L667](#), [L704](#), [L706](#), [L707](#),
[L708](#), [L710](#), [L711](#), [L717](#), [L723](#),
[L734](#), [L738](#), [L764](#), [L766](#), [L768](#),
[L770](#), [L771](#), [L775](#), [L779](#), [L790](#), [L794](#)
- \@multiplelabels [q33](#), [q102](#),
[q160](#), [F25](#), [F31](#), [G49](#), [G55](#), [G89](#), [G95](#)
- \@multiput [L86](#), [L95](#), [L98](#)
- \@multispan [K371](#), [K375](#), [K379](#)
- \@namedef [f50](#), [q382](#), [u135](#), [u136](#), [u160](#),
[v5](#), [v525](#), [w418](#), [D57](#), [D93](#), [D869](#),
[F28](#), [G208](#), [G218](#), [G245](#), [G271](#),
[G483](#), [G492](#), [H412](#), [H413](#), [K175](#),
[M12](#), [M13](#), [M18](#), [M19](#), [M23](#), [M24](#),
[M25](#), [S1055](#), [S1186](#), [T447](#), [X466](#), [X467](#)
- \@nameuse [f51](#), [q334](#), [q367](#),
[q381](#), [M23](#), [R5](#), [T450](#), [V607](#), [V665](#)
- \@nbitem [I168](#), [I221](#)
- \@ne [246](#), [b16](#)
- \@needsf@rmat [S667](#), [S670](#), [S675](#)
- \@needsformat [S655](#), [S665](#), [S669](#)
- \@negargfalse [L186](#)
- \@negargtrue [L185](#)
- \@newcommand [f79](#), [f80](#)
- \@newctr [s13](#), [s15](#), [M8](#)
- \@newenv [f150](#), [f151](#), [f160](#)
- \@newenva [f148](#), [f149](#)

File Key: a=ltldirchk.dtx, b=ltplain.dtx, c=ltvers.dtx, d=ltluatex.dtx, e=ltexpl.dtx, f=ltdefns.dtx, g=ltcmd.dtx, h=lt hooks.dtx, i=ltcmdhooks.dtx, j=ltalloc.dtx, k=ltcntrl.dtx, l=lterror.dtx, m=ltpar.dtx, n=lt para.dtx, o=lt space.dtx, p=ltlogos.dtx, q=ltfiles.dtx, r=ltoutenc.dtx, s=ltcounts.dtx, t=ltlength.dtx, u=ltfssbas.dtx, v=ltfssaxes.dtx, w=ltfssstrc.dtx, x=ltfsscmp.dtx, y=ltfssdcl.dtx, z=ltfssini.dtx, A=fontdef.dtx, B=preload.dtx, C=ltfntcmd.dtx, D=lttextcomp.dtx, E=ltpageno.dtx, F=ltxref.dtx, G=ltmiscen.dtx, H=ltmath.dtx, I=ltlists.dtx, J=ltboxes.dtx, K=lttab.dtx, L=lt pictur.dtx, M=ltthm.dtx, N=ltsect.dtx, O=ltfloat.dtx, P=ltidxglo.dtx, Q=ltbibl.dtx, R=ltpage.dtx, S=ltclass.dtx, T=ltfilehook.dtx, U=ltshipout.dtx, V=ltoutput.dtx, W=lthyphen.dtx, X=ltfinal.dtx

<code>\@newenvb</code>	f150 , f151	<code>\@noitemargtrue</code>	I32 , I143
<code>\@newl@bel</code>	F22 , G24 , G77 , Q10	<code>\@noitemerr</code>	I228 ,
<code>\@newline</code>	o94 , o96		o248 , o266 , o303 , o326 , I69 , I81 , I107
<code>\@newlistfalse</code>		<code>\@noligs</code>	G432 , G453 , G561 , G571 , G582
	V600 , V658 , I29 , I33 , I108 , I182	<code>\@nolnerr</code>	I189 , o42 , o113 , G324 , G331
<code>\@newlisttrue</code>	I29 , I33 , I87	<code>\@nomath</code>	u1 ,
<code>\@next</code>	b275 , O60 , O129 , O319 , O320 ,		u337 , z404 , z439 , z445 , z466 , z468 , z490
	V9 , V213 , V311 , V877 , V897 , V1812	<code>\@nparitemfalse</code>	I30 , I145
<code>\@nextchar</code>		<code>\@nparitemtrue</code>	I30 , I66
	K269 , K270 , K330 , K331 , K332	<code>\@nparlistfalse</code>	I31 , I70
<code>\@nil</code>	a161 , a162 , c13 , c19 , c80 , c92 ,	<code>\@nparlisttrue</code>	I31 , I67
	c93 , c105 , c106 , c153 , c154 , c155 ,	<code>\@normalcr</code>	o52 , o92 , J322
	f53 , f54 , f58 , f63 , f135 , f439 , f440 ,	<code>\@normalsize</code>	S4 , S5
	f585 , f671 , f672 , k13 , k19 , k27 , p14 ,	<code>\@noskipsec</code>	614
	q246 , q247 , q248 , r89 , r110 , r996 ,	<code>\@noskipsecfalse</code>	
	r1000 , r1054 , r1066 , r1068 , u358 ,		q54 , q123 , q178 , N98 , V158 , V185
	u369 , u485 , u500 , u604 , u607 , u608 ,	<code>\@noskipsectrue</code>	N38 , N95
	u616 , v431 , v433 , v465 , v467 , v645 ,	<code>\@notdefinable</code>	
	v647 , v671 , v673 , w350 , w351 ,		f136 , f137 , f141 , f466 , I183
	w353 , w366 , w372 , w376 , w377 ,	<code>\@notprerr</code>	I231 , q66 , q134 , q189
	w413 , w434 , w439 , w519 , w533 ,	<code>\@nthm</code>	M3 , M4
	x26 , x44 , x53 , x57 , y40 , y388 ,	<code>\@nxxtabmar</code>	K11 , K21 , K23 ,
	y396 , y429 , y1045 , y1047 , C58 ,		K25 , K75 , K111 , K112 , K118 , K119
	C62 , K367 , K368 , S92 , S93 , S99 ,	<code>\@obsoletefile</code>	q640
	S107 , S110 , S117 , S134 , S192 , S193 ,	<code>\@oddfont</code>	
	S204 , S205 , S216 , S217 , S219 , S372 ,		R11 , R14 , R15 , V124 , V610 , V669
	S393 , S436 , S441 , S557 , S577 , S581 ,	<code>\@oddfont</code>	R11 , R14 , V123 , V610 , V669
	S587 , S591 , S763 , S772 , S786 , S787 ,	<code>\@onefilewithoptions</code>	
	S1461 , S1466 , S1469 , S1471 , S1472 ,		228 , 809 , S700 , S704 ,
	S1487 , S1507 , S1524 , S1578 , S1600 ,		S710 , S728 , S732 , S738 , S755 , S759 ,
	S1624 , T168 , T176 , T334 , T376 , T378		S765 , S781 , S782 , S849 , S909 , S1437
<code>\@nmbrlistfalse</code>	I33 , I46 , I91	<code>\@onelevel@sanitize</code>	f697 , O42 , O111
<code>\@nmbrlisttrue</code>	I225	<code>\@onfilewithoptions</code>	782
<code>\@nnil</code>	f417 , f423 ,	<code>\@onlypreamble</code>	f66 ,
	k13 , k20 , k21 , k22 , k28 , q247 , q248 ,		f188 , f190 , f199 , f207 , h1254 , q196 ,
	u214 , u218 , u219 , u220 , u235 , w179 ,		q205 , q242 , q642 , q668 , r23 , r24 ,
	w181 , w345 , w347 , w359 , w361 ,		r61 , r62 , r66 , r125 , r145 , r189 , r190 ,
	w366 , w380 , w382 , w389 , w400 ,		r204 , u17 , u115 , u117 , u123 , u139 ,
	w401 , w403 , w434 , w439 , L13 ,		u167 , u182 , u203 , u208 , u250 , u512 ,
	S708 , S709 , S716 , S736 , S737 , S744		w419 , x28 , x36 , x42 , x79 , x83 , x88 ,
<code>\@no@font@optfalse</code>	x17 , x129		x93 , x98 , x108 , x126 , x127 , x128 ,
<code>\@no@lnbk</code>	o9 , o10 , o40		x134 , x138 , x142 , y17 , y19 , y44 ,
<code>\@no@pgbk</code>	o7 , o8 , o32		y46 , y107 , y116 , y136 , y275 , y276 ,
<code>\@nobreakfalse</code>			y279 , y311 , y349 , y351 , y353 , y366 ,
	o120 , o122 , N94 , N129 , N157 ,		y381 , y428 , y430 , y472 , y511 , y527 ,
	O182 , V165 , V192 , V1150 , V1316 , I193		y604 , y698 , y707 , y763 , y766 , y769 ,
<code>\@nobreaktrue</code>	o121 , N126 , O181		y789 , y802 , y856 , y891 , y902 , y916 ,
<code>\@nocnterr</code>	I191		y970 , y990 , y994 , y1058 , C140 ,
<code>\@nocounterr</code>			C141 , D58 , D59 , D60 , D870 , F30 ,
	I191 , s4 , s8 , s16 , s62 , s64 , M21		P12 , P29 , Q44 , Q61 , S10 , S13 , S33 ,
<code>\@nodocument</code>			S34 , S87 , S114 , S122 , S124 , S148 ,
	I196 , n104 , q68 , q136 , q191 ,		S151 , S152 , S159 , S162 , S163 , S171 ,
	G166 , O39 , O108 , V156 , V183 , V212		S172 , S173 , S186 , S213 , S225 , S226 ,
<code>\@noitemargfalse</code>	I32 , I200		S229 , S268 , S282 , S345 , S351 , S414 ,

File Key: a=ltldirchk.dtx, b=ltplain.dtx, c=ltvers.dtx, d=ltluatex.dtx, e=ltexpl.dtx, f=ltdefns.dtx, g=ltcmd.dtx, h=lt hooks.dtx, i=ltcmdhooks.dtx, j=ltalloc.dtx, k=ltcntrl.dtx, l=lterror.dtx, m=ltpar.dtx, n=lt para.dtx, o=lt space.dtx, p=lt logos.dtx, q=lt files.dtx, r=ltoutenc.dtx, s=lt counts.dtx, t=lt length.dtx, u=ltfssbas.dtx, v=ltfssaxes.dtx, w=ltfssstrc.dtx, x=ltfsscmp.dtx, y=ltfssdcl.dtx, z=ltfssini.dtx, A=fontdef.dtx, B=preload.dtx, C=ltfntcmd.dtx, D=lttextcomp.dtx, E=ltpageno.dtx, F=ltxref.dtx, G=ltmiscen.dtx, H=ltmath.dtx, I=ltlists.dtx, J=ltboxes.dtx, K=lttab.dtx, L=lt pictur.dtx, M=ltthm.dtx, N=ltsect.dtx, O=ltfloat.dtx, P=ltidxglo.dtx, Q=ltbibl.dtx, R=lt page.dtx, S=ltclass.dtx, T=ltfilehook.dtx, U=ltshipout.dtx, V=ltoutput.dtx, W=lthyphen.dtx, X=ltfinal.dtx

- S417, S418, S429, S430, S431, S456,
S462, S475, S512, S530, S550, S570,
S594, S599, S603, S606, S614, S619,
S622, S632, S651, S664, S669, S675,
S684, S689, S777, S849, S967, S976,
S984, S985, S1003, S1010, S1019,
S1026, S1027, S1035, S1040, S1045,
S1417, S1418, S1419, S1420, S1422
`\@opargbegintheorem` M32, M35
`\@opcol` V262,
V270, V394, V413, V442, V460, V465
`\@options` S549
`\@othm` M3, M20
`\@outerparskip` I8, I88, I117, I152, I222
`\@outputbox` V120, V483,
V485, V505, V508, V509, V529,
V531, V532, V537, V540, V545,
V547, V554, V560, V562, V636,
V695, V723, V729, V739, V740,
V763, V770, V856, V859, V862,
V868, V869, V2203, V2207, V2208,
V2222, V2228, V2248, V2254, V2263
`\@outputdblcol`
. V468, V2198, V2200, V2244, V2245
`\@outputpage` . . V403, V452, V470,
V590, V2232, V2237, V2270, V2278
`\@oval` L453, L471
`\@ovbtrue` L476, L504, L534
`\@ovdx` L431, L487,
L489, L495, L497, L516, L518,
L526, L528, L543, L551, L553,
L589, L592, L599, L601, L693,
L694, L695, L696, L712, L713,
L714, L717, L733, L753, L754,
L755, L756, L772, L773, L775, L789
`\@ovdy` L431, L488,
L490, L496, L497, L517, L519,
L527, L528, L544, L552, L553,
L564, L569, L577, L581, L700,
L701, L702, L703, L718, L719,
L720, L723, L737, L760, L761,
L762, L763, L776, L777, L779, L793
`\@ovhlinefalse` L477, L505
`\@ovhlinetrue` L456,
L460, L465, L483, L489, L511, L518
`\@ovhorz` L494,
L495, L525, L526, L550, L551, L584
`\@ovltrue` L476, L504, L534
`\@ovri` J33, L431, L486,
L515, L542, L564, L577, L593, L602
`\@ovro` . . . L431, L486, L495, L496,
L515, L526, L527, L542, L551,
L552, L563, L569, L576, L581,
L588, L598, L614, L621, L630, L639
`\@ovrtrue` L476, L504, L534
`\@ovttrue` L476, L504, L534
`\@ovvert` L492,
L493, L521, L523, L546, L548, L557
`\@ovvlinefalse` L477, L505
`\@ovvlinetrue`
. L459, L482, L490, L510, L519
`\@ovxx` L431, L480, L482,
L483, L487, L493, L494, L508,
L510, L511, L516, L523, L525,
L537, L539, L543, L548, L550,
L588, L598, L690, L691, L692,
L696, L705, L706, L715, L716,
L717, L732, L750, L751, L752,
L756, L765, L766, L774, L775, L788
`\@ovyy` L431, L481, L482,
L483, L488, L495, L509, L510,
L511, L517, L526, L538, L539,
L544, L551, L561, L574, L697,
L698, L699, L703, L705, L721,
L722, L723, L736, L757, L758,
L759, L763, L765, L778, L779, L792
`\@p@pfilename`
. . . S92, S99, S107, S110, S117, S122
`\@p@pfilepath` S93, S132, S141
`\@p@pfilepath@aux` . . S133, S134, S142
`\@pagedp` V117, V308,
V313, V1091, V1244, V1841, V1851
`\@pageht` V116,
V309, V313, V315, V316, V317,
V321, V1090, V1243, V1824, V1831
`\@par` m3, m5
`\@parboxrestore`
. J266, J322, J347, J378,
J396, O19, O100, O169, O342,
O360, O470, O489, V219, V601, V659
`\@parboxto` J261
`\@parmoderr` . . . I221, O58, O127, O316
`\@parse@version`
. . . c80, c92, c93, c105, c106, c153,
c154, c155, S198, S204, S205, S216,
S1472, S1487, S1524, S1600, S1624
`\@parse@version@` S192, S193, S198, S210
`\@parse@version@dash` . . . S217, S219
`\@partaux` q3, q222, q266,
q284, q307, q309, q310, q330, q354,
q356, q357, q363, q375, q377, q380
`\@partlist` q231, q235,
q236, q238, q262, q281, q303, q350
`\@partswfalse` q6
`\@partswtrue` q230, q260, q280
`\@pass@options` S368,
S370, S388, S389, S391, S404, S405,
S407, S414, S415, S416, S860, S935

File Key: a=ltldirchk.dtx, b=ltplain.dtx, c=ltvers.dtx, d=ltluatex.dtx, e=ltexpl.dtx,
f=ltdefns.dtx, g=ltcmd.dtx, h=lt hooks.dtx, i=ltcmdhooks.dtx, j=ltalloc.dtx, k=ltcntrl.dtx,
l=lterror.dtx, m=ltpar.dtx, n=ltpara.dtx, o=lt space.dtx, p=ltlogos.dtx, q=ltfiles.dtx,
r=ltoutenc.dtx, s=ltcounts.dtx, t=ltlength.dtx, u=ltfssbas.dtx, v=ltfssaxes.dtx,
w=ltfssstrc.dtx, x=ltfsscmp.dtx, y=ltfssdcl.dtx, z=ltfssini.dtx, A=fontdef.dtx,
B=preload.dtx, C=ltfntcmd.dtx, D=lttextcomp.dtx, E=ltpageno.dtx, F=ltxref.dtx,
G=ltmiscen.dtx, H=ltmath.dtx, I=ltlists.dtx, J=ltboxes.dtx, K=lttab.dtx, L=lt pictur.dtx,
M=ltthm.dtx, N=ltsect.dtx, O=ltfloat.dtx, P=ltidnglo.dtx, Q=ltbibl.dtx, R=lt page.dtx,
S=ltclass.dtx, T=ltfilehook.dtx, U=ltshipout.dtx, V=ltoutput.dtx, W=lthyphen.dtx,
X=ltfinal.dtx

<code>\@pboxswfalse</code>	J264, J340	<code>\@removefromreset</code> s46 , s69 , s72 , s91 , s92	
<code>\@pboxswtrue</code>	J274	<code>\@reqcolroom</code>	V1090,
<code>\@penup</code>	H186, H187		V1091, V1094, V1096, V1097,
<code>\@percentchar</code>			V1102, V1106, V1108, V1136,
.	a106 , S1130 , S1132 , S1134 ,		V1137, V1243, V1244, V1248,
	S1136 , S1263 , S1265 , S1267 , S1269 ,		V1251, V1252, V1257, V1264,
	S1352 , S1354 , S1356 , S1358 , S1406		V1266, V1298, V1299, V1410,
<code>\@picbox</code>	L6 , L31 , L43 , L51 , L52		V1412, V1414, V1417, V1419,
<code>\@picht</code>	L6 , L29 , L42 , L51		V1493, V1496, V1499, V1504,
<code>\@picture</code>	L23 , L24		V1506 , V1986 , V2103 , V2108 , V2111
<code>\@picture@warn</code> L223 , L441 , L445 , L449		<code>\@reserveda</code>	414
<code>\@pkgextension</code>		<code>\@reset@ptions</code>	
781 , S31 , S149 , S160 , S223 , S415 ,		. . . S810 , S847 , S858 , S916 , S958 , S968	
S605 , S608 , S631 , S640 , S711 , S739 ,		<code>\@resetactivechars</code> V575 , V598 , V656	
S766 , S868 , S895 , S1021 , T447 , T453		<code>\@resethfps</code>	V1205 , V1374 , V2053
<code>\@plus</code>	f26 , o456 , N16 ,	<code>\@restorepar</code>	
N191 , N214 , R54 , V2306 , V2307 ,		m5 , o341 , o357 , o375 , o391 , I127 , I135	
V2308 , V2311 , V2312 , V2316 ,		<code>\@reversemarginfalse</code>	O388 , V101
V2317 , V2318 , V2322 , V2323 , V2324		<code>\@reversemargintrue</code>	O387
<code>\@pnumwidth</code>	N203 , N226	<code>\@rightmark</code>	R16 , R51
<code>\@popfilename</code>		<code>\@rightskip</code>	
.	760 , 761 , 779 , S35 , S846 , S957	. . . G363 , G382 , G400 , J295 , J316 , I75	
<code>\@pr@videpackage</code>	S281 , S295 ,	<code>\@rjfieldfalse</code>	K34 , K77
S298 , S319 , S321 , S332 , S334 , S345		<code>\@rjfieldtrue</code>	K125
<code>\@preamble</code>	K190 ,	<code>\@rmfamilyhook</code>	z261 , z304 , z316 , z343
K192 , K200 , K237 , K256 , K258 ,		<code>\@roman</code>	s103 , s109
K259 , K263 , K278 , K296 , K297 , K334		<code>\@rsbox</code>	J439 , J446 , J450
<code>\@preamblecmds</code>		<code>\@rtab</code>	K71 , K86
748 , f66 , q67 , q135 , q190 , S1638 , S1639		<code>\@rule</code>	J411 , J416 , J420
<code>\@preamerr</code>	l210 , K199 , K274 , K355	<code>\@sanitize</code>	f695 , P7 , P18 , P24 , P35
<code>\@process@ptions</code>	S474 , S492 ,	<code>\@savebox</code>	J122 , J129 , J135
S509 , S516 , S517 , S530 , S534 , S536		<code>\@savemarbox</code> O326 , O327 , O330 , O333	
<code>\@process@ptions</code>	S461 , S463 , S475	<code>\@savepicbox</code>	J122 , J129 , J139
<code>\@protected@tstopt</code> f89 , f101 , f580 , f592		<code>\@savsf</code>	o123 , o129 ,
<code>\@providesfile</code>	a98 , a99 , S355 , X627	o138 , o157 , o171 , o183 , o197 , o211	
<code>\@optionlist</code>	S146 , S228 ,	<code>\@savsk</code>	o123 , o128 ,
S460 , S798 , S804 , S925 , S931 , S1022		o139 , o158 , o172 , o184 , o198 , o212	
<code>\@pushfilename</code> 228 , 760 , S35 , S788 , S912		<code>\@scolelt</code>	V788 , V853
<code>\@put</code> L452 , L497 , L528 , L553 , L621 , L639		<code>\@sdblcolelt</code>	V805 , V825 , V854
<code>\@qend</code>	f136 , f671 , I187	<code>\@seccntformat</code>	N60 , N111
<code>\@qrelax</code>	f137 , f671	<code>\@secondoftwo</code>	
<code>\@raw@classoptionslist</code>	S11 , S698	a88 , c84 , e36 , e40 , f212 , f445 , f539 ,	
<code>\@rc@ifdefinable</code>	f130 , f132 , f238 , r14	f589 , f640 , f647 , f667 , f714 , q442 ,	
<code>\@reargdef</code>	f122	q487 , q504 , r131 , s156 , s161 , z365 ,	
<code>\@refundefined</code>		D71 , D838 , D889 , D905 , F21 , R17 ,	
.	q55 , q124 , q179 , F3 , G47 , G87	S155 , S194 , S206 , S243 , S261 , S1629	
<code>\@reinserts</code>	V327 , V330 , V516	<code>\@secpenalty</code>	o17 , N36 , N50
<code>\@remove@eq@value</code>	S432 , S497 , S557	<code>\@sect</code>	N54 , N55
<code>\@remove@tlig</code>	r996 , r1004	<code>\@seqncr</code>	H411
<code>\@remove@tlig@</code>	r996 , r997	<code>\@set@curr@file@aux</code>	T328
<code>\@remove@tlig@@</code>	r997 , r1000	<code>\@setckpt</code>	q375 , q382 , G23 , G76
<code>\@remove@tlig@@@</code>	r1030	<code>\@setfloattypecounts</code> V1074 , V1224 ,	
<code>\@remove@tlig@@@@</code>	r1010	V1395 , V1476 , V1571 , V1692 , V2000	
<code>\@removeelement</code>	k32 , S556 , S565	<code>\@setfontsize</code>	z490

File Key: a=ltldirchk.dtx, b=ltplain.dtx, c=ltvers.dtx, d=ltluatex.dtx, e=ltexpl.dtx, f=ltdefns.dtx, g=ltcmd.dtx, h=lt hooks.dtx, i=ltcmdhooks.dtx, j=ltalloc.dtx, k=ltcntrl.dtx, l=lterror.dtx, m=ltpar.dtx, n=lt para.dtx, o=lt space.dtx, p=lt logos.dtx, q=lt files.dtx, r=ltoutenc.dtx, s=lt counts.dtx, t=lt length.dtx, u=ltfssbas.dtx, v=ltfssaxes.dtx, w=ltfssstrc.dtx, x=ltfsscmp.dtx, y=ltfssdcl.dtx, z=ltfssini.dtx, A=fontdef.dtx, B=preload.dtx, C=ltfntcmd.dtx, D=lttextcomp.dtx, E=ltpageno.dtx, F=ltxref.dtx, G=ltmiscen.dtx, H=ltmath.dtx, I=ltlists.dtx, J=ltboxes.dtx, K=lttab.dtx, L=lt pictur.dtx, M=ltthm.dtx, N=ltsect.dtx, O=ltfloat.dtx, P=ltidxglo.dtx, Q=ltbibl.dtx, R=lt page.dtx, S=ltclass.dtx, T=ltfilehook.dtx, U=ltshipout.dtx, V=ltoutput.dtx, W=lthyphen.dtx, X=ltfinal.dtx

- \@setfps [O34](#)
- \@setfpsbit [O73](#), [O75](#),
[O77](#), [O85](#), [O143](#), [O146](#), [O149](#), [V2044](#)
- \@setmarks [V2214](#), [V2216](#), [V2231](#)
- \@setminipage
..... [J352](#), [O21](#), [O177](#), [O185](#), [O376](#)
- \@setnobreak [O179](#), [O375](#)
- \@setpar [m3](#), [I78](#)
- \@setref [F10](#)
- \@setsize [z490](#)
- \@settab [K71](#), [K93](#)
- \@settodim [t17](#)
- \@settopoint [t22](#)
- \@setupverbvisiblespace
..... [G473](#), [G484](#), [G496](#), [G513](#), [G528](#)
- \@sffamilyhook . [z261](#), [z309](#), [z317](#), [z344](#)
- \@sharp [K196](#), [K235](#), [K265](#),
[K280](#), [K281](#), [K301](#), [K303](#), [K305](#), [K333](#)
- \@shipoutsetup [V590](#)
- \@shortstack [L132](#), [L133](#)
- \@show@... [92](#)
- \@show@DeclareRobustCommand
..... [f507](#), [f521](#), [f566](#), [f617](#), [f620](#)
- \@show@newcommand
..... [94](#), [f508](#), [f521](#), [f574](#), [f600](#), [f617](#), [f623](#)
- \@show@newcommand@aux . [95](#), [f600](#), [f627](#)
- \@showcommandlisthook
..... [92](#), [f504](#), [f506](#), [f514](#)
- \@skipping@modulefalse [c142](#), [c161](#), [c176](#)
- \@skipping@moduletrue ... [c141](#), [c165](#)
- \@sline [L167](#), [L179](#), [L184](#), [L269](#)
- \@slowromancap [s110](#), [s111](#)
- \@spaces [l169](#)
- \@specialoutput [V256](#)
- \@specialpagefalse . [V97](#), [V607](#), [V665](#)
- \@specialpagetrue [R9](#)
- \@specialstyle [R9](#), [V607](#), [V665](#)
- \@sptoken [f680](#), [f690](#)
- \@sqrt [H342](#)
- \@ssect [N53](#), [N112](#)
- \@stackcr [L139](#), [L142](#)
- \@star@or@long [f72](#),
[f77](#), [f124](#), [f146](#), [f152](#), [f178](#), [f187](#), [f221](#)
- \@startcolumn [V263](#), [V270](#), [V775](#)
- \@startdblcolumn
..... [V775](#), [V2236](#), [V2239](#), [V2276](#), [V2282](#)
- \@startfield
..... [K28](#), [K46](#), [K92](#), [K104](#), [K125](#), [K133](#)
- \@startline
..... [K20](#), [K57](#), [K62](#), [K68](#), [K83](#), [K154](#)
- \@startpbox
..... [K193](#), [K236](#), [K266](#), [K332](#), [K384](#), [K386](#)
- \@startsection [N39](#)
- \@starttoc [N149](#)
- \@stopfield [K32](#), [K48](#), [K86](#),
[K93](#), [K125](#), [K127](#), [K140](#), [K142](#), [K154](#)
- \@stopline [K30](#), [K56](#), [K85](#)
- \@stpelt [s20](#), [s23](#)
- \@string@makeletter [99](#), [f701](#)
- \@strip@args [r76](#)
- \@strip@tex@ext
..... [307](#), [q226](#), [q236](#), [q238](#), [q243](#), [q273](#)
- \@strip@tex@ext@aux [q243](#), [q274](#)
- \@svector [L244](#), [L259](#), [L269](#)
- \@sverb . [595](#), [G500](#), [G563](#), [G572](#), [G575](#)
- \@svsec [N57](#), [N60](#), [N66](#), [N78](#)
- \@svsechd [N76](#), [N101](#), [N121](#)
- \@swaptwoargs
..... [q521](#), [q523](#), [S820](#), [T153](#), [T167](#), [T185](#)
- \@sxverbatim [G408](#), [G485](#), [G492](#)
- \@tabacckludge [r223](#),
[r225](#), [r454](#), [r455](#), [D193](#), [D200](#), [D202](#)
- \@tabacol [K178](#), [K258](#)
- \@tabarray [K170](#), [K180](#), [K181](#)
- \@tabclassiv [K180](#), [K330](#)
- \@tabclassz [K179](#), [K282](#)
- \@tabcr [K56](#), [K73](#)
- \@tabfbox [K16](#), [K80](#), [K82](#)
- \@tablab [K72](#), [K126](#)
- \@tabminus [K72](#), [K117](#)
- \@tabplus [K72](#), [K110](#)
- \@tabpush
..... [K11](#), [K77](#), [K85](#), [K140](#), [K143](#), [K145](#)
- \@tabrj [K72](#), [K124](#)
- \@tabular [K174](#), [K177](#), [K178](#)
- \@tabularcr [K180](#), [K208](#)
- \@tempa [F47](#), [F48](#), [F59](#), [F60](#)
- \@tempboxa .. [j13](#), [r69](#), [t17](#), [t18](#), [J29](#),
[J30](#), [J31](#), [J32](#), [J37](#), [J38](#), [J39](#), [J40](#),
[J175](#), [J205](#), [J212](#), [J222](#), [J343](#), [J366](#),
[J456](#), [J457](#), [J458](#), [J465](#), [J466](#), [J467](#),
[J468](#), [L304](#), [L305](#), [L447](#), [L448](#),
[L486](#), [L491](#), [L496](#), [L497](#), [L515](#),
[L520](#), [L527](#), [L528](#), [L542](#), [L545](#),
[L552](#), [L553](#), [L614](#), [L615](#), [L620](#),
[L621](#), [L630](#), [L631](#), [L638](#), [L639](#),
[L724](#), [L742](#), [L780](#), [L798](#), [N138](#),
[N139](#), [O322](#), [O380](#), [V305](#), [V377](#),
[V382](#), [V383](#), [V424](#), [V429](#), [V430](#),
[V566](#), [V626](#), [V633](#), [V634](#), [V685](#),
[V692](#), [V693](#), [V721](#), [V725](#), [V737](#),
[V743](#), [V750](#), [V751](#), [V752](#), [V753](#),
[V757](#), [V765](#), [I205](#), [I211](#), [I212](#), [I214](#)
- \@tempcnta [j7](#), [y771](#), [y772](#),
[y773](#), [y774](#), [y778](#), [K242](#), [K243](#),
[K244](#), [K245](#), [L187](#), [L188](#), [L214](#),
[L215](#), [L216](#), [L229](#), [L230](#), [L231](#),
[L232](#), [L239](#), [L240](#), [L254](#), [L255](#),

File Key: a=ltldirchk.dtx, b=ltplain.dtx, c=ltvers.dtx, d=ltluatex.dtx, e=ltexpl.dtx, f=ltdefs.dtx, g=ltcmd.dtx, h=lt hooks.dtx, i=ltcmdhooks.dtx, j=ltalloc.dtx, k=ltcntrl.dtx, l=lterror.dtx, m=ltpar.dtx, n=lt para.dtx, o=lt space.dtx, p=ltlogos.dtx, q=ltfiles.dtx, r=ltoutenc.dtx, s=ltcounts.dtx, t=ltlength.dtx, u=ltfssbas.dtx, v=ltfssaxes.dtx, w=ltfssstrc.dtx, x=ltfsscmp.dtx, y=ltfssdcl.dtx, z=ltfssini.dtx, A=fontdef.dtx, B=preload.dtx, C=ltfntcmd.dtx, D=lttextcomp.dtx, E=ltpageno.dtx, F=ltxref.dtx, G=ltmiscen.dtx, H=ltmath.dtx, I=ltlists.dtx, J=ltboxes.dtx, K=lttab.dtx, L=lt pictur.dtx, M=ltthm.dtx, N=ltsect.dtx, O=ltfloat.dtx, P=ltidxglo.dtx, Q=ltbibl.dtx, R=ltpage.dtx, S=ltclass.dtx, T=ltfilehook.dtx, U=ltshipout.dtx, V=ltoutput.dtx, W=lthyphen.dtx, X=ltfinal.dtx

L270, L271, L276, L278, L279,
 L280, L281, L282, L285, L287,
 L288, L289, L290, L291, L292,
 L293, L294, L295, L296, L335,
 L336, L337, L338, L339, L360,
 L361, L362, L363, L364, L365,
 L392, L393, L394, L395, L396,
 L414, L416, L418, L419, L421,
 L423, L438, L439, L440, L442,
 L444, L446, L448, L562, L567,
 L575, L579, L616, L617, L618,
 L619, L632, L633, L635, L636,
 L658, L659, L660, L661, L662,
 L663, L711, L731, L771, L787,
 O62, O68, O70, O79, O80, O90,
 O91, O131, O137, O139, O152,
 O153, O159, O160, S1078, S1080,
 S1081, S1082, S1209, S1211, S1212,
 S1213, S1323, S1325, S1326, S1327,
 V16, V18, V20, V934, V935, V936,
 V937, V957, V958, V959, V960,
 V982, V985, V1018, V1021, V1132,
 V1294, V1574, V1577, V1695,
 V1698, V1813, V1815, V1818,
 V1820, V1822, V1844, V2034,
 V2035, V2039, V2045, V2049,
 X213, X218, X219, X220, X350,
 X352, X353, X354, X361, X363,
 X364, X365, X371, X373, X374,
 X375, X382, X384, X385, X386,
 X412, X414, X415, X416, X438,
 X440, X441, X442, X448, X450,
 X451, X452, X481, X486, X487, X488
 \@tempcntb j7, y772, y776,
 y778, L279, L280, L281, L283,
 L284, L285, L562, L563, L567,
 L568, L575, L576, L579, L580,
 O88, O89, O90, O157, O158, O159,
 V17, V20, V21, V2045, V2046,
 V2047, X214, X218, X482, X486
 \@tempdima j10, u219, u224, H173, H176,
 H182, J43, J44, J204, J205, J210,
 J211, J212, J214, J265, J266, J341,
 J345, J423, J426, J427, J454, J456,
 J462, J465, K35, K36, K37, K88,
 K89, K90, K91, K218, K219, L210,
 L211, L213, L214, L215, L216,
 L217, L218, L437, L438, L439,
 L448, L487, L488, L492, L493,
 L516, L517, L521, L523, L543,
 L544, L546, L548, L617, L619,
 L634, L636, L637, L657, L658,
 L659, N196, N197, N219, N220,
 N233, O196, O198, O218, O220,
 O258, O259, O260, V229, V230,
 V231, V487, V489, V535, V537,
 V538, V543, V548, V552, V557,
 V561, V917, V920, V940, V950,
 V962, V972, V1637, V1638, V1641,
 V1642, V1762, V1763, V1767,
 V1768, V1823, V1824, V1825,
 V1826, V1829, V1832, V1835,
 V1837, V2152, V2153, V2155, V2156
 \@tempdimb j10, u220, u225, u624, u628, w179,
 w180, w437, w460, w461, w470,
 w471, w475, w493, w496, w499,
 w501, J268, J269, J424, J427, J455,
 J457, J463, J466, L211, L212, L357,
 L359, L362, L365, L482, L484,
 L485, L510, L513, L514, L539,
 L540, L541, L612, L613, L622,
 L628, L629, L640, L648, L649,
 L654, V940, V941, V942, V943,
 V950, V962, V963, V964, V965, V972
 \@tempdimc j10, w454,
 w455, w457, w458, w460, w461,
 J53, J54, J64, J65, J425, J426,
 J427, L30, L31, L32, L33, L34, L35,
 L60, L61, L63, L64, L332, L333, L334
 \@tempskipa j14, o44,
 o47, o48, o285, o292, o294, o297,
 w181, w182, N42, N44, N45, N50,
 N62, N63, N88, N89, N91, N103,
 N104, N113, N114, V1872, V1873,
 V1875, V1883, I116, I117, I118,
 I150, I152, I153, I154, I222, I223, I224
 \@tempskipb 291, j14, o220, o222, o224, o227,
 o229, o243, o261, o283, o285, o286,
 o290, o292, o294, o295, o318, o321
 \@tempswafalse a78,
 b262, q301, q348, r1529, u94, u673,
 y313, y368, y432, y513, y1033,
 y1039, G25, G78, G423, G444, Q13,
 S1055, S1071, S1186, S1202, T277,
 T298, V988, V1024, V1580, V1701
 \@tempswatruw a79, b268, q299, q304, q346,
 q351, r1532, r1533, u97, u674, u675,
 u678, u681, y316, y371, y435, y516,
 y996, G121, G428, G449, Q13,
 S1052, S1183, T277, T298, V1582,
 V1605, V1703, V1728, V2113, V2130
 \@temptokena 582, j16, G133, G137, G146, G159,
 G160, R26, R27, R34, R35, R48, R49

File Key: a=ltldirchk.dtx, b=ltplain.dtx, c=ltvers.dtx, d=ltluatex.dtx, e=ltexpl.dtx,
 f=ltdefs.dtx, g=ltcmd.dtx, h=lt hooks.dtx, i=ltcmdhooks.dtx, j=ltalloc.dtx, k=ltcntrl.dtx,
 l=lterror.dtx, m=ltpar.dtx, n=ltpara.dtx, o=lt space.dtx, p=ltlogos.dtx, q=ltfiles.dtx,
 r=ltoutenc.dtx, s=ltcounts.dtx, t=ltlength.dtx, u=ltfssbas.dtx, v=ltfssaxes.dtx,
 w=ltfssstrc.dtx, x=ltfsscmp.dtx, y=ltfssdcl.dtx, z=ltfssini.dtx, A=fontdef.dtx,
 B=preload.dtx, C=ltfntcmd.dtx, D=lttextcomp.dtx, E=ltpageno.dtx, F=ltxref.dtx,
 G=ltmiscen.dtx, H=ltmath.dtx, I=ltlists.dtx, J=ltboxes.dtx, K=lttab.dtx, L=lt pictur.dtx,
 M=ltthm.dtx, N=ltsect.dtx, O=ltfloat.dtx, P=ltidxglo.dtx, Q=ltbibl.dtx, R=lt page.dtx,
 S=ltclass.dtx, T=ltfilehook.dtx, U=ltshipout.dtx, V=ltoutput.dtx, W=lthyphen.dtx,
 X=ltfinal.dtx

- \@testdef [G24](#), [G77](#), [G119](#)
- \@testfalse [V12](#), [V14](#), [V15](#)
- \@testfp [V882](#), [V902](#),
[V938](#), [V961](#), [V2037](#), [V2166](#), [V2183](#)
- \@testopt [f33](#), [f79](#),
[f99](#), [f103](#), [f148](#), [o7](#), [o8](#), [o9](#), [o10](#), [H382](#)
- \@testpach [K270](#), [K348](#)
- \@testpatch [K348](#)
- \@testtrue [V13](#), [V21](#),
[V356](#), [V885](#), [V904](#), [V944](#), [V966](#), [V2041](#)
- \@testwrongwidth [V345](#),
[V883](#), [V939](#), [V1112](#), [V1426](#), [V1631](#)
- \@text@composite ... [r76](#), [r1061](#), [r1066](#)
- \@text@composite@x [r76](#)
- \@textbottom
..... [R54](#), [R56](#), [V511](#), [V549](#), [V563](#), [V572](#)
- \@textfloatsheight
..... [V476](#), [V1087](#), [V1089](#),
[V1139](#), [V1140](#), [V1145](#), [V1240](#),
[V1242](#), [V1302](#), [V1304](#), [V1310](#), [V1986](#)
- \@textmin [O285](#), [O286](#),
[O299](#), [O300](#), [V112](#), [V1089](#), [V1093](#),
[V1096](#), [V1097](#), [V1242](#), [V1247](#),
[V1251](#), [V1252](#), [V1414](#), [V1499](#),
[V1598](#), [V1600](#), [V1616](#), [V1720](#),
[V1722](#), [V1740](#), [V2094](#), [V2096](#), [V2098](#)
- \@textsubscript [O424](#), [O432](#)
- \@textsuperscript . [O401](#), [O403](#), [O404](#)
- \@texttop . [R54](#), [R56](#), [V507](#), [V530](#), [V572](#)
- \@tf@r [k25](#), [k26](#)
- \@tfor [k25](#),
[q488](#), [q505](#), [q649](#), [C88](#), [J57](#), [J76](#),
[K268](#), [L478](#), [L506](#), [L535](#), [O63](#), [O132](#)
- \@tforloop [k27](#), [k28](#), [k30](#)
- \@thanks [N11](#), [N34](#)
- \@thefnmark [J380](#), [J398](#),
[O400](#), [O401](#), [O453](#), [O458](#), [O472](#),
[O491](#), [O503](#), [O508](#), [O519](#), [O524](#), [O535](#)
- \@thefoot [V124](#),
[V610](#), [V613](#), [V640](#), [V669](#), [V672](#), [V699](#)
- \@thehead [V123](#),
[V610](#), [V612](#), [V630](#), [V669](#), [V671](#), [V689](#)
- \@themargin [V74](#),
[V611](#), [V613](#), [V625](#), [V670](#), [V672](#), [V684](#)
- \@themark [R25](#), [R26](#), [R33](#), [R34](#), [R49](#), [R52](#)
- \@thirdofthree [f216](#), [r197](#)
- \@thm [M12](#), [M18](#), [M24](#), [M26](#)
- \@thmcounter [M11](#), [M17](#), [M33](#)
- \@thmcountersep [M10](#), [M33](#)
- \@title [N7](#), [N31](#)
- \@tocrmarg [N192](#), [N215](#)
- \@toodeep [I203](#), [I36](#), [I232](#), [I243](#)
- \@toplist [V64](#), [V384](#), [V385](#),
[V431](#), [V432](#), [V716](#), [V722](#), [V732](#),
[V733](#), [V1025](#), [V1037](#), [V1921](#), [V1948](#)
- \@topnewpage [V199](#)
- \@topnum [O271](#), [V105](#), [V1022](#), [V1023](#),
[V1037](#), [V1041](#), [V1458](#), [V1463](#),
[V1551](#), [V1558](#), [V1912](#), [V1939](#), [V1980](#)
- \@toproom [O273](#),
[V106](#), [V1025](#), [V1037](#), [V1913](#), [V1940](#)
- \@topsep [I1](#), [I71](#), [I73](#), [I171](#)
- \@topsepadd [I1](#), [I59](#), [I61](#), [I71](#), [I124](#)
- \@totalleftmargin [614](#), [G419](#), [G441](#),
[J294](#), [J315](#), [K35](#), [K76](#), [K81](#), [I9](#), [I53](#), [I54](#)
- \@trivlist [I48](#), [I57](#), [I92](#)
- \@tryfcolumn [V778](#),
[V798](#), [V816](#), [V832](#), [V2167](#), [V2184](#)
- \@trylist [V841](#), [V844](#), [V877](#), [V897](#), [V919](#)
- \@ttfamilyhook . [z261](#), [z314](#), [z318](#), [z345](#)
- \@twoclasseserror [S596](#), [S1041](#)
- \@twocolumnfalse [V99](#), [V147](#)
- \@twocolumntrue [V206](#)
- \@twoloadclasserror [S845](#), [S956](#), [S1036](#)
- \@twosidefalse [V100](#)
- \@typein [f32](#), [f33](#), [f40](#), [f48](#)
- \@typeset@protect [84](#),
[f102](#), [f243](#), [f250](#), [f252](#), [r26](#), [r32](#), [r210](#),
[r218](#), [z491](#), [G193](#), [G238](#), [G256](#), [X344](#)
- \@uclclist .. [r1498](#), [r1499](#), [r1552](#), [X551](#)
- \@undefind [L57](#), [L71](#), [L82](#), [L91](#),
[L162](#), [L174](#), [L237](#), [L252](#), [L313](#), [L371](#)
- \@undefined [822](#), [a68](#), [a69](#), [a108](#), [a109](#),
[a110](#), [a131](#), [a139](#), [a147](#), [a154](#), [a205](#),
[a209](#), [a235](#), [a242](#), [a326](#), [a327](#), [b65](#),
[b81](#), [b105](#), [b106](#), [b121](#), [b122](#), [b127](#),
[b136](#), [b149](#), [b184](#), [b189](#), [b222](#), [b223](#),
[b246](#), [b256](#), [b291](#), [b338](#), [b348](#), [b350](#),
[b505](#), [b573](#), [b613](#), [b628](#), [c50](#), [c59](#),
[d2](#), [d15](#), [d16](#), [d17](#), [d29](#), [d30](#), [d74](#),
[d84](#), [d173](#), [d181](#), [d189](#), [d197](#), [d229](#),
[d230](#), [d231](#), [d232](#), [d233](#), [d234](#), [d235](#),
[d236](#), [d237](#), [d238](#), [d239](#), [d240](#), [d241](#),
[d242](#), [d243](#), [d244](#), [d245](#), [d246](#), [d247](#),
[d253](#), [e4](#), [e13](#), [e14](#), [e15](#), [e16](#), [e146](#),
[e147](#), [e148](#), [e149](#), [e150](#), [e151](#), [f34](#),
[f223](#), [f307](#), [f308](#), [f339](#), [f367](#), [f371](#),
[f372](#), [f389](#), [f403](#), [f450](#), [f451](#), [f452](#),
[f453](#), [f454](#), [f455](#), [f456](#), [f490](#), [f491](#),
[f492](#), [f493](#), [f494](#), [f495](#), [f513](#), [f514](#),
[f618](#), [f619](#), [f620](#), [f621](#), [f622](#), [f623](#),
[f625](#), [f626](#), [f627](#), [f642](#), [f649](#), [f723](#),
[f724](#), [f725](#), [f734](#), [l28](#), [n131](#), [n132](#),
[n133](#), [n134](#), [n135](#), [o89](#), [o470](#), [o476](#),
[q61](#), [q62](#), [q129](#), [q130](#), [q184](#), [q185](#),
[q273](#), [q274](#), [q413](#), [q414](#), [q415](#), [q420](#),

File Key: a=ltldirchk.dtx, b=ltplain.dtx, c=ltvers.dtx, d=ltluatex.dtx, e=ltexpl.dtx, f=ltdefns.dtx, g=ltcmd.dtx, h=lt hooks.dtx, i=ltcmdhooks.dtx, j=ltalloc.dtx, k=ltcntrl.dtx, l=lterror.dtx, m=ltpar.dtx, n=lt para.dtx, o=lt space.dtx, p=ltlogos.dtx, q=ltfiles.dtx, r=ltoutenc.dtx, s=ltcounts.dtx, t=ltlength.dtx, u=ltfssbas.dtx, v=ltfssaxes.dtx, w=ltfssstrc.dtx, x=ltfsscmp.dtx, y=ltfssdcl.dtx, z=ltfssini.dtx, A=fontdef.dtx, B=preload.dtx, C=ltfntcmd.dtx, D=lttextcomp.dtx, E=ltpageno.dtx, F=ltxref.dtx, G=ltmiscen.dtx, H=ltmath.dtx, I=ltlists.dtx, J=ltboxes.dtx, K=lttab.dtx, L=lt pictur.dtx, M=ltthm.dtx, N=ltsect.dtx, O=ltfloat.dtx, P=ltidnglo.dtx, Q=ltbibl.dtx, R=ltpage.dtx, S=ltclass.dtx, T=ltfilehook.dtx, U=ltshipout.dtx, V=ltoutput.dtx, W=lthyphen.dtx, X=ltfinal.dtx

q441, q462, q477, q532, r195, r197,
 r333, r335, r337, r339, r341, r343,
 r345, r347, r349, r351, r370, r372,
 r374, r458, r700, r703, s170, u434,
 u474, u536, u569, u633, u640, v388,
 v411, v420, v509, v510, v511, v512,
 v513, v514, v515, v516, v517, v518,
 v529, v534, v539, v603, v604, v605,
 v606, v607, v608, v609, v634, v714,
 v716, v717, v719, v720, v721, v723,
 w552, w553, x4, x5, x6, x7, x8, x9,
 x10, x11, x12, x13, x14, x15, x16,
 x17, x18, x19, x20, y629, y736, z35,
 z55, z166, z167, z168, z169, z170,
 z171, z172, z173, z174, z175, z176,
 z178, z179, z180, z325, z343, z344,
 z345, z387, z388, z389, z390, z391,
 z431, z432, z433, z434, z435, z436,
 z447, z569, A15, A50, A65, C37,
 C38, C39, C122, D159, D367, D617,
 D620, D621, D651, D652, D653,
 D656, D657, D658, D659, D660,
 D661, D666, D667, D668, D669,
 D674, D675, D676, D677, D680,
 D681, D682, D684, D685, D690,
 D691, D692, D693, D694, D695,
 D696, D697, D698, D699, D700,
 D701, D702, D703, D704, D705,
 D707, D708, D710, D711, D712,
 D713, D714, D715, D716, D717,
 D718, D719, D721, D722, D723,
 D724, D725, D726, D727, D728,
 D729, D731, D732, D733, D734,
 D735, D736, D737, D738, D739,
 D740, D741, D742, D743, D744,
 D745, D746, D747, D748, D749,
 D750, D751, D752, D753, D754,
 D755, D756, D757, D762, D763,
 D765, D766, D767, D768, D769,
 D770, D771, D772, D774, D775,
 D777, D778, D780, D781, D782,
 D783, D785, D786, D787, D789,
 D791, D792, D793, D794, D795,
 D796, D797, D799, D800, D801,
 D802, D803, D804, D805, D806,
 D807, F53, F69, F70, G101, G153,
 G154, G155, G156, G291, G292,
 G312, G313, G314, G315, G467,
 G494, G495, G496, G497, G530,
 H223, H233, H234, H270, H273,
 H316, H329, J21, J130, J194, J250,
 J417, J447, L18, L467, L468, N230,
 O5, O429, O448, O544, Q33, S4,
 S25, S140, S141, S142, S210, S341,
 S447, S497, S850, S885, S887, S893,
 S946, S961, S962, S977, S1091,
 S1169, S1172, S1222, S1301, S1304,
 S1314, S1315, S1316, S1317, S1318,
 S1319, S1320, S1391, S1394, S1410,
 S1430, S1437, T18, T19, T20, T21,
 T147, T187, T188, T195, T196,
 T197, T325, T453, T459, T460,
 T461, U429, U430, U431, U432,
 U433, U435, U436, U437, U444,
 U445, U447, U449, U450, U451,
 U454, U477, V36, V368, V369, X10,
 X18, X25, X40, X59, X68, X75,
 X84, X94, X146, X147, X256, X269,
 X282, X302, X327, X337, X338,
 X339, X368, X370, X409, X410,
 X429, X430, X431, X432, X433,
 X434, X435, X436, X437, X454,
 X470, X471, X472, X520, X521,
 X581, X616, X617, X618, X619, X620
 \@unexpandable@protect
 ... [f220](#), [f255](#), [f261](#), [f266](#), [q210](#), [K264](#)
 \@unknownoptionerror
 [S972](#), [S1011](#), [S1024](#)
 \@unprocessedoptions
 [780](#), [782](#), [S546](#), [S639](#),
[S850](#), [S886](#), [S887](#), [S943](#), [S947](#), [S1026](#)
 \@unused
 ... [b307](#), [f10](#), [f17](#), [l15](#), [l32](#), [l59](#), [S1415](#)
 \@unusedoptionlist
 .. [q18](#), [q20](#), [q87](#), [q89](#), [q144](#), [q146](#),
[S12](#), [S439](#), [S440](#), [S450](#), [S451](#), [S558](#), [S566](#)
 \@upline [L297](#), [L298](#), [L304](#)
 \@upordown [L195](#), [L196](#), [L204](#), [L225](#), [L273](#)
 \@upvector [L268](#), [L304](#)
 \@use@option [S470](#), [S486](#),
[S504](#), [S520](#), [S522](#), [S539](#), [S541](#), [S551](#)
 \@use@text@encoding [r150](#), [D16](#), [D1141](#)
 \@vbsphack [o219](#)
 \@verb [G563](#), [G572](#), [G575](#)
 \@verbatim .. [G413](#), [G459](#), [G483](#), [G492](#)
 \@verbvisiblespacebox
 [G478](#), [G479](#), [G482](#), [G497](#)
 \@vereq [A457](#), [A458](#)
 \@viipt [u716](#)
 \@vipt [u715](#)
 \@vipt [u714](#)
 \@vline [L166](#), [L178](#), [L297](#)
 \@vobeyspaces
[G406](#), [G459](#), [G485](#), [G513](#), [G528](#), [G575](#)
 \@vpt [u713](#)
 \@vspace [o330](#)
 \@vspace@calcify
[o79](#), [o101](#), [o241](#), [o337](#), [o342](#), [o352](#),

File Key: a=ltldirchk.dtx, b=ltplain.dtx, c=ltvers.dtx, d=ltluatex.dtx, e=ltexpl.dtx,
 f=ltdefns.dtx, g=ltcmd.dtx, h=lt hooks.dtx, i=ltcmdhooks.dtx, j=ltalloc.dtx, k=ltcntrl.dtx,
 l=lterror.dtx, m=ltpar.dtx, n=ltpara.dtx, o=lt space.dtx, p=ltlogos.dtx, q=ltfiles.dtx,
 r=ltoutenc.dtx, s=ltcounts.dtx, t=ltlength.dtx, u=ltfssbas.dtx, v=ltfssaxes.dtx,
 w=ltfssstrc.dtx, x=ltfsscmp.dtx, y=ltfssdcl.dtx, z=ltfssini.dtx, A=fontdef.dtx,
 B=preload.dtx, C=ltfntcmd.dtx, D=lttextcomp.dtx, E=ltpageno.dtx, F=ltxref.dtx,
 G=ltmiscen.dtx, H=ltmath.dtx, I=ltlists.dtx, J=ltboxes.dtx, K=lttab.dtx, L=lt pictur.dtx,
 M=ltthm.dtx, N=ltsect.dtx, O=ltfloat.dtx, P=ltidxglo.dtx, Q=ltbibl.dtx, R=ltpage.dtx,
 S=ltclass.dtx, T=ltfilehook.dtx, U=ltshipout.dtx, V=ltoutput.dtx, W=lthyphen.dtx,
 X=ltfinal.dtx

- o360, G342, H390, K62, K224, L148
- \@vspacer o330
- \@vtryfc V847, V855
- \@vvector L243, L258, L268
- \@warning l166
- \@wckptelt q376, q379
- \@whiledim k7, L123, L203
- \@whilenoop k3
- \@whilenum k3, K244, L104,
L116, L336, L338, L361, L364,
L393, L395, L416, L421, L731, L787
- \@whilesw k10, V264, V394,
V403, V441, V451, V2237, V2277
- \@whileswnoop k10
- \@wholewidth J160, J162, J163, J165,
J167, J168, J169, J170, L2, L126,
L129, L131, L299, L302, L350,
L359, L406, L413, L565, L578,
L590, L600, L679, L680, L728, L784
- \@width
b428, f26, o452, r289, r294, w192,
A618, J165, J167, J218, J225,
J427, J471, K188, K219, K347,
K366, L227, L299, L302, L325,
L333, L350, L359, L384, L392,
L406, L413, L565, L578, L728,
L784, O395, V1851, V2226, V2260
- \@wrglossary P25, P30
- \@wrindex P8, P13
- \@writeckpt q328, q361, q373
- \@writefile
582, 715, q32, q101, q159, G140, N183
- \@writesetup V590
- \@wrong@font@char
r161, u537, u571, u584
- \@wtryfc V857, V867
- \@x@protect f105, f242
- \@x@sf O513, O515
- \@xDeclareMathDelimiter .. y801, y857
- \@xaddvskip o219, o244, o262
- \@xarg L163, L166,
L175, L178, L185, L189, L190,
L226, L228, L238, L239, L243,
L253, L254, L258, L266, L274, L664
- \@xargarraycr K205, K214, K218
- \@xargdef f80
- \@xarraycr K202, K203
- \@xbitor V15, V17
- \@xcentercr G325, G332, G336
- \@xdblarg f693
- \@xdblfloat O264
- \@xdim
L84, L93, L105, L107, L117, L119,
L668, L732, L733, L734, L735,
L741, L788, L789, L790, L791, L797
- \@xeqncr H374
- \@exnoop K238, K248
- \@expast K239, K240
- \@xfloat O28, O29, O34, O266
- \@xfootnote O452, O455
- \@xfootnotemark O501, O505
- \@xfootnotenext O518, O521
- \@xhline K360, K361
- \@xifnch f681, f691
- \@xiipt u720, A164, A166, A167
- \@xipt u719, A163
- \@xivpt u721, A165, A167
- \@xmpar O324, O325
- \@xnewline o60, o61, o72, o73, o93
- \@xnext V10, V11
- \@xnthm M5, M6
- \@xobeysp
. 295, o420, G407, G408, G475, G479
- \@xprocess@options
.. S461, S478, S480, S496, S498, S512
- \@xpt u718, A162, A165, A166
- \@xsect N86, N87, N123
- \@xtabcr K56, K57
- \@xtabularcr K209, K210
- \@xthm M28, M29
- \@xtryfc V844, V872
- \@xtypein f33, f35, f42
- \@xverbatim G408, G459
- \@xvipt u722, A166, A168
- \@xxDeclareMathDelimiter . y786, y790
- \@xxpt u723, A167, A168
- \@xxvpt u724, A168
- \@xxxii j2, r425, r427,
O89, O158, V879, V880, V899,
V900, V935, V936, V958, V959, V2003
- \@xympar O328, O332, O378
- \@yarg ... L163, L167, L175, L179,
L185, L186, L195, L238, L244,
L253, L259, L268, L270, L297, L664
- \@yargarraycr K206, K216, K220
- \@yargd@f f107
- \@yargdef f84, f94, f107, f123
- \@ydim
L85, L94, L105, L108, L117, L119,
L668, L736, L737, L738, L739,
L740, L792, L793, L794, L795, L796
- \@yeqncr H374
- \@ympar O324, O329
- \@ynthm M5, M14
- \@ythm M28, M29
- \@ytryfc V890, V909, V913

File Key: a=ltldirchk.dtx, b=ltplain.dtx, c=ltvers.dtx, d=ltluatex.dtx, e=ltexpl.dtx, f=ltdefns.dtx, g=ltcmd.dtx, h=lthooks.dtx, i=ltcmdhooks.dtx, j=ltalloc.dtx, k=ltcntrl.dtx, l=terror.dtx, m=ltpar.dtx, n=ltpara.dtx, o=ltspace.dtx, p=ltlogos.dtx, q=ltfiles.dtx, r=ltoutenc.dtx, s=ltcounts.dtx, t=ltlength.dtx, u=ltfssbas.dtx, v=ltfssaxes.dtx, w=ltfssstrc.dtx, x=ltfsscmp.dtx, y=ltfssdcl.dtx, z=ltfssini.dtx, A=fontdef.dtx, B=preload.dtx, C=ltfntcmd.dtx, D=lttextcomp.dtx, E=ltpageno.dtx, F=ltxref.dtx, G=ltmiscen.dtx, H=ltmath.dtx, I=ltlists.dtx, J=ltboxes.dtx, K=lttab.dtx, L=ltpicture.dtx, M=ltthm.dtx, N=ltsect.dtx, O=ltfloat.dtx, P=ltidxglo.dtx, Q=ltbibl.dtx, R=ltpage.dtx, S=ltclass.dtx, T=ltfilehook.dtx, U=ltshipout.dtx, V=ltoutput.dtx, W=lthyphen.dtx, X=ltfinal.dtx

\@yyarg	V40
L185, L186, L187, L190, L274, L664	V30, V57
\@ztryfc	V918, V929
\@end	185
\accent@spacefactor	r70, r73, r74
\active@math@prime	H240, H241, V588
\add@accent	r65, r67
\add@percent@to@temptokena	583, G127, G143, G145, G154
\add@unicode@accent	r1040, r1044
\addto@hook	u152, u154, u712, y295, y391, y395, y412, y536, y542, y550, y566, y569, y572, y1005, y1012, y1015
\AddToHook	231
\AddToHookNext	231
\alloc@	b90, b91, b92, b93, b94, b95, b96, b97, b98, b99, b226, d22, d26, d38, u14
\alpha@elt	y45, y299, y486, y588, y1004, y1005
\alpha@list	y41, y43, y308, y474, y486, y531, y586, y587, y1000, y1006, y1007
\apptocmd	231
\atveryend@DEPRECATED	T541, T543
\best@size	w438, w462, w468, w474
\bf@def@ult	z190
\bfdef@ult	416, 489, 496, z135, z194, z205, z206, z207, z274
\bfdefault@previous	489, z201, z204, A106, A116
\bfseries@..	484
\bfseries@rm	484, 487, 496, z63, z142, z145, z167, z205, z210, z282
\bfseries@rm@kernel	487, z66, z142, z170
\bfseries@sf	482, 487, z63, z146, z149, z168, z206, z211, z283
\bfseries@sf@kernel	z67, z146, z171
\bfseries@tt	487, z63, z150, z153, z169, z207, z212, z284
\bfseries@tt@kernel	z68, z150, z172
\bm@b	J37
\bm@c	J37
\bm@l	J37
\bm@r	J37
\bm@s	J37
\bm@t	J37
\bx@A	V30, V57
\bx@AA	V40
\bx@B	V30, V57
\bx@BB	V40
\bx@C	V30, V57
\bx@CC	V40
\bx@D	V30, V57
\bx@DD	V40
\bx@E	V30, V57
\bx@EE	V40
\bx@F	V31, V58
\bx@FF	V41
\bx@G	V31, V58
\bx@GG	V41
\bx@H	V31, V58
\bx@HH	V41
\bx@I	V31, V58
\bx@II	V41
\bx@J	V31, V58
\bx@JJ	V41
\bx@K	V32, V59
\bx@KK	V42
\bx@L	V32, V59
\bx@LL	V42
\bx@M	V32, V59
\bx@MM	V42
\bx@N	V32, V59
\bx@NN	V42
\bx@O	V33, V60
\bx@OO	V43
\bx@P	V33, V60
\bx@PP	V43
\bx@Q	V33, V60
\bx@QQ	V43
\bx@R	V33, V60
\bx@RR	V43
\bx@S	V38
\bx@SS	V44
\bx@T	V38
\bx@TT	V44
\bx@U	V38
\bx@UU	V44
\bx@V	V38
\bx@VV	V44
\bx@W	V39
\bx@WW	V45
\bx@X	V39
\bx@XX	V45
\bx@Y	V39
\bx@YY	V45
\bx@Z	V39
\bx@ZZ	V25, V45, V55
\c@bottomnumber	O269, O274, V2292
\c@dbltopnumber	O268, O283, O297, V2299
\c@enumi	I227
\c@enumii	I227
\c@enumiv	I227
\c@equation	H336, H369, H511
\c@errorcontextlines	I163
\c@footnote	N12, O397, O507

File Key: a=ltldirchk.dtx, b=ltplain.dtx, c=ltvers.dtx, d=ltluatex.dtx, e=ltexpl.dtx, f=ltdefns.dtx, g=ltcmd.dtx, h=lthooks.dtx, i=ltcmdhooks.dtx, j=ltalloc.dtx, k=ltcntrl.dtx, l=lterror.dtx, m=ltpar.dtx, n=ltpara.dtx, o=ltspace.dtx, p=ltlogos.dtx, q=ltfiles.dtx, r=ltoutenc.dtx, s=ltcounts.dtx, t=ltlength.dtx, u=ltfssbas.dtx, v=ltfssaxes.dtx, w=ltfssstrc.dtx, x=ltfsscmp.dtx, y=ltfssdcl.dtx, z=ltfssini.dtx, A=fontdef.dtx, B=preload.dtx, C=ltfntcmd.dtx, D=lttextcomp.dtx, E=ltpageno.dtx, F=ltxref.dtx, G=ltmiscen.dtx, H=ltmath.dtx, I=ltlists.dtx, J=ltboxes.dtx, K=lttab.dtx, L=ltpictur.dtx, M=ltthm.dtx, N=ltsect.dtx, O=ltfloat.dtx, P=ltidxglo.dtx, Q=ltbibl.dtx, R=ltpage.dtx, S=ltclass.dtx, T=ltfilehook.dtx, U=ltshipout.dtx, V=ltoutput.dtx, W=lthyphen.dtx, X=ltfinal.dtx

- \c@mpfootnote J348, O399
- \c@ncel A461, A462
- \c@page E3, E6, E7, V138, V1818
- \c@secnumdepth . N56, N71, N81, N140
- \c@tocdepth N140, N190, N213
- \c@topnumber O267, O271, V2288
- \c@totalnumber ... O270, O276, V2295
- \c@totalpages U348, U432
- \calculate@math@sizes ... u620, w219
- \catcodetable@atletter d93, d238
- \catcodetable@initex d93, d235
- \catcodetable@latex d93, d237
- \catcodetable@string d93, d236
- \c@p@elt u96, u116,
u127, u128, u149, u152, u154, y233,
y315, y370, y434, y515, X458, X459
- \c@p@list
... u98, u114, u128, u156, u157,
y251, y317, y372, y436, y517, X459
- \cf@encoding r34, r41,
r44, r51, r154, u256, u266, u276, u326
- \ch@ck b206, b207, b208, b209,
b236, b248, b249, b250, b251, b279,
b281, b293, b294, b295, b296, b302,
S1096, S1115, S1227, S1248, S1340
- \char@if@alph f701
- \chardef@text@cmd r3
- \check@command f187, f189
- \check@icl
.. C9, C44, C49, C55, C63, C70, C72
- \check@icr
.. C9, C44, C50, C56, C64, C73, C78
- \check@mathfonts .. p5, r302, r328,
r360, r1246, u348, u350, w250, D672
- \check@nocorr@ C46
- \check@orange w379, w380
- \check@single w378, w400
- \cl@ckpt q376, s35
- \cl@page E4
- \col@number ... V95, V148, V208, V220
- \color@begingroup u643, u703, H104,
H134, J29, J89, J176, J344, J381,
J399, K47, K51, O474, O493, V491
- \color@endbox .. J89, O253, O348,
O366, V224, V631, V641, V690, V700
- \color@endgroup
..... u648, u709, H104, H134,
J29, J89, J134, J155, J178, J364,
J385, J402, K49, O478, O496, V495
- \color@hbox
..... J89, V628, V638, V687, V697
- \color@setgroup J89, J134, J153
- \color@vbox J89,
O96, O165, O339, O357, O381, V215
- \conditionally@traceoff
..... 261, g217, g229, l239
- \conditionally@traceon ... g250, l239
- \contionally@traceon 261
- \copy@kernel@robust@command f541, f625
- \count@ a66, a179, a180,
a181, a186, b41, b191, b192, b197,
b199, b205, b206, b207, b208, b209,
b210, b428, b429, c14, c15, c16,
c17, c18, c20, f169, f173, f275, f300,
u677, u683, u685, w22, w302, w304,
w326, w327, y292, y294, y298,
y617, y618, y619, y665, y666, y667,
y726, y727, y728, y774, y775, y776,
y814, y815, y816, y822, y823, y824,
y868, y869, y870, y876, y877, y878,
y937, y938, y939, y945, y946, y947,
C115, C118, L730, L731, L732,
L735, L736, L739, L743, L786,
L787, L788, L791, L792, L795,
L799, S1139, S1141, S1142, S1143,
S1272, S1274, S1275, S1276, S1361,
S1363, S1364, S1365, X225, X226,
X233, X235, X525, X526, X533, X535
- \counterwithin@s s77, s78, s98
- \counterwithin@x s77, s80, s99
- \counterwithout@s s67, s68, s95
- \counterwithout@x s67, s70, s96
- \curr@fontshape . r180, u88, u363,
u371, u375, u377, u519, u525, u528,
u537, u544, u546, u554, u560, u563,
u571, u578, u580, v454, w92, w100,
w142, w169, w477, w497, w529,
w545, w560, y255, y260, z409, z418
- \curr@math@size
..... u352, w256, w262, w267, w284
- \declare@commandcopy
..... 91, f464, f468, f473, f476, f493
- \declare@commandcopy@let
..... 91, f481, f485, f495, f563, f564
- \declare@file@substitution
..... 806, T245, T528, U508
- \declare@robustcommand f221
- \DeclareEncodingSubset@aux
..... D41, D43, D59
- \DeclareFontEncoding@
.. u122, u124, u139, X369, X389, X455
- \DeclareFontEncoding@saved
..... X369, X389, X471
- \DeclareFontShape@ u21, u22
- \DeclareRobustCommand 232, 238
- \DeclareSymbolFontAlphabet@
..... y992, y995
- \def 232, 234, 242, 243

File Key: a=ltldirchk.dtx, b=ltplain.dtx, c=ltvers.dtx, d=ltluatex.dtx, e=ltexpl.dtx, f=ltdefns.dtx, g=ltcmd.dtx, h=lt hooks.dtx, i=ltcmdhooks.dtx, j=ltalloc.dtx, k=ltcntrl.dtx, l=lterror.dtx, m=ltpar.dtx, n=lt para.dtx, o=lt space.dtx, p=ltlogos.dtx, q=ltfiles.dtx, r=ltoutenc.dtx, s=ltcounts.dtx, t=ltlength.dtx, u=ltfssbas.dtx, v=ltfssaxes.dtx, w=ltfssstrc.dtx, x=ltfsscmp.dtx, y=ltfssdcl.dtx, z=ltfssini.dtx, A=fontdef.dtx, B=preload.dtx, C=ltfntcmd.dtx, D=lttextcomp.dtx, E=ltpageno.dtx, F=ltxref.dtx, G=ltmiscen.dtx, H=ltmath.dtx, I=ltlists.dtx, J=ltboxes.dtx, K=lttab.dtx, L=lt pictur.dtx, M=ltthm.dtx, N=ltsect.dtx, O=ltfloat.dtx, P=ltidxglo.dtx, Q=ltbibl.dtx, R=ltpage.dtx, S=ltclass.dtx, T=ltfilehook.dtx, U=ltshipout.dtx, V=ltoutput.dtx, W=lthyphen.dtx, X=ltfinal.dtx

`\default@ds`
 . . [S428](#), [S457](#), [S521](#), [S540](#), [S970](#), [S972](#)
`\default@family` [u129](#), [u161](#),
 [u487](#), [u501](#), [u504](#), [u529](#), [u564](#), [X460](#)
`\default@M` [u136](#), [u176](#), [u179](#), [u183](#), [X467](#)
`\default@mextra` [x10](#), [x89](#)
`\default@series` [u129](#), [u162](#),
 [u488](#), [u502](#), [u505](#), [u526](#), [u561](#), [X460](#)
`\default@shape` [u130](#), [u163](#),
 [u489](#), [u503](#), [u506](#), [u524](#), [u559](#), [X461](#)
`\default@T` [u170](#), [u173](#), [u183](#), [u272](#)
`\define@mathalphabet` [x18](#), [x131](#)
`\define@mathgroup` [x19](#), [x135](#)
`\define@newfont` [u355](#), [u364](#)
`\delayed@f@adjustment` . [385](#), [412](#),
 [429](#), [501](#), [u290](#), [v395](#), [v396](#), [v397](#),
 [v399](#), [v400](#), [v411](#), [v616](#), [v617](#), [v619](#),
 [v620](#), [w122](#), [w126](#), [w134](#), [w138](#), [z515](#)
`\delayed@merge@font@series`
 . [v396](#), [v460](#), [v475](#), [v514](#), [w133](#), [w136](#)
`\delayed@merge@font@shape`
 . [v617](#), [v666](#), [v721](#), [w132](#), [w135](#)
`\development@branch@name`
 . [c11](#), [c36](#), [c50](#), [c51](#), [c52](#), [c59](#), [c60](#), [c61](#)
`\dimen@` [b41](#), [b425](#),
 [b426](#), [b462](#), [b463](#), [b465](#), [b467](#), [l28](#),
 [l29](#), [o349](#), [o354](#), [o383](#), [o388](#), [r429](#),
 [r430](#), [r432](#), [r433](#), [r796](#), [r797](#), [u214](#),
 [u216](#), [u222](#), [u235](#), [u238](#), [u242](#), [u623](#),
 [u624](#), [u625](#), [u629](#), [w451](#), [w452](#),
 [w453](#), [w454](#), [w458](#), [D102](#), [D104](#),
 [D931](#), [D933](#), [H72](#), [H73](#), [H186](#),
 [H187](#), [H188](#), [H189](#), [J464](#), [J467](#),
 [K176](#), [K177](#), [V508](#), [V510](#), [V531](#), [V533](#)
`\dimen@i` [b41](#)
`\dimen@ii` [b41](#), [u218](#), [u223](#)
`\disable@package@load`
 . [807](#), [821](#), [T442](#), [U466](#)
`\display` [H191](#), [H195](#), [H196](#)
`\do@add@percent@to@temptokena` ...
 . [G131](#), [G137](#), [G155](#)
`\do@emfont@update` ... [z410](#), [z414](#), [z434](#)
`\do@noligs` [G577](#), [G582](#)
`\do@subst@correction` [u84](#), [w482](#), [w537](#)
`\document@default@language`
 . [q51](#), [q52](#), [q120](#), [q121](#), [V597](#), [X272](#)
`\document@select@group` ... [y137](#), [y268](#)
`\dont@add@percent@to@temptokena` .
 . [G130](#), [G132](#), [G156](#)
`\do@store@version` [y114](#), [y119](#)
`\ds@` [S459](#), [S974](#)
`\dt@pfalse` [H192](#)
`\dt@ptrue` [H191](#)
`\e@alloc` [b51](#), [b52](#), [b53](#),
 [b55](#), [b56](#), [b63](#), [b64](#), [b66](#), [b68](#), [b79](#),
 [b82](#), [b84](#), [b138](#), [b230](#), [d15](#), [d49](#), [d79](#),
 [d89](#), [d178](#), [d186](#), [d194](#), [d202](#), [X12](#), [X33](#)
`\e@alloc@attribute@count`
 . [d66](#), [d74](#), [d75](#), [d76](#), [d80](#), [d229](#)
`\e@alloc@bytecode@count`
 . [d70](#), [d189](#), [d190](#), [d191](#), [d195](#), [d245](#)
`\e@alloc@ccodetable@count`
 . [d67](#), [d84](#), [d85](#), [d86](#), [d90](#), [d233](#)
`\e@alloc@chardef` . [b60](#), [b102](#), [b210](#),
 [b211](#), [d48](#), [d178](#), [d186](#), [d194](#), [d202](#), [X12](#)
`\e@alloc@intercharclass@top` ... [X21](#)
`\e@alloc@luachunk@count`
 . [d71](#), [d197](#), [d198](#), [d199](#), [d203](#), [d247](#)
`\e@alloc@luafunction@count` . [d68](#),
 [d173](#), [d174](#), [d175](#), [d179](#), [d239](#), [d241](#)
`\e@alloc@top`
 . [b55](#), [b63](#), [b102](#), [b188](#), [b259](#),
 [d47](#), [d80](#), [d179](#), [d187](#), [d195](#), [d203](#), [X12](#)
`\e@alloc@whatsit@count`
 . [d69](#), [d181](#), [d182](#), [d183](#), [d187](#), [d243](#)
`\e@ch@ck` [b142](#), [b152](#), [d51](#), [d55](#)
`\e@insert@top` . [b257](#), [b259](#), [b276](#), [b291](#)
`\e@mathgroup@top`
 . [b79](#), [b124](#), [y56](#), [y145](#), [y176](#)
`\em@currfont` [498](#), [z409](#), [z420](#)
`\em@force` .. [498](#), [z418](#), [z423](#), [z426](#), [z436](#)
`\emfontdeclare@clist` [497](#),
 [z399](#), [z401](#), [z405](#), [z410](#), [z415](#), [z421](#), [z432](#)
`\empty@sfcnt`
 . [w490](#), [w491](#), [w492](#), [w506](#), [w511](#), [w563](#)
`\enc@update`
 . [u257](#), [u259](#), [u275](#), [u278](#), [w147](#), [w173](#)
`\end@dblfloat` [O205](#)
`\end@float` [733](#), [O189](#), [O227](#), [O243](#), [O383](#)
`\endlinechar` [243](#)
`\err@rel@i` [x12](#), [x99](#), [x132](#), [x136](#)
`\error@fontshape` [u482](#),
 [u497](#), [u522](#), [u557](#), [w107](#), [w527](#), [y254](#)
`\escapechar` [240](#)
`\et@xmaxfam` [d22](#), [d26](#), [d30](#), [d38](#)
`\et@xmaxregs`
 . [d29](#), [d31](#), [d32](#), [d33](#), [d34](#), [d35](#), [d36](#), [d37](#)
`\every@math@size` [u78](#), [w235](#), [w247](#)
`\execute@size@function`
 . [w362](#), [w390](#), [w404](#), [w421](#)
`\expand@font@defaults`
 . [185](#), [z83](#), [z190](#), [z200](#), [z220](#),
 [z270](#), [z281](#), [z292](#), [z325](#), [z354](#), [D26](#), [D83](#)
`expand@font@defaults` [z252](#)
`\external@font`
 . [w84](#), [w87](#), [w98](#), [w102](#), [w104](#), [w391](#),
 [w405](#), [w467](#), [w501](#), [w569](#), [w571](#), [w573](#)

File Key: a=ltldirchk.dtx, b=ltplain.dtx, c=ltvers.dtx, d=ltluatex.dtx, e=ltexpl.dtx,
 f=ltdefns.dtx, g=ltcmd.dtx, h=lt hooks.dtx, i=ltcmdhooks.dtx, j=ltalloc.dtx, k=ltcntrl.dtx,
 l=lterror.dtx, m=ltpar.dtx, n=lt para.dtx, o=lt space.dtx, p=ltlogos.dtx, q=ltfiles.dtx,
 r=ltoutenc.dtx, s=ltcounts.dtx, t=ltlength.dtx, u=ltfssbas.dtx, v=ltfssaxes.dtx,
 w=ltfssstrc.dtx, x=ltfsscmp.dtx, y=ltfssdcl.dtx, z=ltfssini.dtx, A=fontdef.dtx,
 B=preload.dtx, C=ltfntcmd.dtx, D=lttextcomp.dtx, E=ltpageno.dtx, F=ltxref.dtx,
 G=ltmiscen.dtx, H=ltmath.dtx, I=ltlists.dtx, J=ltboxes.dtx, K=lttab.dtx, L=lt pictur.dtx,
 M=ltthm.dtx, N=ltsect.dtx, O=ltfloat.dtx, P=ltidxglo.dtx, Q=ltbibl.dtx, R=lt page.dtx,
 S=ltclass.dtx, T=ltfilehook.dtx, U=ltshipout.dtx, V=ltoutput.dtx, W=lt hyphen.dtx,
 X=ltfinal.dtx

\extra@def	x9, x84	v676, v678, v713, w124, w128, w130, y272, z514, z532, z548, z593
\extract@alph@from@version	u597 , u603 , y151, y182, y214	\f@shape@saved
\extract@default@composite	r1053, r1060	\f@size r180, r869, r1211, u88, u317, u323, u362, u507, u546, u580, u622, u623, u626, u627, w121, w142, w167, w169, w180, w200, w215, w218, w221, w226, w233, w240, w252, w255, w261, w267, w284, w285, w288, w293, w359, w366, w385, w387, w402, w453, w455, w457, w473, w474, w479, w493, w505, w510, w522, w530, w535, w561, w575, z409, z418, D102, D931
\extract@default@composite@a ...	r1062, r1066	\f@user@size . w473, w478, w522, w535
\extract@default@composite@b ...	r1064, r1068	\famdef@ult
\extract@font	u378 , w81	\filec@ntents
\extract@fontinfo	w358, w365 S1050, S1053, S1056, S1067, S1088, S1181, S1184, S1187, S1198, S1219, S1312, S1334, S1422, S1637
\extract@rangefontinfo	w375 , w382, w401, w434	\filec@ntents@checkdir
\extract@sizefn	w350 , w372 S1073, S1075, S1089, S1204, S1206, S1220, S1319
\f@baselineskip	r870, r1212, u317, u324, u508, w121, w167, w182, w186, w201, w215, w226, w240	\filec@ntents@force
\f@depth	O291, V345 S1069, S1200, S1315
\f@encoding	d252, d267, d275, r178, u251 , u270, u273, u274, u276, u326, u358, u363, u382, u384, u386, u391, u393, u424, u486, u518, u553, v439, v443, v652, v656, w91, w128, w307, w517, y239	\filec@ntents@noheader
\f@family 385 , 489 , 495 , u279 , u287, u299, u309, u320, u359, u363, u382, u384, u386, u391, u393, u425, u504, u529, u564, v439, v443, v652, v656, w91, w128, y239, y270, z86, z113, z114, z210, z211, z212, z229, z230, z231, z282, z283, z284, z293, z294, z295, z357, z370, z512, z530, z546, D23, D27, D29, D31, D63, D66, D80, D84, D86, D88, D93, D100, D897, D900, D914, D923, D929, D1144	 S1071, S1202, S1317
\f@linespread	u320 , w120, w166, w183, w184, w187, w195, w198, w209, w212	\filec@ntents@nosearch
\f@series ..	403 , 412 , 413 , 415 , 419 , 428 , 429 , 485 – 487 , 491 , 494 , 496 , 503 , p14, u279 , u310, u321, u360, u363, u505, u526, u561, v400, v410, v419, v429, v463, v482, v483, v652, v656, w125, w128, w131, y271, z79, z95, z97, z101, z102, z103, z124, z130, z133, z135, z157, z373, z377, z513, z531, z547, z592, D7, D613 S1072, S1203, S1318
\f@series@saved	w125, w131	\filec@ntents@opt ..
\f@shape 420 , 428 , 503 , u279 , u289, u301, u311, u322, u361, u363, u506, u524, u559, v439, v443, v620, v627, v633, v643, v650, v654, v657, v660, v669,	S1053, S1056, S1058, S1184, S1187, S1189, S1314
		\filec@ntents@OPTION
		\filec@ntents@overwrite
	 S1070, S1201, S1316
		\filec@ntents@where S1074, S1076, S1101, S1205, S1207, S1232, S1320
		\filename@area
		a246, a252, a259, a265, a272, a278, a285, q544, q559, q569, q601, q602, q606, q631, q634, q651, q663, q665, S827
		\filename@base
		a295, a302, a315, q544, q559, q569, q601, q603, q606, q631, q634, q658, q663, S828
		\filename@dot
		a313, a319
		\filename@dots
		a297, a299, a304
		\filename@ext
		812 , a294, a301, a311, a313, q545, q560, q570, q597, q598, q601, q604, q606, q627, q628, q631, q634, q659, S829
		\filename@parse a110, a242 , q542, q557, q567, q596, q626, q656, S826
		\filename@path ..
		a247, a248, a253, a260, a261, a266, a273, a274, a279

File Key: a=ltldirchk.dtx, b=ltplain.dtx, c=ltvers.dtx, d=ltluatex.dtx, e=ltexpl.dtx, f=ltdefs.dtx, g=ltcmd.dtx, h=lthooks.dtx, i=ltcmdhooks.dtx, j=ltalloc.dtx, k=ltcntrl.dtx, l=lterror.dtx, m=ltpar.dtx, n=ltpara.dtx, o=ltspace.dtx, p=ltlogos.dtx, q=ltfiles.dtx, r=ltoutenc.dtx, s=ltcounts.dtx, t=ltlength.dtx, u=ltfssbas.dtx, v=ltfssaxes.dtx, w=ltfssstrc.dtx, x=ltfsscmp.dtx, y=ltfssdcl.dtx, z=ltfssini.dtx, A=fontdef.dtx, B=preload.dtx, C=ltfntcmd.dtx, D=lttextcomp.dtx, E=ltpageno.dtx, F=ltxref.dtx, G=ltmiscen.dtx, H=ltmath.dtx, I=ltlists.dtx, J=ltboxes.dtx, K=lttab.dtx, L=ltpictur.dtx, M=ltthm.dtx, N=ltsect.dtx, O=ltfloat.dtx, P=ltidxglo.dtx, Q=ltbibl.dtx, R=ltpage.dtx, S=ltclass.dtx, T=ltfilehook.dtx, U=ltshipout.dtx, V=ltoutput.dtx, W=lthyphen.dtx, X=ltfinal.dtx

\filename@simple a250, a263,
a276, a286, a290, a292, a307, a309
\finish@module@release c86, c90
\finph@nt
H104, H106, H110, H111, H119, H120
\finsh@sh
H134, H136, H140, H141, H145, H146
\fix@penalty C101
\fixed@sfcnt w565, w566, w567
\fl@trace V240, V267, V323,
V351, V358, V379, V426, V472,
V525, V540, V541, V542, V543,
V554, V555, V556, V557, V558,
V568, V781, V800, V819, V837,
V839, V978, V982, V994, V995,
V996, V997, V1003, V1006, V1014,
V1018, V1029, V1034, V1039,
V1040, V1041, V1042, V1049,
V1052, V1060, V1071, V1077,
V1082, V1087, V1093, V1094,
V1099, V1104, V1105, V1106,
V1114, V1118, V1123, V1127,
V1132, V1143, V1144, V1146,
V1164, V1173, V1179, V1188,
V1191, V1197, V1207, V1211,
V1221, V1227, V1233, V1239,
V1246, V1248, V1254, V1259,
V1261, V1263, V1271, V1276,
V1282, V1287, V1293, V1307,
V1308, V1311, V1332, V1341,
V1347, V1356, V1359, V1366,
V1376, V1380, V1392, V1398,
V1403, V1408, V1412, V1416,
V1417, V1424, V1429, V1433,
V1440, V1449, V1453, V1457,
V1458, V1462, V1463, V1473,
V1479, V1485, V1491, V1495,
V1501, V1503, V1511, V1516,
V1521, V1529, V1538, V1543,
V1548, V1550, V1555, V1557,
V1568, V1574, V1584, V1590,
V1594, V1595, V1600, V1601,
V1607, V1610, V1611, V1612,
V1619, V1620, V1621, V1629,
V1634, V1646, V1647, V1654,
V1657, V1665, V1669, V1673,
V1674, V1678, V1679, V1689,
V1695, V1705, V1711, V1715,
V1716, V1722, V1723, V1730,
V1733, V1734, V1735, V1743,
V1744, V1745, V1754, V1759,
V1772, V1774, V1781, V1784,
V1793, V1797, V1801, V1802,
V1806, V1807, V1859, V1864,
V1870, V1880, V1887, V1897,
V1993, V2006, V2007, V2011,
V2014, V2016, V2019, V2022,
V2024, V2065, V2072, V2077,
V2083, V2088, V2092, V2098,
V2106, V2108, V2115, V2120,
V2125, V2127, V2133, V2135,
V2142, V2171, V2173, V2188,
V2190, V2204, V2229, V2233,
V2238, V2250, V2267, V2272, V2280
\fl@tracemessage V1897
\fl@traceval V1897
\float@count
. b51, b52, b53, b62, b188,
b205, b210, b212, b213, b222, b230
\fmtversion@topatch
X578, X580, X592, X593, X605, X613
\font@info w99, w365, w434, w439
\font@name r179,
r182, u86, u194, u196, u354, u369,
u545, u579, w84, w88, w90, w105,
w141, w144, w154, w168, w171,
w330, w331, w332, w333, w334, w339
\font@submax w441, w470, w471,
G42, G44, G82, G84, X285, X287, X296
\fps@dbl O34
\frozen@everydisplay u344, u350
\frozen@everymath u344, u348
\g@addto@macro
. 192, f813, i41, v731, S131, S384,
S982, S998, S999, U352, U358, U399
\G@refundefinedfalse 573, F5
\G@refundefinedtrue
. F3, F12, Q21, Q48, Q65
\gen@sfcnt w502, w503, w504
\genb@sfcnt w507, w508, w509
\genb@x w510, w512
\genb@y w512
\get@cdp y388, y396, y429
\get@external@font w83, w96, w536
\getanddefine@fonts u592, u610,
w320, y59, y87, y132, y148, y179,
y210, y295, y359, y393, y395, y412,
y535, y536, y568, y569, y1011, y1012
\glb@currsize q41, q110, q168,
u341, w217, w252, w256, w262, w285
\glb@settings u342, w217, w264, w295
\gobble@finish@module@release
. c97, c99, c158
\group@elt y35,
y293, y330, y331, y352, y356, y1043
\group@list
. y297, y337, y350, y355, y356,

File Key: a=ltldirchk.dtx, b=ltplain.dtx, c=ltvers.dtx, d=ltluatex.dtx, e=ltexpl.dtx,
f=ltdefns.dtx, g=ltcmd.dtx, h=lt hooks.dtx, i=ltcmdhooks.dtx, j=ltalloc.dtx, k=ltcntrl.dtx,
l=lterror.dtx, m=ltpar.dtx, n=lt para.dtx, o=lt space.dtx, p=lt logos.dtx, q=lt files.dtx,
r=ltoutenc.dtx, s=lt counts.dtx, t=lt length.dtx, u=ltfssbas.dtx, v=ltfssaxes.dtx,
w=ltfssstrc.dtx, x=ltfsscmp.dtx, y=ltfssdcl.dtx, z=ltfssini.dtx, A=fontdef.dtx,
B=preload.dtx, C=ltfntcmd.dtx, D=lttextcomp.dtx, E=ltpageno.dtx, F=ltxref.dtx,
G=ltmiscen.dtx, H=ltmath.dtx, I=ltlists.dtx, J=ltboxes.dtx, K=lttab.dtx, L=lt pictur.dtx,
M=ltthm.dtx, N=ltsect.dtx, O=ltfloat.dtx, P=ltidxglo.dtx, Q=ltbibl.dtx, R=lt page.dtx,
S=ltclass.dtx, T=ltfilehook.dtx, U=ltshipout.dtx, V=ltoutput.dtx, W=lthyphen.dtx,
X=ltfinal.dtx

- y385, y611, y659, y720, y805, y808,
 y859, y862, y928, y931, y998, y1049
 \h@false H81
 \h@true H82, H83
 \hb@xt@ b472, [f29](#), r425,
 H197, H362, H424, H439, H451,
 H478, H508, J44, J65, J83, J205,
 J472, J476, J477, J478, K37, L31,
 L43, L62, L74, L105, L117, L265,
 L299, L302, L305, L307, L314,
 L373, L452, L588, L598, L741,
 L797, N203, N226, N233, V630,
 V640, V689, V699, V1843, V2223,
 V2224, V2228, V2255, V2256, V2262
 \hexnumber@ y632, y640, y675,
 y683, y704, y714, y740, y748, y756,
 y765, y768, y777, y778, y817, y825,
 y871, y879, y898, y899, y909, y910,
 y915, y941, y949, y954, y956, [z498](#)
 \hgl@ b427, b428
 \hmode@bgroup r67, [r75](#),
 r327, r356, r390, r396, r427, r438,
 r445, r476, r483, r486, r488, r496,
 r512, r730, r760, r766, r798, r805,
 r833, r836, r893, r1245, C7, D625, D632
 \hmode@start@before@group
 r68, r151, r153, r159, [r184](#)
 \if@afterindent [N124](#), [N131](#)
 \if@compatibility [S2](#), [S597](#)
 \if@endpe [G214](#), [G221](#), [G273](#), [G288](#), [I138](#)
 \if@eqnsw [H344](#), [H409](#)
 \if@fcolmade
 .. [V95](#), [V264](#), [V394](#), [V403](#), [V441](#),
[V451](#), [V779](#), [V799](#), [V817](#), [V846](#),
[V926](#), [V2170](#), [V2187](#), [V2237](#), [V2277](#)
 \if@filesw [q5](#), [q36](#), [q105](#),
[q163](#), [q296](#), [q308](#), [q329](#), [q343](#), [q355](#),
[q362](#), [q374](#), [G21](#), [G48](#), [G74](#), [G88](#),
[N153](#), [Q4](#), [Q8](#), [Q19](#), [Q28](#), [Q36](#), [Q47](#),
[Q64](#), [S1099](#), [S1116](#), [S1230](#), [S1249](#), [U361](#)
 \if@firstamp [K251](#)
 \if@firstcolumn [V95](#), [V246](#), [V279](#),
[V396](#), [V444](#), [V1815](#), [V2201](#), [V2246](#)
 \if@font@series@context
 [495](#), [z362](#), [z381](#), [z389](#)
 \if@forced@series [413](#), [v401](#), [z78](#)
 \if@ignore [G4](#), [G216](#), [G223](#), [G274](#), [G289](#)
 \if@includeinrelease
 [c70](#), [c73](#), [c120](#), [f767](#)
 \if@inlabel
[V161](#), [V188](#), [I28](#), [I65](#), [I102](#), [I160](#), [I183](#)
 \if@insert
[V95](#), [V1057](#), [V1169](#), [V1203](#), [V1337](#),
[V1372](#), [V1446](#), [V1535](#), [V1662](#), [V1790](#)
 \if@minipage [o239](#), [o257](#), [o276](#), [o311](#),
[G418](#), [G440](#), [J323](#), [K79](#), [O20](#), [I149](#)
 \if@mparswitch [V95](#), [V1817](#)
 \if@multiplelabels [F31](#)
 \if@negarg [L157](#), [L198](#), [L212](#), [L273](#)
 \if@newlist [G460](#), [V599](#), [V644](#), [V657](#),
[V703](#), [I29](#), [I33](#), [I69](#), [I78](#), [I106](#), [I166](#)
 \if@nmbrlist [I33](#), [I201](#)
 \if@no@font@opt [x16](#), [x110](#), [x129](#)
 \if@nobreak [o120](#), [o147](#), [o278](#), [o313](#),
[q202](#), [q214](#), [J286](#), [J307](#), [N47](#), [N128](#),
[O180](#), [O373](#), [R29](#), [R37](#), [V165](#),
[V192](#), [V335](#), [V1148](#), [V1314](#), [I167](#), [I192](#)
 \if@noitemarg [I32](#), [I199](#)
 \if@noparitem [I30](#), [I157](#)
 \if@noparlist [I31](#), [I114](#)
 \if@noskipsec .. [o147](#), [J287](#), [J308](#),
[N38](#), [N40](#), [N97](#), [O374](#), [V155](#), [V182](#), [I58](#)
 \if@ovb
 . [L427](#), [L495](#), [L526](#), [L551](#), [L562](#), [L575](#)
 \if@ovhline [L459](#), [L590](#)
 \if@ovl
 . [L427](#), [L493](#), [L522](#), [L547](#), [L592](#), [L601](#)
 \if@ovr
 . [L427](#), [L492](#), [L521](#), [L546](#), [L589](#), [L599](#)
 \if@ovt
 . [L427](#), [L494](#), [L525](#), [L550](#), [L567](#), [L579](#)
 \if@ovvline [L459](#), [L565](#)
 \if@partsw [q5](#), [q300](#), [q347](#)
 \if@pboxsw [J278](#), [J406](#)
 \if@reversemargin [V101](#), [V1820](#)
 \if@reversemarginpar [V95](#)
 \if@rjfield [K19](#), [K33](#)
 \if@skipping@module . [c125](#), [c137](#), [c140](#)
 \if@specialpage ... [V95](#), [V606](#), [V664](#)
 \if@tempsta ... [a78](#), [a79](#), [a80](#), [b270](#),
[j9](#), [q306](#), [q353](#), [r1534](#), [u99](#), [u687](#),
[y318](#), [y373](#), [y437](#), [y518](#), [y1042](#), [G50](#),
[G90](#), [G425](#), [G446](#), [Q75](#), [S1128](#),
[S1261](#), [S1350](#), [T283](#), [T284](#), [T304](#),
[T305](#), [V990](#), [V1026](#), [V1626](#), [V1751](#)
 \if@test [V12](#),
[V13](#), [V887](#), [V906](#), [V946](#), [V968](#),
[V1032](#), [V1116](#), [V1125](#), [V1274](#),
[V1285](#), [V1427](#), [V1514](#), [V1632](#), [V1757](#)
 \if@twocolumn [q26](#), [q95](#),
[q152](#), [O32](#), [O210](#), [O235](#), [V95](#), [V139](#),
[V267](#), [V278](#), [V395](#), [V443](#), [V467](#),
[V781](#), [V837](#), [V1814](#), [V2172](#), [V2189](#)
 \if@twoside ... [V95](#), [V138](#), [V609](#), [V667](#)
 \ifdt@p [H190](#), [H192](#)
 \ifFileExists@ [q407](#), [q434](#)
 \ifG@refundefined [F3](#), [F4](#), [F5](#)
 \ifh@ [H76](#), [H114](#), [H123](#)

File Key: a=ltldirchk.dtx, b=ltplain.dtx, c=ltvers.dtx, d=ltluatex.dtx, e=ltexpl.dtx,
 f=ltdefs.dtx, g=ltcmd.dtx, h=lt hooks.dtx, i=ltcmdhooks.dtx, j=ltalloc.dtx, k=ltcntrl.dtx,
 l=ltterror.dtx, m=ltpar.dtx, n=lt para.dtx, o=lt space.dtx, p=ltlogos.dtx, q=ltfiles.dtx,
 r=ltoutenc.dtx, s=ltcounts.dtx, t=ltlength.dtx, u=ltfssbas.dtx, v=ltfssaxes.dtx,
 w=ltfssstrc.dtx, x=ltfsscmp.dtx, y=ltfssdcl.dtx, z=ltfssini.dtx, A=fontdef.dtx,
 B=preload.dtx, C=ltfntcmd.dtx, D=lttextcomp.dtx, E=ltpageno.dtx, F=ltxref.dtx,
 G=ltmiscen.dtx, H=ltmath.dtx, I=ltlists.dtx, J=ltboxes.dtx, K=lttab.dtx, L=lt pictur.dtx,
 M=ltthm.dtx, N=ltsect.dtx, O=ltfloat.dtx, P=ltidxglo.dtx, Q=ltbibl.dtx, R=ltpage.dtx,
 S=ltclass.dtx, T=ltfilehook.dtx, U=ltshipout.dtx, V=ltoutput.dtx, W=lthyphen.dtx,
 X=ltfinal.dtx

`\ifin@` [r1550](#), [r1553](#),
[x50](#), [x52](#), [y1](#), [y22](#), [y282](#), [y384](#), [y386](#),
[y447](#), [y460](#), [y530](#), [y532](#), [y560](#), [y612](#),
[y626](#), [y660](#), [y672](#), [y721](#), [y737](#), [y806](#),
[y809](#), [y830](#), [y860](#), [y863](#), [y926](#), [y929](#),
[y932](#), [y999](#), [y1001](#), [y1030](#), [z145](#),
[z149](#), [z153](#), [S241](#), [S259](#), [S469](#), [S503](#)
`\ifmath@fonts` [u204](#), [w222](#)
`\ifmaybe@ic` [C82](#), [C91](#)
`\ifnot@nil` [w343](#), [w360](#), [w381](#)
`\iftc@forced` [D871](#), [D882](#), [D1150](#)
`\ifv@` [H75](#), [H113](#), [H122](#)
`\ifx` [243](#)
`\in@` [416](#), [r1548](#), [r1551](#),
[x49](#), [x51](#), [y1](#), [y21](#), [y281](#), [y383](#), [y385](#),
[y443](#), [y456](#), [y529](#), [y531](#), [y558](#), [y610](#),
[y621](#), [y658](#), [y669](#), [y719](#), [y733](#), [y804](#),
[y807](#), [y827](#), [y858](#), [y861](#), [y923](#), [y927](#),
[y930](#), [y997](#), [y1000](#), [y1028](#), [z143](#),
[z147](#), [z151](#), [S240](#), [S257](#), [S466](#), [S502](#)
`\in@@`
[v489](#), [v490](#), [v492](#), [v493](#), [y5](#), [y6](#), [y7](#), [y9](#)
`\in@false` [y10](#)
`\in@true` [y12](#)
`\init@restore@glb@settings`
..... [w265](#), [w268](#), [w270](#)
`\init@restore@version`
..... [y62](#), [y91](#), [y108](#), [y123](#), [y124](#)
`\init@series@setup`
..... [v732](#), [v737](#), [z141](#), [z182](#)
`\input@path` [10](#), [804](#),
[812](#), [818](#), [a109](#), [a131](#), [a133](#), [a139](#),
[a141](#), [a147](#), [a149](#), [a154](#), [a156](#), [a166](#),
[a233](#), [q420](#), [q441](#), [q462](#), [q489](#), [q506](#)
`\insscunt` [b37](#), [b51](#), [b52](#), [b53](#),
[b62](#), [b90](#), [b91](#), [b92](#), [b94](#), [b247](#), [b248](#),
[b249](#), [b250](#), [b251](#), [b252](#), [b263](#), [b264](#),
[b265](#), [b266](#), [b267](#), [b271](#), [b273](#), [b292](#),
[b293](#), [b294](#), [b295](#), [b296](#), [b297](#), [V61](#)
`\install@mathalphabet`
..... [u587](#), [u604](#), [u611](#), [y301](#),
[y304](#), [y390](#), [y391](#), [y488](#), [y540](#), [y543](#),
[y550](#), [y565](#), [y566](#), [y573](#), [y1013](#), [y1015](#)
`\is@range` [w376](#), [w377](#)
`\kernel@ifnextchar`
. [c87](#), [f81](#), [f100](#), [f150](#), [f678](#), [f693](#), [S365](#)
`\kernel@make@fragile`
..... [f375](#), [f794](#), [f795](#),
[f796](#), [f797](#), [f798](#), [f799](#), [f800](#), [f801](#),
[f802](#), [f803](#), [f804](#), [f805](#), [f806](#), [f807](#),
[f808](#), [f809](#), [o24](#), [o25](#), [o26](#), [o27](#), [o28](#),
[r172](#), [r173](#), [G394](#), [G395](#), [G396](#), [H90](#),
[H91](#), [H92](#), [H93](#), [H166](#), [H167](#), [H168](#),
[K160](#), [K161](#), [K162](#), [L823](#), [L824](#),
[L825](#), [L826](#), [L827](#), [L828](#), [L829](#),
[L830](#), [L831](#), [L832](#), [L833](#), [L834](#),
[N23](#), [N24](#), [N25](#), [N26](#), [N27](#), [R43](#), [R44](#)
`\l@ngrel@x` [f74](#), [f75](#), [f76](#), [f120](#), [f167](#)
`\l@nohyphenation` .. [G422](#), [G562](#), [X269](#)
`\last@fontshape` [u520](#), [u538](#), [u555](#), [u572](#)
`\latexrelease@postexpl` [g2269](#)
`\leavevmode@ifvmode` [o461](#),
[o462](#), [o470](#), [A624](#), [A626](#), [A628](#),
[A630](#), [H115](#), [H141](#), [H206](#), [H226](#), [H227](#)
`\load@onefile@withoptions`
..... [779](#), [S831](#), [S856](#), [S961](#)
`\load@onefilewithoptions`
..... [782](#), [S778](#), [S911](#), [S1437](#)
`\lower@bound` [w386](#), [w387](#), [w398](#)
`\ltx@sh@ft`
..... [b464](#), [r390](#), [r397](#), [r476](#), [r484](#), [r760](#), [r767](#)
`\m@ne` [246](#), [b39](#)
`\m@th` [b444](#), [b456](#), [p13](#), [A337](#),
[A459](#), [A461](#), [A462](#), [A465](#), [A506](#),
[A530](#), [A533](#), [A537](#), [A540](#), [A547](#),
[A550](#), [A557](#), [A560](#), [A642](#), [H68](#), [H71](#),
[H106](#), [H136](#), [H154](#), [H156](#), [H172](#),
[H191](#), [H353](#), [H439](#), [H451](#), [H478](#),
[H488](#), [J278](#), [J432](#), [K181](#), [N199](#),
[N222](#), [O400](#), [O409](#), [O416](#), [O437](#), [O444](#)
`\makeph@nt` [H101](#), [H103](#)
`\makesm@sh` [H131](#), [H133](#)
`\mandatory@arg` [w414](#), [w501](#), [w505](#),
[w510](#), [w517](#), [w519](#), [w524](#), [w526](#),
[w531](#), [w533](#), [w546](#), [w562](#), [w569](#), [w571](#)
`\math@bgroup` [459](#),
[u618](#), [w306](#), [w312](#), [y53](#), [y81](#), [y142](#),
[y164](#), [y173](#), [y204](#), [C130](#), [C131](#), [C138](#)
`\math@egroup`
..... [u618](#), [w310](#), [w311](#), [C131](#), [C132](#), [C139](#)
`\math@fonts` [u588](#), [u593](#),
[w232](#), [w336](#), [y60](#), [y89](#), [y149](#), [y180](#), [y212](#)
`\math@fontsfalse`
. [p7](#), [r302](#), [r329](#), [r360](#), [r1247](#), [u77](#),
[u206](#), [u216](#), [u239](#), [D103](#), [D673](#), [D932](#)
`\math@fontstrue` [u204](#), [u630](#)
`\math@version`
..... [u7](#), [u336](#), [u592](#), [u596](#), [u598](#), [u599](#),
[u601](#), [w230](#), [y56](#), [y59](#), [y64](#), [y65](#),
[y69](#), [y84](#), [y88](#), [y93](#), [y94](#), [y98](#), [y111](#),
[y112](#), [y113](#), [y126](#), [y127](#), [y128](#), [y145](#),
[y148](#), [y152](#), [y154](#), [y156](#), [y160](#), [y176](#),
[y179](#), [y183](#), [y185](#), [y187](#), [y191](#), [y207](#),
[y211](#), [y215](#), [y217](#), [y219](#), [y223](#), [z470](#)
`\mathchar@type` . [y704](#), [y714](#), [y765](#),
[y768](#), [y777](#), [y793](#), [y898](#), [y909](#), [y972](#)
`\mathph@nt` [H99](#), [H105](#)
`\mathsm@sh` [H129](#), [H135](#)

File Key: a=ltldirchk.dtx, b=ltplain.dtx, c=ltvers.dtx, d=ltluatex.dtx, e=ltexpl.dtx,
f=ltdefns.dtx, g=ltcmd.dtx, h=lt hooks.dtx, i=ltcmdhooks.dtx, j=ltalloc.dtx, k=ltcntrl.dtx,
l=lterror.dtx, m=ltpar.dtx, n=lt para.dtx, o=lt space.dtx, p=ltlogos.dtx, q=ltfiles.dtx,
r=ltoutenc.dtx, s=ltcounts.dtx, t=ltlength.dtx, u=ltfssbas.dtx, v=ltfssaxes.dtx,
w=ltfssstrc.dtx, x=ltfsscmp.dtx, y=ltfssdcl.dtx, z=ltfssini.dtx, A=fontdef.dtx,
B=preload.dtx, C=ltfntcmd.dtx, D=lttextcomp.dtx, E=ltpageno.dtx, F=ltxref.dtx,
G=ltmiscen.dtx, H=ltmath.dtx, I=ltlists.dtx, J=ltboxes.dtx, K=lttab.dtx, L=lt pictur.dtx,
M=ltthm.dtx, N=ltsect.dtx, O=ltfloat.dtx, P=ltidxglo.dtx, Q=ltbibl.dtx, R=ltpage.dtx,
S=ltclass.dtx, T=ltfilehook.dtx, U=ltshipout.dtx, V=ltoutput.dtx, W=lthyphen.dtx,
X=ltfinal.dtx

<code>\maybe@ic</code>	C63 , C64 , C83
<code>\maybe@ic@</code>	C83
<code>\maybe@icfalse</code>	C97
<code>\maybe@ictrue</code>	C87
<code>\maybe@load@fontshape</code>	327 , r71 , v476 , v515 , w127 , z99
<code>\mb@b</code>	J55 , J66 , J74 , J84
<code>\mb@l</code>	J55 , J59 , J65 , J74 , J78 , J83 , L137 , L141
<code>\mb@r</code>	J55 , J59 , J65 , J74 , J78 , J83 , L137 , L141
<code>\mb@t</code>	J56 , J63 , J75 , J82
<code>\md@def@ult</code>	z190
<code>\mddef@ult</code>	z133 , z195 , z224 , z225 , z226 , z275
<code>\mddefault@previous</code>	z221 , z223 , A107 , A117
<code>\mdseries@..</code>	483 , 488
<code>\mdseries@rm</code>	z63 , z173 , z224 , z229 , z293
<code>\mdseries@sf</code>	z63 , z174 , z225 , z230 , z294
<code>\mdseries@tt</code>	z63 , z175 , z226 , z231 , z295
<code>\meaning</code>	234 , 240 , 241 , 243 , 244
<code>\merge@font@series</code>	412 – 414 , v408 , v425 , v426 , v507 , v509 , w133
<code>\merge@font@series@</code> ..	v428 , v433 , v510
<code>\merge@font@series@without@substitution</code>	414 , v460 , v475 , v512 , w136
<code>\merge@font@series@without@substitution@</code>	v460 , v513
<code>\merge@font@shape</code>	v626 , v639 , v640 , v711 , v716 , w132
<code>\merge@font@shape@</code> ..	v642 , v647 , v717
<code>\merge@font@shape@without@substitution</code>	v666 , v719 , w135
<code>\merge@font@shape@without@substitution@</code>	v666 , v720
<code>\n@space</code>	A625 , A627 , A629 , A631 , A636 , A637 , A638 , A639 , A642
<code>\new@command</code>	f77 , f78 , f131 , f165 , f184 , f239
<code>\new@environment</code>	f146 , f147 , f159
<code>\new@fontshape</code>	x2 , x4 , x22 , x24
<code>\new@mathalphabet</code> ...	y441 , y462 , y473
<code>\new@mathgroup</code>	b78 , b80 , b98 , b100 , d27 , u14 , y321
<code>\new@mathversion</code>	y20 , y278 , y280
<code>\new@module@skip</code>	c126 , c138 , c140
<code>\new@moduledate</code>	c79 , c92 , c140
<code>\new@modulename</code>	c140
<code>\new@symbolfont</code>	y322 , y354
<code>\newcommand</code>	232 , 234 , 238 , 239
<code>\NewCommandCopy</code>	238
<code>\NewDocumentCommand</code>	232
<code>\newlinechar</code>	243
<code>\newmathalphabet@</code>	x14
<code>\newmathalphabet@@</code>	x109
<code>\newmathalphabet@@@</code>	x15 , x109
<code>\nfss@catcodes</code>	392 , u19 , u120 , u387 , u388 , u395 , u442 , A40 , A45 , A135 , V3
<code>\nfss@text</code>	r315 , r317 , z501 , C5 , C122 , F13
<code>\no@alphabet@error</code> ..	u4 , y300 , y302 , y478 , y479 , y493 , y502 , y588 , y589
<code>\noaccents@</code>	u633 , A129
<code>\noexpand</code>	244
<code>\non@alpherr</code>	u612 , u614 , y72 , y101 , y117 , y163 , y194 , y226 , y1050
<code>\not@base</code>	z573 , z577 , z578 , z579 , z580 , z581 , z582 , z583 , z584 , z585 , z586 , z587
<code>\not@math@alphabet</code>	v527 , v532 , v537 , v683 , v687 , v690 , v693 , v696 , v699 , v702 , v705 , z5 , z8 , z11 , z14 , z17 , z20 , z23 , z26 , z29 , z199 , z219 , z238 , z243 , z248 , z280 , z291 , z302 , z307 , z312 , z328 , z331 , z334 , z337 , z340 , z450
<code>\now@and@everyjob</code> ..	d207 , d213 , r1011
<code>\o@lign</code>	b458 , r390 , r397 , r476 , r484 , r760 , r767
<code>\on@line</code>	270 , h294 , h565 , l8 , l15 , l165 , z79 , z82 , G177 , G232 , G249 , G282 , J150 , S839 , S950
<code>\operator@font</code>	A643 , H3 , H4 , H5 , H6 , H7 , H8 , H9 , H10 , H11 , H12 , H13 , H14 , H15 , H16 , H17 , H18 , H19 , H20 , H21 , H22 , H23 , H24 , H25 , H26 , H27 , H28 , H29 , H30 , H31 , H32 , H33 , H34 , H37 , H40
<code>\optional@arg</code>	w415 , w494 , w496 , w568 , w571
<code>\outer@nobreak</code>	O181 , O251 , O255 , O346 , O364
<code>\p@</code>	b311
<code>\p@enum</code>	626
<code>\p@equation</code>	H351 , H487
<code>\p@selectfont</code>	w119
<code>\par</code>	142
<code>\par@deathcycles</code>	I56 , I77 , I79 , I80
<code>\patch@level</code>	36 , c1 , c36 , c41 , c43 , c45 , c49 , c58 , X581 , X593 , X595
<code>\patchcmd</code>	232
<code>\ph@nt</code>	H81 , H82 , H83 , H97
<code>\pickup@font</code>	r181 , u195 , u353 , u547 , u581 , w143 , w170 , w331 , w333 , w335
<code>\pictur@</code>	L21
<code>\pkgcls@arg</code>	S1448 , S1571

File Key: a=ltldirchk.dtx, b=ltplain.dtx, c=ltvers.dtx, d=ltluatex.dtx, e=ltexpl.dtx, f=ltdefns.dtx, g=ltcmd.dtx, h=lt hooks.dtx, i=ltcmdhooks.dtx, j=ltalloc.dtx, k=ltcntrl.dtx, l=lterror.dtx, m=ltpar.dtx, n=lt para.dtx, o=lt space.dtx, p=ltlogos.dtx, q=ltfiles.dtx, r=ltoutenc.dtx, s=ltcounts.dtx, t=ltlength.dtx, u=ltfssbas.dtx, v=ltfssaxes.dtx, w=ltfssstrc.dtx, x=ltfsscmp.dtx, y=ltfssdcl.dtx, z=ltfssini.dtx, A=fontdef.dtx, B=preload.dtx, C=ltfntcmd.dtx, D=lttextcomp.dtx, E=ltpageno.dtx, F=ltxref.dtx, G=ltmiscen.dtx, H=ltmath.dtx, I=ltlists.dtx, J=ltboxes.dtx, K=lttab.dtx, L=lt pictur.dtx, M=ltthm.dtx, N=ltsect.dtx, O=ltfloat.dtx, P=ltidxglo.dtx, Q=ltbibl.dtx, R=lt page.dtx, S=ltclass.dtx, T=ltfilehook.dtx, U=ltshipout.dtx, V=ltoutput.dtx, W=lthyphen.dtx, X=ltfinal.dtx

- \pkgcls@candidate .. [S1435](#), [S1450](#),
[S1526](#), [S1530](#), [S1534](#), [S1602](#), [S1605](#)
- \pkgcls@debug [S1425](#), [S1441](#), [S1442](#),
[S1443](#), [S1444](#), [S1445](#), [S1502](#), [S1503](#),
[S1504](#), [S1505](#), [S1514](#), [S1519](#), [S1537](#),
[S1546](#), [S1561](#), [S1595](#), [S1596](#), [S1597](#)
- \pkgcls@innerdate .. [S1430](#), [S1475](#), [S1478](#), [S1484](#), [S1623](#)
- \pkgcls@mindate .. [796](#), [S1455](#), [S1464](#), [S1480](#), [S1485](#)
- \pkgcls@name .. [S1447](#), [S1490](#)
- \pkgcls@parse@date@arg . [S1449](#), [S1460](#)
- \pkgcls@parse@date@arg@ [S1466](#), [S1469](#)
- \pkgcls@parse@date@arg@version ..
..... [S1476](#), [S1497](#)
- \pkgcls@releasedate ..
..... [S1435](#), [S1531](#), [S1535](#), [S1606](#)
- \pkgcls@rollbackdate@error ..
..... [S1527](#), [S1585](#), [S1603](#)
- \pkgcls@show@selection ..
..... [S1554](#), [S1559](#), [S1609](#), [S1614](#)
- \pkgcls@targetdate ..
..... [796](#), [S1430](#), [S1462](#), [S1470](#),
[S1473](#), [S1474](#), [S1478](#), [S1486](#), [S1487](#),
[S1500](#), [S1508](#), [S1523](#), [S1525](#), [S1555](#),
[S1566](#), [S1568](#), [S1593](#), [S1599](#), [S1601](#)
- \pkgcls@targetlabel ..
..... [796](#), [S1430](#), [S1463](#), [S1483](#), [S1498](#),
[S1510](#), [S1542](#), [S1575](#), [S1613](#), [S1616](#)
- \pkgcls@use@this@release . [S1511](#),
[S1528](#), [S1530](#), [S1543](#), [S1553](#), [S1605](#)
- \pr@@s .. [H246](#), [H254](#)
- \pr@@t .. [H249](#), [H255](#)
- \pr@m@s .. [H243](#), [H244](#)
- \preload@sizes .. [x11](#), [x94](#)
- \prepare@family@series@update ...
..... [492](#), [z76](#),
[z179](#), [z239](#), [z244](#), [z249](#), [z303](#), [z308](#), [z313](#)
- \pretocmd .. [231](#)
- \prim@s .. [H240](#), [H242](#), [H254](#)
- \prime@s .. [H241](#)
- \process@table . [q40](#), [q109](#), [q167](#), [y232](#)
- \protected .. [220](#)
- \protected@edef .. [335](#), [476](#),
[940](#), [f253](#), [s158](#), [z399](#), [F43](#), [F47](#),
[F55](#), [F59](#), [F66](#), [J379](#), [J397](#), [N60](#),
[O471](#), [O490](#), [X558](#), [X567](#), [X572](#), [X573](#)
- \protected@file@percent ..
..... [582](#), [715](#), [G122](#),
[G129](#), [G144](#), [G152](#), [G153](#), [N165](#), [N172](#)
- \protected@wlog [S311](#), [S313](#), [S327](#), [S341](#)
- \protected@write ..
... [q201](#), [q206](#), [F33](#), [N181](#), [P14](#), [P31](#)
- \protected@xdef .. [f253](#),
[N11](#), [O453](#), [O503](#), [O519](#), [S303](#), [S322](#)
- \provide@command .. [f178](#), [f179](#)
- \ps@empty .. [R10](#), [X144](#)
- \ps@plain .. [R13](#)
- \q@curr@file ..
.... [S1050](#), [S1090](#), [S1092](#), [S1097](#),
[S1123](#), [S1221](#), [S1223](#), [S1228](#), [S1256](#)
- \quote@name .. [q398](#), [q414](#)
- \quote@name .. [q398](#),
[q411](#), [q413](#), [q490](#), [q492](#), [q501](#), [S1221](#)
- \r@@t .. [H66](#)
- \reenable@package@load ..
..... [807](#), [821](#), [849](#), [T442](#), [U455](#)
- \reinstall@nfss@defs .. [v685](#),
[v726](#), [v730](#), [v732](#), [v736](#), [v737](#), [v740](#)
- \relax .. [244](#)
- \rem@pt .. [u329](#)
- \remove@angles .. [w347](#), [w370](#)
- \remove@nil .. [y36](#)
- \remove@star .. [w347](#), [w353](#)
- \remove@tlig .. [r1001](#), [r1003](#),
[r1005](#), [r1030](#), [r1035](#), [r1071](#), [r1073](#), [r1075](#)
- \remove@to@nnil [u328](#), [w347](#), [w373](#), [w486](#)
- \renew@command .. [f124](#), [f125](#), [f185](#), [f193](#)
- \renew@environment .. [f152](#), [f153](#)
- \requested@test@context ..
..... [495](#), [z356](#), [z373](#), [z377](#)
- \reserved@b .. [472](#)
- \reserved@a ..
.. [421](#), [495](#), [a121](#), [a125](#), [a126](#), [a195](#),
[a196](#), [a199](#), [a217](#), [a221](#), [a243](#), [a250](#),
[a253](#), [a255](#), [a256](#), [a263](#), [a266](#), [a268](#),
[a269](#), [a276](#), [a279](#), [a281](#), [a331](#), [a332](#),
[a333](#), [b193](#), [c13](#), [c19](#), [c34](#), [e22](#), [e23](#),
[e24](#), [e25](#), [e26](#), [e47](#), [e52](#), [e89](#), [e95](#),
[e105](#), [f117](#), [f120](#), [f133](#), [f134](#), [f135](#),
[f137](#), [f184](#), [f185](#), [f186](#), [f192](#), [f193](#),
[f194](#), [f195](#), [f198](#), [f218](#), [f226](#), [f230](#),
[f288](#), [f292](#), [f326](#), [f330](#), [f354](#), [f358](#),
[f469](#), [f477](#), [f478](#), [f526](#), [f529](#), [f544](#),
[f546](#), [f549](#), [f558](#), [f579](#), [f586](#), [f608](#),
[f611](#), [f675](#), [f684](#), [k33](#), [k37](#), [l185](#),
[o409](#), [o412](#), [q211](#), [q212](#), [q234](#), [q243](#),
[q252](#), [q304](#), [q351](#), [q421](#), [q423](#), [q428](#),
[q430](#), [q442](#), [q444](#), [q449](#), [q451](#), [q452](#),
[q453](#), [q463](#), [q465](#), [q470](#), [q472](#), [q487](#),
[q493](#), [q497](#), [q504](#), [q510](#), [q514](#), [q543](#),
[q546](#), [q547](#), [q549](#), [q558](#), [q561](#), [q568](#),
[q571](#), [q593](#), [q594](#), [q595](#), [q599](#), [q607](#),
[q624](#), [q625](#), [q629](#), [q635](#), [q657](#), [q661](#),
[q665](#), [r81](#), [r82](#), [r86](#), [r89](#), [r97](#), [r107](#),
[r110](#), [r119](#), [r138](#), [r143](#), [r995](#), [r999](#),
[r1049](#), [r1058](#), [u30](#), [u31](#), [u32](#), [u41](#),

File Key: a=ltldirchk.dtx, b=ltplain.dtx, c=ltvers.dtx, d=ltluatex.dtx, e=ltexpl.dtx, f=ltdefns.dtx, g=ltcmd.dtx, h=lt hooks.dtx, i=ltcmdhooks.dtx, j=ltalloc.dtx, k=ltcntrl.dtx, l=lterror.dtx, m=ltpar.dtx, n=lt para.dtx, o=lt space.dtx, p=ltlogos.dtx, q=ltfiles.dtx, r=ltoutenc.dtx, s=ltcounts.dtx, t=ltlength.dtx, u=ltfssbas.dtx, v=ltfssaxes.dtx, w=ltfssstrc.dtx, x=ltfsscmp.dtx, y=ltfssdcl.dtx, z=ltfssini.dtx, A=fontdef.dtx, B=preload.dtx, C=ltfntcmd.dtx, D=lttextcomp.dtx, E=ltpageno.dtx, F=ltxref.dtx, G=ltmiscen.dtx, H=ltmath.dtx, I=ltlists.dtx, J=ltboxes.dtx, K=lttab.dtx, L=lt pictur.dtx, M=ltthm.dtx, N=ltsect.dtx, O=ltfloat.dtx, P=ltidxglo.dtx, Q=ltbibl.dtx, R=lt page.dtx, S=ltclass.dtx, T=ltfilehook.dtx, U=ltshipout.dtx, V=ltoutput.dtx, W=lt hyphen.dtx, X=ltfinal.dtx

u44, u47, u63, u66, u69, u105, u108,
 u110, u147, u151, u389, u392, u519,
 u520, u535, u538, u543, u554, u555,
 u568, u572, u577, u604, u607, u608,
 u616, v435, v436, v439, v440, v455,
 v456, v468, v469, v648, v649, v652,
 v653, v674, v675, w196, w198,
 w200, w210, w212, w215, w344,
 w345, w358, w359, x53, x57, y388,
 y397, y399, y443, y446, y456, y459,
 y557, y559, y621, y625, y669, y671,
 y732, y735, y827, y829, y923, y925,
 y1027, y1029, y1045, y1047, y1048,
 y1053, z37, z38, z109, z110, z369,
 z370, C47, C48, C53, C54, C65,
 C68, C88, C95, G120, G121, G175,
 G176, G181, G230, G231, G235,
 G247, G248, G252, G280, G281,
 G285, G296, G297, H404, H405,
 H406, H407, H409, J57, J58, J61,
 J76, J77, J80, J145, J151, K241,
 K245, K250, K269, K360, K361,
 L199, L201, L205, L478, L479,
 L506, L507, L535, L536, O29, O30,
 O32, O33, O63, O67, O72, O74,
 O76, O78, O83, O84, O132, O136,
 O142, O145, O148, O151, S235,
 S243, S247, S253, S261, S265, S373,
 S375, S376, S377, S381, S394, S396,
 S397, S398, S402, S577, S581, S587,
 S591, S653, S654, S657, S699, S703,
 S715, S716, S718, S727, S731, S743,
 S744, S746, S754, S758, S770, S771,
 S772, S774, S783, S786, S787, S789,
 S859, S905, S918, S959, S1061,
 S1062, S1064, S1192, S1193, S1195,
 S1237, S1238, S1240, S1244, S1452,
 S1457, S1509, S1510, S1541, S1542,
 S1612, S1613, S1637, S1639, T161,
 T167, T168, T169, V37, V46, V48,
 V50, V877, V897, V1963, V1965,
 V1966, V2055, V2057, V2063,
 V2066, X212, X229, X230, X231,
 X238, X239, X240, X477, X480,
 X511, X517, X518, X529, X530,
 X531, X538, X539, X540, X553,
 X554, X558, X559, X562, X563,
 X567, X568, X594, X597, X598, X615
 \reserved@b
 .. a122, a123, e48, e54, f109, f111,
 f118, f135, f136, f227, f228, f230,
 f289, f290, f292, f327, f328, f330,
 f355, f356, f358, f527, f529, f545,
 f549, f583, f586, f676, f686, k33,
 k34, k37, o410, o411, o418, q302,
 q304, q349, q351, q488, q490, q492,
 q505, q507, q509, q593, q594, q595,
 q660, q666, r90, r97, r112, r119,
 r998, r999, r1058, r1067, r1069,
 u31, u32, u38, u95, u97, u150,
 u151, u605, u616, v454, v455, v457,
 x47, x54, x71, x73, y314, y316,
 y369, y371, y396, y397, y398, y433,
 y435, y514, y516, y561, y562, y563,
 y570, y730, y734, y736, z116, z121,
 z125, z126, z133, z134, C52, C53,
 C66, C68, C95, C96, K246, K248,
 K250, O43, O44, O112, O113,
 S236, S237, S238, S240, S255, S258,
 S373, S394, S707, S713, S716, S735,
 S741, S744, S762, S768, S772, S783,
 S790, S1106, S1107, S1110, S1111,
 S1146, S1147, S1149, S1176, S1239,
 S1240, S1243, S1244, S1279, S1280,
 S1282, S1308, S1368, S1369, S1371,
 S1398, V786, V789, V803, V806,
 V823, V826, X215, X217, X221,
 X483, X485, X489, X554, X563, X615
 \reserved@c . a123, a128, f681, f684,
 f686, f689, q650, u96, u97, u606,
 u609, x48, x55, x61, x68, y33, y37,
 y315, y316, y370, y371, y434, y435,
 y515, y516, y538, y547, y562, y576,
 y817, y834, y843, y871, y882, y940,
 y953, y955, z118, z121, z131, z132,
 z135, z136, C67, C69, C76, S784,
 S786, S787, S1091, S1096, S1097,
 S1115, S1123, S1129, S1151, S1159,
 S1222, S1227, S1228, S1248, S1256,
 S1262, S1284, S1291, S1339, S1340,
 S1341, S1351, S1373, S1380, S1408,
 X219, X224, X232, X477, X487,
 X508, X509, X510, X512, X513,
 X514, X515, X516, X524, X532, X617
 \reserved@d a126,
 a129, e46, e51, f674, f683, q648,
 q650, x61, x68, x70, x74, y825,
 y834, y843, y879, y882, y948, y953,
 y957, z123, z124, z129, z130, X618
 \reserved@e o57,
 o59, o69, o71, o100, o107, o115,
 x39, x45, x70, x73, x74, y34, y39, X619
 \reserved@f o58, o59,
 o70, o71, o115, r1535, r1536, r1537,
 r1538, r1540, r1547, u190, u192,
 u198, u199, w382, w393, w397,
 w401, w407, w410, w449, w486,
 w489, x27, x38, x45, x71, x73, X620

File Key: a=ltldirchk.dtx, b=ltplain.dtx, c=ltvers.dtx, d=ltluatex.dtx, e=ltexpl.dtx,
 f=ltdefns.dtx, g=ltcmd.dtx, h=lt hooks.dtx, i=ltcmdhooks.dtx, j=ltalloc.dtx, k=ltcntrl.dtx,
 l=lterror.dtx, m=ltpar.dtx, n=lt para.dtx, o=lt space.dtx, p=lt logos.dtx, q=lt files.dtx,
 r=ltoutenc.dtx, s=lt counts.dtx, t=lt length.dtx, u=ltfssbas.dtx, v=ltfssaxes.dtx,
 w=ltfssstrc.dtx, x=ltfsscmp.dtx, y=ltfssdcl.dtx, z=ltfssini.dtx, A=fontdef.dtx,
 B=preload.dtx, C=ltfntcmd.dtx, D=lttextcomp.dtx, E=ltpageno.dtx, F=ltxref.dtx,
 G=ltmiscen.dtx, H=ltmath.dtx, I=ltlists.dtx, J=ltboxes.dtx, K=lttab.dtx, L=lt pictur.dtx,
 M=ltthm.dtx, N=ltsect.dtx, O=ltfloat.dtx, P=ltidxglo.dtx, Q=ltbibl.dtx, R=lt page.dtx,
 S=ltclass.dtx, T=ltfilehook.dtx, U=ltshipout.dtx, V=ltoutput.dtx, W=lthyphen.dtx,
 X=ltfinal.dtx

<code>\reset@font</code>	<code>\set@color</code>
..... 487 , z154 , z506 , z537 , z552 , J88
z567 , F13 , J376 , J394 , O175 , O371 ,	<code>\set@curr@file</code> 779 , 780 , 815 ,
O466 , O485 , Q20 , R14 , V616 , V675	817–819 , q225 , q234 , q261 , q269 ,
<code>\restglb@settings</code>	q388 , q406 , S818 , S1089 , S1220 , T265
w268 , w278	<code>\set@display@protect</code>
<code>\restore@mathversion</code> f8 , f16 , f251 , l7 , l14 , l34 , l61 , S314
..... y107 , y110 , y125 , y133	<code>\set@fontsize</code>
<code>\restore@protect</code> u317 , u319 , w121 , w167 , w177
f253	<code>\set@mathaccent</code>
<code>\rlh@</code> y630 , y638 , y673 , y681 , y699
A464 , A465	<code>\set@mathchar</code>
<code>\rm@def@ult</code>	y754 , y764
z190	<code>\set@mathdelimiter</code> ..
<code>\rmdef@ult</code>	y831 , y840 , y892
489 , 495 , z191 ,	<code>\set@mathradical</code>
z210 , z229 , z271 , z282 , z293 , D27 , D84	y276 , y950
<code>\robust@command@act</code>	<code>\set@mathsymbol</code>
..... 89 , 91 , 92 , 235 , 237 ,	y738 , y746 , y767
241 , f410 , f411 , f413 , f479 , f503 , i70	<code>\set@simple@size@args</code>
<code>\robust@command@act@chk@args</code> w348 , w361 , w368 , w389 , w403
..... 89 , 90 , f435 , f456	<code>\set@size@funct@args</code> w351 , w353 , w411
<code>\robust@command@act@do</code> ..	<code>\set@size@funct@args@</code>
90 , f421 , f453	w411
<code>\robust@command@act@end</code>	<code>\set@target@series</code>
..... 89 , f418 , f419 , f431 , f434 , f454 u288 , u300 , v437 , v441 ,
<code>\robust@command@act@loop</code>	v444 , v447 , v470 , v472 , v481 , v516
..... 90 , f415 , f421 , f451	<code>\set@typeset@protect</code>
<code>\robust@command@act@loop@aux</code> f251 , f270 , K197 , K235 ,
..... f421 , f452	U84 , U128 , V603 , V605 , V661 , V663
<code>\robust@command@chk@safe</code>	<code>\SetMathAlphabet@</code> ...
86 ,	y450 , y519 , y528
95 , f303 , f414 , f435 , f455 , f560 , f572	<code>\SetSymbolFont@</code>
<code>\s@fct@</code>	y340 , y374 , y382
w426 , w490	<code>\sf@def@ult</code>
<code>\s@fct@alias</code>	z190
..... w552	<code>\sf@size</code>
<code>\s@fct@fixed</code>	p6 , r302 , u224 , u243 , u628 , w328 ,
..... w565	w332 , D672 , O409 , O416 , O437 , O444
<code>\s@fct@gen</code>	<code>\sfdef@ult</code>
w502	489 , z192 ,
<code>\s@fct@genb</code>	z211 , z230 , z272 , z283 , z294 , D29 , D86
w507	<code>\sh@ft</code>
<code>\s@fct@sgen</code>	b462
w502	<code>\show@kernel@robust@command</code> f566 , f626
<code>\s@fct@sgenb</code>	<code>\ShowCommand</code>
w507	238
<code>\s@fct@sub</code>	<code>\sixt@@n</code>
w514	a71 ,
<code>\s@fct@subf</code>	b16 , b64 , b66 , b96 , b97 , b98 , d30 ,
w557	u14 , y84 , y207 , y616 , y618 , y664 ,
<code>\saved@space@catcode</code> ...	y666 , y725 , y727 , y773 , y775 , y813 ,
X341 , X410	y815 , y821 , y823 , y867 , y869 , y875 ,
<code>\scan@@fontshape</code>	y877 , y936 , y938 , y944 , y946 , L278 ,
x7 , x40 , x43	L293 , L295 , O62 , O80 , O131 , O153 ,
<code>\scan@fontshape</code>	V1005 , V1051 , V1190 , V1358 ,
x6 , x26 , x37	V1592 , V1656 , V1713 , V1783 ,
<code>\scantokens</code>	V2009 , V2018 , V2074 , V2090 , V2123
234 , 241–243	<code>\sixt@@n_</code>
<code>\scriptfont@name</code>	246
w333 , w338	<code>\size@update</code>
<code>\section</code> w146 , w172 , w185 , w204 , w206
233	<code>\sizefn@info</code>
<code>\select@group</code> u589 , u608 , y48 , y268 , w352 , w354 , w362 , w390 , w404
y305 , y443 , y496 , y505 , y543 , y575	<code>\skip@</code>
<code>\series@change@debug</code>	b41 , b424 , b426 ,
..... z72 , z79 , z82 , z93 , z96 ,	b427 , b429 , o83 , o442 , C105 , C108
z100 , z112 , z120 , z125 , z131 , z134 , z136	<code>\sp@n</code>
<code>\series@check@toks</code> ..	K379
v490 , v492 , v499	
<code>\series@drop@one@m</code> ..	
v496 , v500 , v518	
<code>\series@maybe@drop@one@m</code>	
..... u31 , v483 ,	
v485 , v517 , z103 , z122 , z128 , z194 , z195	
<code>\series@maybe@drop@one@m@x</code> v486 , v488	
<code>\seriesdefault@kernel</code> 504 , z155 , z626	
<code>\set@@mathdelimiter</code>	
y880 , y914	

File Key: a=ltldirchk.dtx, b=ltplain.dtx, c=ltvers.dtx, d=ltluatex.dtx, e=ltexpl.dtx, f=ltdefns.dtx, g=ltcmd.dtx, h=lt hooks.dtx, i=ltcmdhooks.dtx, j=ltalloc.dtx, k=ltcntrl.dtx, l=lterror.dtx, m=ltpar.dtx, n=lt para.dtx, o=lt space.dtx, p=ltlogos.dtx, q=ltfiles.dtx, r=ltoutenc.dtx, s=ltcounts.dtx, t=ltlength.dtx, u=ltfssbas.dtx, v=ltfssaxes.dtx, w=ltfssstrc.dtx, x=ltfsscmp.dtx, y=ltfssdcl.dtx, z=ltfssini.dtx, A=fontdef.dtx, B=preload.dtx, C=ltfntcmd.dtx, D=lttextcomp.dtx, E=ltpageno.dtx, F=ltxref.dtx, G=ltmiscen.dtx, H=ltmath.dtx, I=ltlists.dtx, J=ltboxes.dtx, K=lttab.dtx, L=lt pictur.dtx, M=ltthm.dtx, N=ltsect.dtx, O=ltfloat.dtx, P=ltidglo.dtx, Q=ltbibl.dtx, R=lt page.dtx, S=ltclass.dtx, T=ltfilehook.dtx, U=ltshipout.dtx, V=ltoutput.dtx, W=lthyphen.dtx, X=ltfinal.dtx

\split@name	D1090, D1092, D1094, D1096,
. u357 , u369 , u483 , u498 , w519 , w533	D1098, D1100, D1102, D1104,
\ssf@size	D1106, D1108, D1110, D1112,
r1246 , u225 , u244 , u629 , w328 , w334	D1114, D1116, D1118, D1120, D1122
\string@makeletter	\tc@error D110 , D918 , D939
.. f701 , S789 , S827 , S828 , S829 , T169	\tc@errorwarn
\strip@prefix D21 , D76 , D78 , D825 , D827 ,
a323 , f228 , f290 , f328 , f356 , f698 , u586	D829 , D830 , D877 , D878 , D879 , D912
\strip@pt	\tc@fake@euro D97 , D352 , D926 , D1007
b466 ,	\tc@forcedfalse D871
u216 , u222 , u223 , u224 , u225 ,	\tc@forcedtrue D876
u238 , u242 , u329 , u628 , u629 , w180	\tc@oldstylesubst D16 , D20
\sub@sfcnt ... w514 , w515 , w516 , w543	\tc@subst D77 , D109 , D911 , D938
\subf@sfcnt w557 , w558 , w559	\tc@swap@accent D112 , D113
\subst@correction u85 , u91	\test@font@series@context
\subst@fontshape x8 , x80 495 , z358 , z368 , z388
\subst@size w465	\text@command C8 , C46
\sw@slant C91 , C101	\textfont@name w331 , w337
\t@st@ic C90 , C94	\tf@size u223 , u243 , u627 , w328 , w330
\target@meta@family@value	\thr@@ b16 , b486 ,
..... z85 , z110 , z117 , z119	b496 , b530 , w58 , w254 , w260 , w273 ,
\target@series@value	w280 , w287 , w292 , H362 , H507 ,
. 485 , 490 , 491 , z84 , z92 , z95 , z97 ,	L287 , L288 , L290 , L291 , L329 ,
z101 , z102 , z103 , z126 , z132 , z133 , z135	L355 , L388 , L411 , X70 , X78 , I232 , I243
\tc@check@accent D109 ,	\toks@ 416 , b41 , c91 , c94 , c96 ,
D161 , D163 , D165 , D167 , D169 ,	c103 , c107 , c110 , c115 , f815 , f816 ,
D171 , D173 , D175 , D177 , D179 ,	o408 , o409 , o414 , u148 , u152 , u154 ,
D181 , D183 , D185 , D187 , D189 ,	u157 , u221 , u226 , y6 , y7 , y291 ,
D191 , D938 , D1014 , D1016 , D1018	y295 , y301 , y304 , y309 , y355 , y356 ,
\tc@check@symbol D109 ,	y358 , y359 , y389 , y391 , y395 , y412 ,
D211 , D213 , D215 , D217 , D219 ,	y415 , y474 , y486 , y487 , y488 , y534 ,
D221 , D223 , D225 , D227 , D229 ,	y536 , y542 , y550 , y554 , y566 , y569 ,
D231 , D233 , D235 , D237 , D239 ,	y572 , y580 , y582 , y1003 , y1005 ,
D241 , D243 , D245 , D247 , D249 ,	y1007 , y1010 , y1012 , y1015 , y1018 ,
D251 , D253 , D255 , D257 , D259 ,	y1050 , y1051 , S424 , S425 , S427 ,
D261 , D263 , D265 , D267 , D269 ,	S428 , V2209 , V2210 , V2211 , V2212
D271 , D273 , D275 , D277 , D279 ,	\try@load@fontshape
D281 , D283 , D285 , D287 , D289 , 415 , u372 , u380 ,
D291 , D293 , D295 , D297 , D299 ,	u426 , u531 , v479 , w520 , y240 , y257
D301 , D303 , D305 , D307 , D310 ,	\try@simple@size w356 , w481
D312 , D314 , D316 , D318 , D320 ,	\try@simples w439 , w445 , w449
D322 , D324 , D326 , D328 , D330 ,	\try@size@range ... w101 , w356 , w432
D332 , D334 , D336 , D338 , D340 ,	\try@size@substitution .. w103 , w436
D342 , D344 , D346 , D348 , D350 ,	\tryif@simple w447 , w448
D354 , D356 , D358 , D360 , D362 ,	\tryis@simple w448
D364 , D366 , D938 , D1008 , D1010 ,	\tt@def@ult z190
D1012 , D1020 , D1022 , D1024 ,	\ttdef@ult 489 , z193 ,
D1026 , D1028 , D1030 , D1032 ,	z212 , z231 , z273 , z284 , z295 , D31 , D88
D1034 , D1036 , D1038 , D1040 ,	\tw@ 246 , b16
D1042 , D1044 , D1046 , D1048 ,	\two@digits a86 , a185 , a186 ,
D1050 , D1052 , D1054 , D1056 ,	f2 , w512 , S1046 , S1135 , S1268 , S1357
D1058 , D1060 , D1062 , D1064 ,	\type@restoreinfo w202 , w207
D1066 , D1068 , D1070 , D1072 ,	\undeclare@... 849
D1074 , D1076 , D1078 , D1080 ,	
D1082 , D1084 , D1086 , D1088 ,	

File Key: a=ltldirchk.dtx, b=ltplain.dtx, c=ltvers.dtx, d=ltluatex.dtx, e=ltexpl.dtx, f=ltdefns.dtx, g=ltcmd.dtx, h=lthooks.dtx, i=ltcmdhooks.dtx, j=ltalloc.dtx, k=ltcntrl.dtx, l=lterror.dtx, m=ltpar.dtx, n=ltpara.dtx, o=ltspace.dtx, p=ltlogos.dtx, q=ltfiles.dtx, r=ltoutenc.dtx, s=ltcounts.dtx, t=ltlength.dtx, u=ltfssbas.dtx, v=ltfssaxes.dtx, w=ltfssstrc.dtx, x=ltfsscmp.dtx, y=ltfssdcl.dtx, z=ltfssini.dtx, A=fontdef.dtx, B=preload.dtx, C=ltfntcmd.dtx, D=lttextcomp.dtx, E=ltpageno.dtx, F=ltxref.dtx, G=ltmiscen.dtx, H=ltmath.dtx, I=ltlists.dtx, J=ltboxes.dtx, K=lttab.dtx, L=ltpictur.dtx, M=ltthm.dtx, N=ltsect.dtx, O=ltfloat.dtx, P=ltidxglo.dtx, Q=ltbibl.dtx, R=ltpage.dtx, S=ltclass.dtx, T=ltfilehook.dtx, U=ltshipout.dtx, V=ltoutput.dtx, W=lthyphen.dtx, X=ltfinal.dtx

- \undecclare@file@substitution ... 806, [T245](#)
- \unqu@tefilef@und ... [T143](#)
- \unquote@name . [q392](#), [q398](#), [q415](#), [T316](#)
- \unrestored@protected@xdef . [f253](#),
[O458](#), [O508](#), [O524](#), [O535](#), [R25](#), [R49](#)
- \update@series@target@value
..... [z87](#), [z108](#), [z180](#)
- \update@ucllc@with@cyrillic
..... [r1497](#), [r1525](#), [r1555](#), [r1563](#)
- \upper@bound . [w383](#), [w384](#), [w385](#), [w398](#)
- \use@mathgroup
..... [u595](#), [u613](#), [u615](#), [w299](#), [y63](#),
[y92](#), [y456](#), [y558](#), [y561](#), [y1028](#), [y1052](#)
- \UTF@four@octets@noexpand [X557](#), [X566](#)
- \UTF@three@octets@noexpand
..... [X556](#), [X565](#)
- \UTF@two@octets@noexpand [X555](#), [X564](#)
- \UTFviii@four@octets [X396](#), [X401](#), [X407](#)
- \UTFviii@four@octets@@ .. [X396](#), [X407](#)
- \UTFviii@four@octets@combine . [X431](#)
- \UTFviii@four@octets@noexpand . [X437](#)
- \UTFviii@four@octets@string .. [X434](#)
- \UTFviii@invalid [X335](#), [X428](#)
- \UTFviii@invalid@err [X393](#), [X398](#), [X404](#)
- \UTFviii@invalid@err@@ .. [X393](#), [X404](#)
- \UTFviii@three@octets
..... [X395](#), [X400](#), [X406](#)
- \UTFviii@three@octets@@ . [X395](#), [X406](#)
- \UTFviii@three@octets@combine . [X430](#)
- \UTFviii@three@octets@noexpand [X436](#)
- \UTFviii@three@octets@string . [X433](#)
- \UTFviii@two@octets [X394](#), [X399](#), [X405](#)
- \UTFviii@two@octets@@ .. [X394](#), [X405](#)
- \UTFviii@two@octets@combine .. [X429](#)
- \UTFviii@two@octets@noexpand . [X435](#)
- \UTFviii@two@octets@string ... [X432](#)
- \UTFviii@undefined@err
..... [X392](#), [X397](#), [X403](#)
- \UTFviii@undefined@err@@ [X392](#), [X403](#)
- \v@false [H82](#)
- \v@true [H81](#), [H83](#)
- \ver@<file>.<ext> [779](#)
- \verb@balance@group . [G508](#), [G510](#),
[G523](#), [G525](#), [G538](#), [G540](#), [G546](#), [G547](#)
- \verb@egroup [G508](#), [G511](#),
[G523](#), [G526](#), [G538](#), [G541](#), [G547](#), [G551](#)
- \verb@eol@error .. [G548](#), [G560](#), [G570](#)
- \verbatim@font
..... [G432](#), [G453](#), [G461](#), [G561](#), [G571](#)
- \verbatim@nolig@list ... [G576](#), [G582](#)
- \version@elt
..... [y18](#), [y31](#), [y32](#), [y288](#), [y289](#),
[y338](#), [y358](#), [y449](#), [y487](#), [y579](#), [y1008](#)
- \version@list [y16](#),
[y21](#), [y32](#), [y281](#), [y289](#), [y343](#), [y364](#),
[y383](#), [y454](#), [y499](#), [y529](#), [y584](#), [y1021](#)
- \vgl@ [b424](#), [b425](#)
- \voidb@x [b311](#), [b455](#), [t18](#)
- \warn@rel@i [x5](#),
[x25](#), [x29](#), [x81](#), [x85](#), [x90](#), [x95](#), [x119](#), [x140](#)
- \wrong@fontshape [u376](#), [u513](#)
- \x@protect
..... [f231](#), [f242](#), [f293](#), [f331](#), [f359](#), [f530](#), [f550](#)
- \xe@alloc@ [X42](#), [X52](#)
- \xe@alloc@intercharclass [X21](#)
- \xe@ch@ck [X43](#), [X47](#)
- \z@ [246](#), [b311](#)
- \z@skip [b311](#)
- \zap@space
..... [q261](#), [q281](#), [z399](#), [Q29](#), [S236](#), [S378](#),
[S399](#), [S411](#), [S576](#), [S676](#), [S697](#), [S715](#),
[S726](#), [S743](#), [S753](#), [S770](#), [S1059](#), [S1190](#)
- tex commands:
- \tex_afterassignment:D
..... [g1868](#), [U49](#), [U138](#)
- \tex_aftergroup:D ... [834](#), [U57](#), [U144](#)
- \tex_currentgrouplevel:D
..... [U48](#), [U56](#), [U137](#), [U143](#)
- \tex_deadcycles:D [U91](#)
- \tex_def:D [240](#), [i161](#)
- \tex_endlinechar:D
..... [g1362](#), [g1364](#), [g1365](#), [g1371](#), [i239](#)
- \tex_escapechar:D [152](#), [153](#),
[g1265](#), [g1752](#), [g1778](#), [i152](#), [T210](#), [T224](#)
- \tex_everypar:D [272–274](#),
[n18](#), [n22](#), [n26](#), [n34](#), [n77](#), [n79](#), [n89](#), [n90](#)
- \tex_indent:D [n84](#)
- \tex_kern:D [n63](#)
- \tex_lastnodetype:D [276](#), [n62](#)
- \tex_lowercase:D [g1628](#), [g1758](#)
- \tex_newlinechar:D [i240](#)
- \tex_noindent:D [273](#), [n24](#), [n94](#)
- \tex_par:D [275](#), [276](#),
[n16](#), [n65](#), [n72](#), [n96](#), [n137](#), [n138](#), [n139](#)
- \tex_parskip:D [n23](#)
- \tex_setbox:D [U50](#), [U139](#)
- \tex_shipout:D ... [U126](#), [U371](#), [U501](#)
- \tex_unskip:D [275](#), [n58](#)
- \tex_vss:D [U334](#)
- \TeXOrMath [s127](#), [s143](#)
- \textacutedbl
..... [r915](#), [r1145](#), [D234](#), [D235](#), [D721](#), [D971](#)
- \textascendercompwordmark
..... [r854](#), [D154](#), [D685](#), [D954](#)
- \textasciicute
..... [r965](#), [r1106](#), [D236](#), [D237](#), [D722](#), [D995](#)

File Key: a=ltldirchk.dtx, b=ltplain.dtx, c=ltvers.dtx, d=ltluatex.dtx, e=ltexpl.dtx, f=ltdefns.dtx, g=ltcmd.dtx, h=lthooks.dtx, i=ltcmdhooks.dtx, j=ltalloc.dtx, k=ltcntrl.dtx, l=terror.dtx, m=ltpar.dtx, n=ltpara.dtx, o=ltspace.dtx, p=ltlogos.dtx, q=ltfiles.dtx, r=ltoutenc.dtx, s=ltcounts.dtx, t=ltlength.dtx, u=ltfssbas.dtx, v=ltfssaxes.dtx, w=ltfssstrc.dtx, x=ltfsscmp.dtx, y=ltfssdcl.dtx, z=ltfssini.dtx, A=fontdef.dtx, B=preload.dtx, C=ltfntcmd.dtx, D=lttextcomp.dtx, E=ltpageno.dtx, F=ltxref.dtx, G=ltmiscen.dtx, H=ltmath.dtx, I=ltlists.dtx, J=ltboxes.dtx, K=lttab.dtx, L=ltpictur.dtx, M=ltthm.dtx, N=ltsect.dtx, O=ltfloat.dtx, P=ltidxglo.dtx, Q=ltbibl.dtx, R=ltpage.dtx, S=ltclass.dtx, T=ltfilehook.dtx, U=ltshipout.dtx, V=ltoutput.dtx, W=lthyphen.dtx, X=ltfinal.dtx

<code>\textasciibreve</code>	r913, r1144, D238, D239, D723, D968	r706, r707, r708, r709, r710, r711, r712, r1244, r1447, r1448, r1449, r1450
<code>\textasciicaron</code>	r914, r1143, D240, D241, D724, D969	<code>\textcompwordmark</code>
<code>\textasciicircum</code>	r286, r558, r1080 D34, D39, D91, D618, D916
<code>\textasciidieresis</code>	r953, r1093, D242, D243, D725, D985 334, r290, r291, r564, r1150
<code>\textasciigrave</code>	r904, r1072, D244, D245, D726, D966	<code>\textcopyleft</code>
<code>\textasciimacron</code>	r960, r1101, D246, D247, D727, D990	r956, D262, D263, D368, D736, D1115, D1116
<code>\textasciitilde</code>	r287, r559, r1085	<code>\textcopyright</code>
<code>\textasteriskcentered</code>	348, r267, r717, r864, r865, r1207, s130, s136, D120, D577, D642, D961	r283, r317, r954, r1094, D131, D654, D986
<code>\textbackslash</code> ...	r268, r560, r718, r1079	<code>\textcurrency</code> r949, r1089, D335, D336, D778, D853, D857, D1011, D1012
<code>\textbaht</code>	r939, r1148, D252, D253, D731, D1101, D1102	<code>\textdagger</code> r275, r312, r725, r917, r1169, s131, s137, D123, D581, D645, D972
<code>\textbar</code>	r269, r561, r719, r1083	<code>\textdaggerdbl</code>
<code>\textbardbl</code>	r270, r720, r919, r1162, s135, D127, D365, D366, D578, D650, D797, D974	... r274, r313, r724, r918, r1170, s132, s138, D122, D580, D644, D973
<code>\textbf</code>	484, C19	<code>\textdblhyphen</code>
<code>\textbigcircle</code>	r729, r892, r1224, D254, D255, D732, D1053, D1054	r876, D266, D267, D369, D738, D1025, D1026
<code>\textblank</code>	r861, r1221, D345, D346, D785, D1023, D1024	<code>\textdblhyphenchar</code>
<code>\textborn</code>	r905, D256, D257, D387, D733, D1059, D1060	r912, D264, D265, D370, D737, D1071, D1072
<code>\textbraceleft</code> r271, r308, r562, r721, r1082		<code>\textdegree</code> r961, r1102, D132, D656, D991
<code>\textbraceright</code> r272, r309, r563, r722, r1084		<code>\textdied</code>
<code>\textbrokenbar</code> r951, r1091, D128, D651, D983	r907, D268, D269, D388, D739, D1063, D1064
<code>\textbullet</code>	r273, r723, r921, r1171, D121, D579, D643, D976	<code>\textdiscount</code>
<code>\textcapitalcompwordmark</code> r853, D153, D684, D953	r941, r1182, D270, D271, D740, D1105, D1106
<code>\textcelsius</code>	r922, r1190, D129, D353, D354, D652, D791, D977	<code>\textdiv</code> . r978, r1127, D133, D657, D1005
<code>\textcent</code> . r947, r1087, D130, D653, D980		<code>\textdivorced</code>
<code>\textcentoldstyle</code> 550, r924, D258, D259, D392, D395, D734, D1075, D1076	r906, r1227, D272, D273, D741, D1061, D1062
<code>\textcircled</code> 349, 544, r279, r283, r300, r301, r730, r893, D155, D156, D655, D671, D687, D858, D1123, D1125		<code>\textdollar</code>
<code>\textcircledP</code>	r958, r1192, D260, D261, D735, D1117, D1118	r255, r307, r438, r565, r798, r862, r1076, D114, D115, D623, D625, D959, D1131, D1133
<code>\textcolonmonetary</code>	r926, r1183, D313, D314, D765, D1077, D1078	<code>\textdollaroldstyle</code>
<code>\textcommaabove</code>	r353, r355, r369, r370, r458, r459, r700, r701 550, r923, D274, D275, D393, D394, D742, D1073, D1074
<code>\textcommabelow</code>	r324, r326, r332, r333, r703, r704, r705,	<code>\textdong</code>
		r935, r1187, D315, D316, D766, D1095, D1096
		<code>\textdownarrow</code>
		r903, r1202, D317, D318, D767, D1057, D1058
		<code>\texteightoldstyle</code>
		r886, D214, D215, D384, D710, D1043, D1044
		<code>\textellipsis</code>
		r296, r321, r1172
		<code>\textemdash</code>
		r256, r407, r411, r566, r570, r785, r1154
		<code>\textendash</code>
		r257, r408, r410, r567, r569, r786, r1153
		<code>\textestimated</code>
		r942, r1198, D329, D330, D774, D856, D1009, D1010
		<code>\texteuro</code>
		r976, r1188, D351, D352, D789, D854, D1006, D1007
		<code>\textexclamdown</code>
		.. r258, r412, r414, r571, r787, r1086
		<code>\textfiguredash</code> .
		r410, r569, r1152, r1158

File Key: a=ltldirchk.dtx, b=ltplain.dtx, c=ltvers.dtx, d=ltluatex.dtx, e=ltexpl.dtx, f=ltdefs.dtx, g=ltcmd.dtx, h=lt hooks.dtx, i=ltcmdhooks.dtx, j=ltalloc.dtx, k=ltcntrl.dtx, l=ltterror.dtx, m=ltpar.dtx, n=lt para.dtx, o=lt space.dtx, p=ltlogos.dtx, q=ltfiles.dtx, r=ltoutenc.dtx, s=ltcounts.dtx, t=ltlength.dtx, u=ltfssbas.dtx, v=ltfssaxes.dtx, w=ltfssstrc.dtx, x=ltfsscmp.dtx, y=ltfssdcl.dtx, z=ltfssini.dtx, A=fontdef.dtx, B=preload.dtx, C=ltfntcmd.dtx, D=lttextcomp.dtx, E=ltpageno.dtx, F=ltxref.dtx, G=ltmiscen.dtx, H=ltmath.dtx, I=ltlists.dtx, J=ltboxes.dtx, K=lttab.dtx, L=lt pictur.dtx, M=ltthm.dtx, N=ltsect.dtx, O=ltfloat.dtx, P=ltidnglo.dtx, Q=ltbibl.dtx, R=ltpage.dtx, S=ltclass.dtx, T=ltfilehook.dtx, U=ltshipout.dtx, V=ltoutput.dtx, W=lthyphen.dtx, X=ltfinal.dtx

`\textfiveoldstyle` r883, D216, D217, D381, D711, D1037, D1038
`\textfloatsep` V728, V741, V2104, V2154, V2303
`\textflorin` r925, r1141, D333, D334, D777, D978
`\textfont` w337, H238
`\textfouroldstyle` r882, D218, D219, D380, D712, D1035, D1036
`\textfraction` V1917, V1920, V1944, V1947, V2096, V2297
`\textfractionsolidus` r877, r1179, D337, D338, D780, D962
`\textgravedbl` r916, r1146, D248, D249, D728, D970
`\textgreater` r281, r572, r740, r1078
`\textguarani` r929, D276, D277, D391, D743, D1083, D1084
`\textheight` q22, q23, q91, q92, q148, q149, O257, O258, O261, O287, O301, U371, V78, V225, V226, V274, V399, V447, V474, V645, V704, V763, V815, X142, X143
`\texthorizontalbar` r411, r570, r1155, r1160
`\texthyphen` r260, r417, r574, r789
`\texthyphenchar` ... r259, r416, r573, r788
`\textinterrobang` r933, r1178, D349, D350, D787, D1091, D1092
`\textinterrobangdown` ... r934, r1228, D347, D348, D786, D1093, D1094
`\textit` C21
`\textlangle` r888, r1219, D309, D310, D762, D1047, D1048
`\textlbrackdbl` r900, D210, D211, D389, D707, D964
`\textleaf` r908, D278, D279, D373, D744, D1065, D1066
`\textleftarrow` r859, r1199, D319, D320, D768, D1019, D1020
`\textlegacyasteriskcentered` D589, D800
`\textlegacybardbl` D589, D801
`\textlegacybullet` D589, D802
`\textlegacydagger` D589, D804
`\textlegacydaggerdbl` D589, D803
`\textlegacyparagraph` D589, D805
`\textlegacyperiodcentered` .. D589, D806
`\textlegacysection` D589, D807
`\textless` r280, r575, r739, r1077
`\textlira` r931, r1184, D321, D322, D769, D1087, D1088
`\textlnot` . r957, r1099, D134, D658, D988
`\textlquill` r945, r1180, D280, D281, D745, D1111, D1112
`\textmarried` r909, r1226, D282, D283, D746, D1067, D1068
`\textmd` C19
`\textmho` r891, r1197, D284, D285, D747, D1051, D1052
`\textminus` r889, r1203, D343, D344, D783, D963
`\textmu` r966, r1107, D341, D342, D782, D996
`\textmusicalnote` r910, r1225, D286, D287, D748, D1069, D1070
`\textnaira` r928, r1185, D288, D289, D749, D1081, D1082
`\textnineoldstyle` r887, D220, D221, D385, D713, D1045, D1046
`\textnonbreakinghyphen` r409, r568, r1151, r1156
`\textnormal` C15
`\textnumero` r940, r1191, D331, D332, D775, D1103, D1104
`\textogonekcentered` r487, r698, r699
`\textohm` r899, r1196, D339, D340, D781, D855, D1008
`\textonehalf` r974, r1116, D135, D659, D1002
`\textoneoldstyle` r879, D222, D223, D377, D714, D1029, D1030
`\textonequarter` r973, r1115, D136, D660, D1001
`\textonesuperior` r970, r1110, D137, D355, D356, D661, D792, D999
`\textopenbullet` r943, r1223, D290, D291, D750, D1107, D1108
`\textordfeminine` r305, r955, r1095, D138, D662, D987
`\textordmasculine` r306, r971, r1111, D139, D664, D1000
`\TextOrMath` s130, s131, s132, s133, s134, s135, s136, s137, s138, s150
`\textparagraph` . r276, r310, r726, r967, r1108, s134, D124, D582, D646, D997
`\textperiodcentered` r277, r727, r968, r1109, D125, D583, D647, D998
`\textpertenthousand` 547, r492, r937, r1174, D306, D307, D308, D759, D1097, D1098, D1135
`\textperthousand` r490, r920, r1173, D118, D119, D639, D975, D1134
`\textpeso` r930, r1189, D292, D293, D751, D1085, D1086
`\textpilcrow` r938, D294, D295, D386, D752, D1099, D1100
`\textpm` ... r962, r1103, D140, D666, D992
`\textquestiondown` r261, r413, r415, r576, r790, r1118
`\textquotedbl` r579, r1074

File Key: a=ltldirchk.dtx, b=ltplain.dtx, c=ltvers.dtx, d=ltluatex.dtx, e=ltexpl.dtx, f=ltdefns.dtx, g=ltcmd.dtx, h=lt hooks.dtx, i=ltcmdhooks.dtx, j=ltalloc.dtx, k=ltcntrl.dtx, l=lterror.dtx, m=ltpar.dtx, n=lt para.dtx, o=lt space.dtx, p=ltlogos.dtx, q=ltfiles.dtx, r=ltoutenc.dtx, s=ltcounts.dtx, t=ltlength.dtx, u=ltfssbas.dtx, v=ltfssaxes.dtx, w=ltfssstrc.dtx, x=ltfsscmp.dtx, y=ltfssdcl.dtx, z=ltfssini.dtx, A=fontdef.dtx, B=preload.dtx, C=ltfntcmd.dtx, D=lttextcomp.dtx, E=ltpageno.dtx, F=ltxref.dtx, G=ltmiscen.dtx, H=ltmath.dtx, I=ltlists.dtx, J=ltboxes.dtx, K=lttab.dtx, L=lt pictur.dtx, M=ltthm.dtx, N=ltsect.dtx, O=ltfloat.dtx, P=ltidxglo.dtx, Q=ltbibl.dtx, R=ltpage.dtx, S=ltclass.dtx, T=ltfilehook.dtx, U=ltshipout.dtx, V=ltoutput.dtx, W=lthyphen.dtx, X=ltfinal.dtx

<code>\textquotedblleft</code>	<code>\textthreeoldstyle</code>
..... r262, r418, r577, r791, r1166	r881, D228, D229, D379, D717, D1033, D1034
<code>\textquotedblright</code>	<code>\textthreequarters</code>
..... r263, r419, r578, r792, r1167 r975, r1117, D146, D675, D1003
<code>\textquoteleft</code> r264, r420, r580, r793, r1163	<code>\textthreequartersemdash</code> . r858, D145,
<code>\textquoteright</code> r265, r421, r581, r794, r1164	D357, D358, D375, D674, D793, D958
<code>\textquotesingle</code>	<code>\textthreesuperior</code>
..... r863, r1070, D141, D667, D960	r964, r1105, D147, D359, D360, D676, D794, D994
<code>\textquotestraightbase</code>	<code>\texttildelow</code>
..... r855, D142, D371, D668, D955	r911, r1147, D250, D251, D729, D967
<code>\textquotestraightdblbase</code>	<code>\texttimes</code> r977, r1121, D148, D677, D1004
..... r856, D143, D372, D669, D956	<code>\texttrademark</code>
<code>\textrangle</code> r303, r936, r1195, D149, D678, D979
r890, r1220, D311, D312, D763, D1049, D1050	<code>\texttt</code>
<code>\textrbrackdbl</code> C15
r901, D212, D213, D390, D708, D965	<code>\texttwelveudash</code>
<code>\textrecipe</code>	r857, D150, D361, D362, D374, D680, D795, D957
r932, r1193, D296, D297, D753, D1089, D1090	<code>\texttwooldstyle</code>
<code>\textreferencemark</code>	r880, D230, D231, D378, D718, D1031, D1032
r969, r1177, D298, D299, D754, D1119, D1120	<code>\texttwosuperior</code>
<code>\textregistered</code>	r963, r1104, D151, D363, D364, D681, D796, D993
r300, r301, r959, r1100, D144, D670, D989	<code>\textulc</code>
<code>\textrightarrow</code>	v526, C28, C29, C35, C37
r860, r1201, D323, D324, D770, D1021, D1022	<code>textulc</code>
<code>\textrm</code>	C25
C15	<code>\textunderscore</code> ..
<code>\textrquill</code>	r288, r315, r584, r1081
r946, r1181, D300, D301, D755, D1113, D1114	<code>\textup</code>
<code>\textsc</code>	418, C21
C21	<code>\textuparrow</code>
<code>\textsection</code>	r902, r1200, D325, D326, D771, D1055, D1056
r278, r311, r582, r728, r952, r1092, s133, D126, D584, D585, D648, D984	<code>\textvisiblespace</code>
<code>\textservicemark</code>	r292, r585, r1222
r944, r1194, D302, D303, D756, D1109, D1110	<code>\textwidth</code>
<code>\textsevenoldstyle</code> q24, q93, q150, J346, O266, V79, V144, V201, V218, V630, V640, V689, V699, V2223, V2255, X143
r885, D224, D225, D383, D715, D1041, D1042	<code>\textwon</code>
<code>\textsf</code>	r927, r1186, D327, D328, D772, D1079, D1080
C15	<code>\textyen</code> ..
<code>\textsixoldstyle</code>	r950, r1090, D152, D682, D982
r884, D226, D227, D382, D716, D1039, D1040	<code>\textzerooldstyle</code>
<code>\textsl</code>	r878, D232, D233, D376, D719, D1027, D1028
C21	<code>\TH</code>
<code>\textssc</code>	r535, r1123, X571
v536, C31, C39	<code>\th</code>
<code>textssc</code>	r586, r1129, X571
C25	<code>\thanks</code>
<code>\textsterling</code> ..	706
r266, r319, r445, r583, r805, r948, r1088, D116, D117, D624, D632, D981, D1130, D1132	<code>\thanks</code>
<code>\textstyle</code>	N10, N26
p15, A469, H63	<code>\the</code>
<code>\textsubscript</code>	274
O419	<code>thebibliography</code> (environment)
<code>\textsuperscript</code>	745
r303, r305, r306, D663, D665, D679, O402	<code>\theenum</code>
<code>\textsurd</code>	626
r972, r1218, D304, D305, D757, D1121, D1122	<code>\theequation</code>
<code>\textsw</code>	H339, H351, H426, H487
v531, C30, C38	<code>\thefootnote</code> ...
<code>textsw</code>	O396, O503, O508, O528
C25	<code>\thempfn</code>
<code>\TextSymbolUnavailable</code>	J348, O453, O458, O519, O524, O527
r3, r758	<code>\thempfootnote</code>
	J348, O398
	<code>\thepage</code>
	845, q208, E6, F14, F34, N164, N171, N177, P15, P32, Q23, R14, V244, V275, V1827
	<code>\Theta</code>
	A299
	<code>\theta</code>
	A275
	<code>\thetotalpages</code>
	829, 845, U350, U433

File Key: a=ltldirchk.dtx, b=ltplain.dtx, c=ltvers.dtx, d=ltluatex.dtx, e=ltexpl.dtx, f=ltdefns.dtx, g=ltcmd.dtx, h=lthooks.dtx, i=ltcmdhooks.dtx, j=ltalloc.dtx, k=ltcntrl.dtx, l=lterror.dtx, m=ltpar.dtx, n=ltpara.dtx, o=ltspace.dtx, p=ltlogos.dtx, q=ltfiles.dtx, r=ltoutenc.dtx, s=ltcounts.dtx, t=ltlength.dtx, u=ltfssbas.dtx, v=ltfssaxes.dtx, w=ltfssstrc.dtx, x=ltfsscmp.dtx, y=ltfssdcl.dtx, z=ltfssini.dtx, A=fontdef.dtx, B=preload.dtx, C=ltfntcmd.dtx, D=lttextcomp.dtx, E=ltpageno.dtx, F=ltxref.dtx, G=ltmiscen.dtx, H=ltmath.dtx, I=ltlists.dtx, J=ltboxes.dtx, K=lttab.dtx, L=ltpicture.dtx, M=ltthm.dtx, N=ltsect.dtx, O=ltfloat.dtx, P=ltidxglo.dtx, Q=ltbibl.dtx, R=ltpage.dtx, S=ltclass.dtx, T=ltfilehook.dtx, U=ltshipout.dtx, V=ltoutput.dtx, W=lthyphen.dtx, X=ltfinal.dtx

- \thicklines [L124](#)
- \thickmuskip [A646](#), [H215](#), [H217](#), [H230](#)
- \thickspace [H201](#)
- \thinlines [L124](#), [L816](#), [L833](#)
- \thinmuskip [A644](#), [H207](#), [H209](#), [H225](#), [H231](#)
- \thinspace
 - . [o460](#), [o466](#), [o467](#), [H176](#), [H201](#), [H238](#)
- \thispagestyle [R6](#)
- \tilde [A519](#)
- \time [a179](#), [a183](#)
- \times [A396](#)
- \title [706](#)
- \title [N6](#), [N7](#), [N21](#), [N23](#), [N31](#)
- tl commands:
 - \c_empty_tl [h67](#), [h941](#)
 - \c_noalue_tl
 - . [127](#), [195](#), [g453](#), [g787](#), [g833](#), [g894](#), [g1011](#), [g1127](#), [g1232](#), [g1377](#), [g1384](#), [g1424](#), [g1476](#), [g1557](#), [g1584](#), [g1666](#)
 - \c_space_tl [g24](#), [g86](#), [g104](#), [g115](#), [g116](#), [g132](#), [g140](#), [g150](#), [g153](#), [g155](#), [g157](#), [g159](#), [g167](#), [g173](#), [g222](#), [g223](#), [n40](#), [T514](#), [T517](#)
 - \tl_clear:N [g237](#), [g279](#), [g308](#), [g375](#), [g376](#), [g573](#), [g682](#), [g692](#), [g693](#), [g694](#), [g696](#), [g818](#), [g886](#), [g972](#), [g1038](#), [g1039](#), [g1055](#), [g1266](#), [g1616](#), [i148](#), [i149](#)
 - \tl_const:Nn [g1873](#), [U311](#), [U316](#)
 - \tl_count:N [g326](#), [g1640](#), [T507](#)
 - \tl_count:n ... [g512](#), [g517](#), [g839](#), [g893](#)
 - \tl_gc_clear:N [218](#), [219](#)
 - \tl_gc_clear_new:N [h128](#)
 - \tl_gput_right:Nn [209](#), [h459](#), [h1204](#), [n36](#)
 - \tl_gremove_all:Nn [204](#)
 - \tl_gremove_once:Nn [204](#), [h37](#), [h37](#), [h464](#)
 - \tl_gset:Nn [h106](#), [h165](#), [h360](#), [h363](#), [h384](#), [h387](#), [h1166](#), [h1178](#), [h1194](#), [h1232](#), [i104](#), [n13](#), [T51](#)
 - \tl_gset_eq:NN [h65](#), [h1201](#), [h1212](#)
 - \tl_if_blank:nTF
 - . . . [g508](#), [g1021](#), [g1438](#), [g1479](#), [h1184](#)
 - \tl_if_empty:N [222](#)
 - \tl_if_empty:NTF
 - . . . [g247](#), [g248](#), [g393](#), [g623](#), [h114](#), [h214](#), [h216](#), [h809](#), [h816](#), [h934](#), [h936](#)
 - \tl_if_empty:nTF
 - [g1435](#), [g1753](#), [g1779](#), [g1814](#), [g2089](#), [h188](#), [h201](#), [h204](#), [h348](#), [h377](#), [h419](#), [i100](#), [i175](#), [T40](#), [T46](#), [T58](#), [T107](#), [T114](#), [T116](#), [T118](#), [T381](#)
 - \tl_if_empty_p:N
 - [h1010](#), [h1011](#), [U68](#), [U115](#), [U369](#)
 - \tl_if_empty_p:n [h402](#)
 - \tl_if_eq:NNTF [g268](#)
 - \tl_if_eq:nnTF [g635](#), [g852](#), [g1758](#)
 - \tl_if_exist:N [220](#)
 - \tl_if_exist:NTF [h82](#), [h112](#), [h944](#), [h984](#), [h992](#), [h1020](#), [h1032](#)
 - \tl_if_exist_p:N [h134](#), [h135](#)
 - \tl_if_in:NnTF [g1052](#), [h469](#)
 - \tl_if_in:nnTF [143](#), [g524](#), [g1018](#), [g1044](#)
 - \tl_if_noalue:nTF ... [g299](#), [g318](#), [g365](#), [g822](#), [g1508](#), [g2237](#), [g2238](#), [g2239](#), [g2240](#), [g2241](#), [g2242](#), [h182](#)
 - \tl_if_single:nTF [g1605](#), [g2220](#)
 - \tl_if_single_token:nTF
 - [163](#), [g583](#), [g1750](#), [g2222](#)
 - \tl_item:Nn [g1710](#)
 - \tl_log:n [h37](#), [h39](#), [h772](#)
 - \tl_map_function:nN
 - . . [g319](#), [g509](#), [g510](#), [g837](#), [g887](#), [g2248](#)
 - \tl_map_inline:Nn [g633](#)
 - \tl_map_inline:nn [g1124](#)
 - \tl_new:N [g11](#), [g12](#), [g13](#), [g14](#), [g17](#), [g21](#), [g28](#), [g29](#), [g30](#), [g33](#), [g37](#), [g38](#), [g40](#), [g41](#), [g46](#), [g47](#), [g1033](#), [g1034](#), [g1250](#), [g1602](#), [g1717](#), [g2243](#), [h25](#), [h26](#), [h27](#), [h29](#), [h30](#), [h34](#), [h76](#), [h85](#), [h88](#), [h97](#), [h98](#), [h163](#), [h477](#), [h595](#), [h596](#), [h597](#), [i6](#), [i8](#), [i9](#), [i10](#), [n12](#), [T8](#), [T9](#), [T10](#), [T11](#), [T59](#), [U53](#), [U205](#)
 - \tl_put_left:Nn [g734](#), [g970](#)
 - \tl_put_right:Nn . [g294](#), [g320](#), [g463](#), [g501](#), [g515](#), [g532](#), [g537](#), [g665](#), [g749](#), [g757](#), [g773](#), [g781](#), [g795](#), [g797](#), [g804](#), [g816](#), [g827](#), [g833](#), [g902](#), [g927](#), [g1042](#), [g1043](#), [g1050](#), [g1060](#), [g1128](#), [g1162](#), [g1389](#), [g1399](#), [g1618](#), [g1663](#), [i143](#), [i144](#)
 - \tl_replace_all:Nnn [g1631](#)
 - \tl_rescan:nn [i14](#), [i14](#), [i254](#), [i280](#)
 - \tl_set:Nn [g22](#), [g84](#), [g85](#), [g238](#), [g239](#), [g240](#), [g241](#), [g266](#), [g317](#), [g939](#), [g957](#), [g977](#), [g1015](#), [g1121](#), [g1148](#), [g1155](#), [g1168](#), [g1174](#), [g1180](#), [g1186](#), [g1192](#), [g1198](#), [g1204](#), [g1210](#), [g1241](#), [g1263](#), [g1285](#), [g1598](#), [g1599](#), [g1625](#), [g1649](#), [g1680](#), [g1684](#), [g1710](#), [g1843](#), [g1844](#), [h602](#), [h622](#), [h623](#), [h628](#), [h629](#), [h644](#), [h651](#), [h652](#), [h697](#), [h704](#), [h994](#), [i137](#), [i139](#), [i156](#), [i187](#), [i192](#), [i251](#), [i277](#), [T81](#), [T82](#), [T83](#), [T84](#), [U47](#), [U136](#), [U210](#), [U236](#), [U261](#), [U288](#)
 - \tl_set_eq:NN ... [g256](#), [g262](#), [g270](#), [g313](#), [g971](#), [g1054](#), [g1127](#), [h632](#), [h655](#)
 - \tl_show:N [152](#), [g1733](#), [g1740](#)
 - \tl_show:n [h37](#), [h38](#), [h777](#), [h848](#)
 - \tl_tail:N [g1319](#)
 - \tl_to_str:N [g1375](#), [g1382](#)

File Key: a=ltldirchk.dtx, b=ltplain.dtx, c=ltvers.dtx, d=ltluatex.dtx, e=ltexpl.dtx, f=ltdefns.dtx, g=ltcmd.dtx, h=lt hooks.dtx, i=ltcmdhooks.dtx, j=ltalloc.dtx, k=ltcntrl.dtx, l=lterror.dtx, m=ltpar.dtx, n=ltpara.dtx, o=ltspace.dtx, p=ltlogos.dtx, q=ltfiles.dtx, r=ltoutenc.dtx, s=ltcounts.dtx, t=ltlength.dtx, u=ltfssbas.dtx, v=ltfssaxes.dtx, w=ltfssstrc.dtx, x=ltfsscmp.dtx, y=ltfssdcl.dtx, z=ltfssini.dtx, A=fontdef.dtx, B=preload.dtx, C=ltfntcmd.dtx, D=lttextcomp.dtx, E=ltpageno.dtx, F=ltxref.dtx, G=ltmiscen.dtx, H=ltmath.dtx, I=ltlists.dtx, J=ltboxes.dtx, K=lttab.dtx, L=lt pictur.dtx, M=ltthm.dtx, N=ltsect.dtx, O=ltfloat.dtx, P=ltidxglo.dtx, Q=ltbibl.dtx, R=ltpage.dtx, S=ltclass.dtx, T=ltfilehook.dtx, U=ltshipout.dtx, V=ltoutput.dtx, W=lthyphen.dtx, X=ltfinal.dtx

- \tl_to_str:n 142,
152, g66, g77, g183, g187, g388,
g396, g409, g420, g424, g460, g554,
g578, g586, g601, g614, g626, g645,
g679, g1376, g1383, g1552, g1579,
g1659, g1660, g1726, g1739, g1767,
g1771, h232, h294, h474, h799,
h811, h868, i93, i98, i249, i266, T541
- \tl_trim_spaces:n 240, g179, g346,
g463, g960, g962, g1551, g1578,
g1680, g1881, h261, i177, T41, T42
- \tl_trim_spaces_apply:nN
..... g583, g1744, h184
- \tl_use:N h159, h763
- \tmspace 603, H201
- \to A440, A442
- \today a184, a188, a196, a199, N33
- token commands:
 - \c_space_token g1833
 - \token_if_active:NTF g1391
 - \token_if_cs:NTF 153, g1607
 - \token_if_eq_catcode:NNTF g1072
 - \token_if_eq_charcode:NNTF
..... g341, g1307, g1339, g1350
 - \token_if_eq_meaning:NNTF
..... g592, g595, g603,
g1847, g2224, g2226, i255, T419, T431
 - \token_if_macro:NTF 237, i64
 - \token_to_meaning:N ... i60, i252, i278
 - \token_to_str:N ... 134, 152, g66,
g74, g77, g346, g712, g934, g1085,
g1231, g1392, g1551, g1578, g1754,
g1767, g1771, g1780, g1821, g1822,
g1823, g1882, g2134, g2135, g2148,
g2149, g2186, g2187, g2200, g2201,
h229, i59, i60, i82, i271, i274, T239
- \toks 274, b31, b63, b95,
d36, n38, y485, y486, y496, y505, X621
- \toksdef b46, b63, b95, d222
- \tokszero d222
- \tolerance b317, u647, u692, u707, R58, R66
- \top A320
- \topfigrule V727, V2325
- \topfraction O273, V2291
- \topmargin V71, V624, V683
- \topmark V2209, V2218
- \topsep 615, H492, I2, I59
- \topskip
..... 303, b377, q59, q127, q182, V128, I1
- \totalheight J33, J34, J35
- totalpages 829
- trace commands:
 - trace_stack_levels X82
- \tracefloats V1897
- \tracefloatsoff V1897
- \tracefloatvals V1897
- \traceoff 261
- \tracelon 261
- \tracingall 261, b480
- \tracingassigns .. b498, b532, b549, b590
- \tracingcommands b496,
b514, b530, b539, b552, b576, b593
- \tracingfonts w17, w54,
w58, w86, w118, w153, w194, w224,
w238, w254, w260, w273, w280,
w287, w292, w301, w314, w322, w325
- \tracinggroups ... b488, b523, b561, b601
- \tracingifs b489, b524, b560, b600
- \tracinglostchars . b333, b481, b486,
b509, b521, b540, b564, b584, b604
- \tracingmacros b495,
b513, b529, b541, b563, b583, b603
- \tracingnesting .. b491, b526, b558, b598
- \tracingnone b545
- \tracingoff w118, w322
- \tracingon w119, w323
- \tracingonline
.. b475, b551, b575, b592, b622, u656
- \tracingoutput b476, b555, b579, b596, b619
- \tracingpages b485,
b508, b520, b540, b565, b585, b605
- \tracingparagraphs b487,
b510, b522, b541, b562, b582, b602
- \tracingrestores b497,
b515, b531, b541, b550, b581, b591
- \tracingscantokens
.. b490, b505, b525, b559, b573, b599
- \tracingstacklevels
..... 929, b336, b339, b341,
b342, b346, b350, b481, b493, b557
- \tracingstats b484,
b507, b519, b539, b566, b586, b606, X2
- \triangle A322
- \triangleleft A358, A482
- \triangleright A359, A482
- \TrimSpaces g2244
- trivlist (environment) 189
- \trivlist 622, G351, G401, G403,
G417, G439, H477, K78, M35, M37, 189
- \ttdefault z12,
z151, z193, z249, z273, z313, z341, A50
- \ttfamily 185, 492, z10, z11,
z242, z311, z312, z339, z340, C17, G461
- ttfamily z252
- \ttsubstdefault A20, A32, D32, D89
- \twocolumn V199
- \twocolumn[] 303
- \typein 75, 75

File Key: a=ltldirchk.dtx, b=ltplain.dtx, c=ltvers.dtx, d=ltluatex.dtx, e=ltexpl.dtx, f=ltdefs.dtx, g=ltcmd.dtx, h=lthooks.dtx, i=ltcmdhooks.dtx, j=ltalloc.dtx, k=ltcntrl.dtx, l=lterror.dtx, m=ltpar.dtx, n=ltpara.dtx, o=ltspace.dtx, p=ltlogos.dtx, q=ltfiles.dtx, r=ltoutenc.dtx, s=ltcounts.dtx, t=ltlength.dtx, u=ltfssbas.dtx, v=ltfssaxes.dtx, w=ltfssstrc.dtx, x=ltfsscmp.dtx, y=ltfssdcl.dtx, z=ltfssini.dtx, A=fontdef.dtx, B=preload.dtx, C=ltfntcmd.dtx, D=lttextcomp.dtx, E=ltpageno.dtx, F=ltxref.dtx, G=ltmiscen.dtx, H=ltmath.dtx, I=ltlists.dtx, J=ltboxes.dtx, K=lttab.dtx, L=ltpictur.dtx, M=ltthm.dtx, N=ltsect.dtx, O=ltfloat.dtx, P=ltidxglo.dtx, Q=ltbibl.dtx, R=ltpage.dtx, S=ltclass.dtx, T=ltfilehook.dtx, U=ltshipout.dtx, V=ltoutput.dtx, W=lthyphen.dtx, X=ltfinal.dtx

<code>\typein</code>	f31	r1096 , r1098 , r1099 , r1100 , r1101 ,
<code>\typeout</code>	75	r1102 , r1103 , r1104 , r1105 , r1106 ,
<code>\typeout</code>	95 , 270 , 415 , 828 , a73 ,	r1107 , r1108 , r1109 , r1110 , r1111 ,
	a116 , a172 , a197 , a199 , a211 , a226 ,	r1112 , r1114 , r1115 , r1116 , r1117 ,
	a233 , a244 , a257 , a270 , a283 , a321 ,	r1118 , r1119 , r1120 , r1121 , r1122 ,
	c21 , c38 , c43 , c48 , f3 , f36 , f43 , f567 ,	r1123 , r1124 , r1125 , r1126 , r1127 ,
	f568 , f601 , f602 , f607 , l74 , q200 ,	r1128 , r1129 , r1130 , r1131 , r1132 ,
	q575 , q576 , q582 , q616 , q654 , q664 ,	r1133 , r1134 , r1135 , r1136 , r1137 ,
	q667 , u366 , v478 , z73 , z416 , z597 ,	r1138 , r1139 , r1140 , r1141 , r1142 ,
	z607 , z617 , A9 , A125 , P8 , P25 ,	r1143 , r1144 , r1145 , r1146 , r1147 ,
	S309 , S325 , S337 , S1426 , S1625 ,	r1148 , r1149 , r1150 , r1151 , r1152 ,
	S1628 , U105 , U116 , U148 , V1898 ,	r1153 , r1154 , r1155 , r1156 , r1158 ,
	X262 , X575 , X582 , X594 , X595 , X603	r1160 , r1162 , r1163 , r1164 , r1165 ,
U		
<code>\u</code>	r237 , r386 ,	r1166 , r1167 , r1168 , r1169 , r1170 ,
	r472 , r589 , r596 , r616 , r623 , r754 ,	r1171 , r1172 , r1173 , r1174 , r1175 ,
	r1234 , r1309 , r1310 , r1325 , r1326 ,	r1176 , r1177 , r1178 , r1179 , r1180 ,
	r1335 , r1336 , r1349 , r1350 , r1351 ,	r1181 , r1182 , r1183 , r1184 , r1185 ,
	r1375 , r1376 , r1401 , r1402 , D163 , D194	r1186 , r1187 , r1188 , r1189 , r1190 ,
<code>\uccode</code>	X226 ,	r1191 , r1192 , r1193 , r1194 , r1195 ,
	X234 , X241 , X243 , X246 , X248 ,	r1196 , r1197 , r1198 , r1199 , r1200 ,
	X526 , X534 , X541 , X543 , X546 , X548	r1201 , r1202 , r1203 , r1205 , r1206 ,
<code>\Ucharcat</code>	z477	r1207 , r1218 , r1219 , r1220 , r1221 ,
<code>\uchyph</code>	b355	r1222 , r1223 , r1224 , r1225 , r1226 ,
<code>\ulcdefault</code>	v526 , v605	r1227 , r1228 , r1229 , r1230 , r1231 ,
<code>\ulcshape</code>	417 , v526 , v604 , v698 , v699 , z402 , C29	r1232 , r1233 , r1234 , r1235 , r1236 ,
<code>\Umathcode</code>	b127 , d30 , o476 , A15 , A50 , A65 ,	r1237 , r1238 , r1239 , r1240 , r1241 ,
	D159 , D367 , G467 , X146 , X337 , X520	r1242 , r1243 , r1244 , D376 , D377 ,
<code>\unboldmath</code>	z468	D378 , D379 , D380 , D381 , D382 ,
<code>\UndeclareTextCommand</code>	543 , r191 , D115 ,	D383 , D384 , D385 , D386 , D387 ,
	D117 , D119 , D308 , D585 , D1130 ,	D388 , D389 , D390 , D391 , D392 , D393
	D1131 , D1132 , D1133 , D1134 , D1135	<code>\UnicodeFontFile</code>
<code>\undefined</code>	a9 , a11 , a17 , a57 ,	r1037
	s92 , s93 , s94 , s95 , s96 , s97 , s98 , s99 ,	<code>\UnicodeFontName</code>
	A116 , A117 , X154 , X166 , X167 , X187	r1038
<code>\undefinedpagestyle</code>	R4 , R8	<code>\UnicodeFontTeXLigatures</code>
<code>\underbar</code>	b445 , f788 , f809	r993 , r1034
<code>\underbrace</code>	A540	<code>\unicoderead</code>
<code>\underline</code>	630	d143 , d157 , d158 , d159 , d160 , d165
<code>\underline</code>	b445 , J428 , J429	<code>uninstall</code>
<code>\unexpanded</code>	f584 ,	d832
	f595 , G191 , G237 , G254 , S1454 , S1456	<code>\unitlength</code>
<code>\unhcopy</code>	b447 , K345 , L742 , L798	826 , 827 ,
<code>\unicodedataline</code>	d143 , d146 , d160 , d161 , d162	843 , J53 , J64 , J73 , J83 , L5 , L29 ,
<code>\UnicodeEncodingName</code>	r981 , r987 , r1039 ,	L30 , L32 , L34 , L42 , L43 , L44 , L45 ,
	r1050 , r1054 , r1070 , r1072 , r1074 ,	L60 , L63 , L73 , L74 , L84 , L85 , L93 ,
	r1076 , r1077 , r1078 , r1079 , r1080 ,	L94 , L107 , L108 , L119 , L164 , L176 ,
	r1081 , r1082 , r1083 , r1084 , r1085 ,	L241 , L256 , L316 , L318 , L332 ,
	r1086 , r1087 , r1088 , r1089 , r1090 ,	L340 , L342 , L357 , L375 , L377 ,
	r1091 , r1092 , r1093 , r1094 , r1095 ,	L392 , L397 , L399 , L414 , L417 ,
		L422 , L480 , L481 , L508 , L509 ,
		L537 , L538 , L612 , L628 , L648 ,
		L654 , L690 , L691 , L693 , L694 ,
		L697 , L698 , L700 , L701 , L712 ,
		L713 , L715 , L716 , L718 , L719 ,
		L721 , L722 , L750 , L751 , L753 ,
		L754 , L757 , L758 , L760 , L761 ,
		L772 , L774 , L776 , L778 , U328 , X135
		<code>\unkern</code>
		u675
		<code>\unless</code>
		d151 , d159 , d161

File Key: a=ltldirchk.dtx, b=ltplain.dtx, c=ltvers.dtx, d=ltluatex.dtx, e=ltexpl.dtx, f=ltdefns.dtx, g=ltcmd.dtx, h=lt hooks.dtx, i=ltcmdhooks.dtx, j=ltalloc.dtx, k=ltcntrl.dtx, l=lterror.dtx, m=ltpar.dtx, n=lt para.dtx, o=lt space.dtx, p=lt logos.dtx, q=lt files.dtx, r=ltoutenc.dtx, s=lt counts.dtx, t=lt length.dtx, u=ltfssbas.dtx, v=ltfssaxes.dtx, w=ltfssstrc.dtx, x=ltfsscmp.dtx, y=ltfssdcl.dtx, z=ltfssini.dtx, A=fontdef.dtx, B=preload.dtx, C=ltfntcmd.dtx, D=lttextcomp.dtx, E=ltpageno.dtx, F=ltxref.dtx, G=ltmiscen.dtx, H=ltmath.dtx, I=ltlists.dtx, J=ltboxes.dtx, K=lttab.dtx, L=lt pictur.dtx, M=ltthm.dtx, N=ltsect.dtx, O=ltfloat.dtx, P=ltidnglo.dtx, Q=ltbibl.dtx, R=ltpage.dtx, S=ltclass.dtx, T=ltfilehook.dtx, U=ltshipout.dtx, V=ltoutput.dtx, W=lthyphen.dtx, X=ltfinal.dtx

<code>\unlhd</code>	z585	<code>\use_none:nnnn</code>	g1188
<code>\unpenalty</code>		<code>\use_none:nnnnn</code>	g1182
.. 271 , u678 , u682 , C116 , G433 , G455		<code>\use_none:nnnnnn</code>	g1176
<code>\unrhd</code>	z587	<code>\use_none:nnnnnnn</code>	g1170
<code>\unsetattribute</code>	42	<code>\use_none_delimit_by_q_recursion_</code>	
<code>\unsetattribute</code>	d82 , d232	stop:w	g269
<code>\unskip</code>	271	<code>\use_none_delimit_by_q_stop:w</code> ..	
<code>\unvcopy</code>	H180	g1654 , g1673 , g1677
<code>\Uparrow</code>	A584	<code>\usebox</code>	J156
<code>\uparrow</code>	A578	<code>\usecounter</code>	I225 , I238
<code>\upbracefill</code>	A543 , A560	<code>\usefont</code>	501 , r1562 ,
<code>\updefault</code>	512 , v688 ,	u80 , u282 , u652 , z562 , D7 , D613 , G471	
z21 , A94 , A101 , A103 , A111 , A113		<code>\UseHook</code>	166 , 169 , 177 , 179 , 180 ,
<code>\Updownarrow</code>	A588	190 , 191 , 231–233 , 243 , 803 , 805 ,	
<code>\updownarrow</code>	A582	811 , h1244 , h1282 , i165 , i169 , i271 ,	
<code>\uplus</code>	A378	i274 , q314 , q315 , q319 , q320 , q324 ,	
<code>\uppercase</code>	X558	q325 , w145 , z196 , z208 , z215 , z227 ,	
<code>\upshape</code>	417 , 419 , 496 , r442 ,	z234 , z240 , z245 , z250 , z516 , z533 ,	
r732 , r802 , r895 , v686 , v687 , z19 ,		G173 , G185 , G188 , G212 , G215 ,	
z20 , z402 , z413 , z446 , z504 , C24 , D629		G219 , G222 , S869 , S870 , S873 ,	
<code>\Upsilon</code>	A304	S874 , S896 , S897 , S900 , S901 , T170 ,	
<code>\upsilon</code>	A286	T171 , T174 , T175 , U119 , U169 , U373	
use commands:		<code>\UseLegacyTextSymbols</code>	D576 , D799
<code>\use:N</code>	g708 , h713 ,	<code>\UseOneTimeHook</code>	
h714 , h754 , T407 , T411 , T413 , T430		167 , 169 , 179 , h1244 , h1283 , q15 ,	
<code>\use:n</code>	g137 , g593 , g596 ,	q57 , q70 , G14 , G18 , G28 , G29 , G31	
g2221 , g2223 , g2228 , h236 , h896 ,		<code>\usepackage</code>	169 , 170 , 183 ,
i85 , i153 , i159 , i197 , i218 , i246 , i263		228 , 803–805 , 830 , S597 , S643 , S1441	
<code>\use:nn</code>	h684 , h912 , h1108	<code>\UseRawInputEncoding</code> .	X367 , X423 , X470
<code>\use:nnn</code>		<code>\UseTextAccent</code> .	330 , r149 , r150 , r188 ,
161 , 162 , g2134 , g2148 , g2186 , g2200		D110 , D111 , D113 , D156 , D158 ,	
<code>\use_i:nn</code>	g1043 , g1782	D939 , D1124 , D1125 , D1127 , D1128	
<code>\use_i:nnn</code>	207 , g1760	<code>\UseTextSymbol</code>	330 ,
<code>\use_i_delimit_by_q_recursion_</code>		r150 , r186 , D109 , D352 , D938 , D1007	
stop:nw	i225	<code>\usetikzlibrary</code>	170 , 228
<code>\use_i_delimit_by_q_stop:nw</code> ..	g355		
<code>\use_ii:nn</code>	143 , 207 ,		
g1027 , g1415 , g1428 , g1467 , g1485 ,			
g1510 , g1543 , g1570 , g1781 , g2232 , i37			
<code>\use_ii:nnn</code>	g1026 , g1484 , g1764 , g2225		
<code>\use_ii_iii:nnn</code> ...	T214 , T228 , T240		
<code>\use_iii:nn</code>	207		
<code>\use_iii:nnn</code>			
....	g1444 , g1759 , g1762 , g2227 , T47		
<code>\use_iii:nnnn</code>	g1443		
<code>\use_iv:nnnn</code>	g341 , g347		
<code>\use_none:n</code>	220 ,		
g94 , g597 , g1021 , g1022 , g1025 ,			
g1028 , g1042 , g1069 , g1206 , g1479 ,			
g1480 , g1483 , g1486 , g1716 , h7 ,			
h544 , h560 , h682 , i225 , T131 , T134 , U7			
<code>\use_none:nn</code>	g1200 ,		
g1438 , g1439 , g1442 , g1445 , h868 , h894			
<code>\use_none:nnn</code>	143 , g1194 , g1436		

V	
<code>\v</code>	r238 , r387 , r471 , r592 ,
r593 , r594 , r598 , r600 , r603 , r605 ,	
r607 , r613 , r619 , r620 , r621 , r625 ,	
r627 , r630 , r632 , r634 , r640 , r755 ,	
r1239 , r1319 , r1320 , r1321 , r1322 ,	
r1331 , r1332 , r1365 , r1366 , r1371 ,	
r1372 , r1383 , r1384 , r1391 , r1392 ,	
r1395 , r1396 , r1418 , r1419 , r1420 ,	
r1421 , r1422 , r1423 , r1424 , r1425 ,	
r1426 , r1427 , r1428 , r1437 , r1438 ,	
r1439 , r1440 , r1443 , r1444 , D165 , D195	
<code>\vadjust</code>	
265 , 267 , o37 , o59 , o71 , o100 , o107 ,	
o341 , o357 , o375 , o391 , O201 , O223	
<code>\valign</code>	D101 , D930
<code>\value</code>	366
<code>\value</code>	s14 , Q9

File Key: a=lt`dirchk`.dtx, b=lt`plain`.dtx, c=lt`vers`.dtx, d=lt`luatex`.dtx, e=lt`expl`.dtx, f=lt`defns`.dtx, g=lt`cmd`.dtx, h=lt`hooks`.dtx, i=lt`cmdhooks`.dtx, j=lt`alloc`.dtx, k=lt`cntrl`.dtx, l=lt`error`.dtx, m=lt`par`.dtx, n=lt`para`.dtx, o=lt`space`.dtx, p=lt`logos`.dtx, q=lt`files`.dtx, r=lt`outenc`.dtx, s=lt`counts`.dtx, t=lt`length`.dtx, u=lt`fssbas`.dtx, v=lt`fssaxes`.dtx, w=lt`fssstrc`.dtx, x=lt`fsscmp`.dtx, y=lt`fssdcl`.dtx, z=lt`fssini`.dtx, A=lt`fontdef`.dtx, B=lt`preload`.dtx, C=lt`fnfntcmd`.dtx, D=lt`textcomp`.dtx, E=lt`pageno`.dtx, F=lt`xref`.dtx, G=lt`miscen`.dtx, H=lt`math`.dtx, I=lt`lists`.dtx, J=lt`boxes`.dtx, K=lt`tab`.dtx, L=lt`ptictur`.dtx, M=lt`thm`.dtx, N=lt`tsect`.dtx, O=lt`float`.dtx, P=lt`idxglo`.dtx, Q=lt`bibl`.dtx, R=lt`page`.dtx, S=lt`class`.dtx, T=lt`filehook`.dtx, U=lt`shipout`.dtx, V=lt`output`.dtx, W=lt`thyphen`.dtx, X=lt`final`.dtx

- `\varbigtriangledown` A362, A365
`\varbigtriangleup` A363, A364
`\varepsilon` p15, A291
`\varphi` A296
`\varpi` A293
`\varrho` A294
`\varsigma` A295
`\vartheta` A292
`\vbadness` 841,
 b319, U212, U214, U263, U265, V2206
`\vbox` 266, 275, 825, 841
vbox commands:
 `\vbox_set_to_ht:Nnn` U215, U266
 `\vbox_to_zero:n` U326
 `\vbox_unpack:N` U226, U276
`\vdash` A404
`\vdots` A510
`\vec` A524
`\vector` l219, L233, L817, L834
`\vee` A367, A369
`\verb` G465, G490, G503, G509,
 G518, G524, G534, G539, G552, G554
verbatim (environment) G459
`\verbatim` G459
verbatim* (environment) G483
`\verbvisiblespace`
 G465, G467, G474, G478, G490, G495
`\Vert` A571, A573
`\vert` A576
`\vfil` b434, D102, D105, D931,
 D934, L566, L578, U384, U396,
 V175, V194, V412, V459, V627, V686
`\vfilneg` b434
`\vfuzz` 841, b363,
 R61, R68, U211, U213, U262, U264
`\vglue` b424
`\vline` K366
`\vphantom` r497, r513, H75
`\vrule` 920, b428, o452, r293,
 r295, r502, r518, w190, A558, A559,
 A561, A562, J165, J167, J218,
 J225, J427, J471, K186, K219,
 K347, K366, L227, L299, L302,
 L324, L333, L350, L359, L383,
 L391, L406, L413, L565, L578,
 L726, L782, V1851, V2226, V2259
`\vskip` 265, 284, 291
`\vspace` o330, o400, o401, o402
`\vsplit` V382, V429, V2208
- W**
- `\wedge` A366, A368
`\whatsit` 41, d186
- `\widehat` A527
`\widetilde` A526
`\widowpenalties` b106
`\widowpenalty` b326, u665
`\width` J30
`\wlog` 766,
 a100, b40, b145, b239, b254, b284,
 b299, d6, d7, d8, d54, S339, X46, X628
`\wp` A312
`\wr` A382
`\write` 184, 827, 837
- X**
- `\x` u333, u334, X316, X318
`\XeTeXcharclass` u640,
 X25, X33, X40, X53, X59, X68, X75
`\XeTeXcharclassCL` X159
`\XeTeXcharclassCM` X163
`\XeTeXcharclassEX` X160
`\XeTeXcharclassID` X157
`\XeTeXcharclassIS` X161
`\XeTeXcharclassNS` X162
`\XeTeXcharclassOP` X158
`\XeTeXcharglyph` r1035
`\XeTeXdashbreakstate` X259
`\XeTeXglyph` r1035
`\XeTeXintercharclasses` X153, X186
`\XeTeXinterchartoks` X154,
 X168, X169, X170, X171, X172,
 X173, X174, X175, X176, X177,
 X178, X179, X180, X181, X182,
 X187, X192, X193, X194, X195,
 X196, X197, X198, X199, X200,
 X201, X202, X203, X204, X205, X206
`\XeTeXmathcode` X147, X521
`\XeTeXrevision` X27
`\XeTeXuseglyphmetrics` X256, X258
`\XeTeXversion` z476, X27
`\Xi` A301
`\xi` A281
`\xtxHanGlue`
 X166, X190, X198, X199, X200,
 X201, X202, X203, X204, X205, X206
`\xtxHanSpace` X167, X191,
 X192, X193, X194, X195, X196, X197
- Y**
- `\year` .. a185, c11, c14, S1135, S1268, S1357
- Z**
- `\Z` X238, X517, X538
`\z` X229, X518, X529
`\zeta` A273

File Key: a=ltldirchk.dtx, b=ltplain.dtx, c=ltvers.dtx, d=ltluatex.dtx, e=ltexpl.dtx, f=ltdefns.dtx, g=ltcmd.dtx, h=lt hooks.dtx, i=ltcmdhooks.dtx, j=ltalloc.dtx, k=ltcntrl.dtx, l=lterror.dtx, m=ltpar.dtx, n=lt para.dtx, o=lt space.dtx, p=ltlogos.dtx, q=ltfiles.dtx, r=ltoutenc.dtx, s=ltcounts.dtx, t=ltlength.dtx, u=ltfssbas.dtx, v=ltfssaxes.dtx, w=ltfssstrc.dtx, x=ltfsscmp.dtx, y=ltfssdcl.dtx, z=ltfssini.dtx, A=fontdef.dtx, B=preload.dtx, C=ltfntcmd.dtx, D=lttextcomp.dtx, E=ltpageno.dtx, F=ltxref.dtx, G=ltmiscen.dtx, H=ltmath.dtx, I=ltlists.dtx, J=ltboxes.dtx, K=lttab.dtx, L=lt pictur.dtx, M=ltthm.dtx, N=ltsect.dtx, O=ltfloat.dtx, P=ltidxglo.dtx, Q=ltbibl.dtx, R=ltpage.dtx, S=ltclass.dtx, T=ltfilehook.dtx, U=ltshipout.dtx, V=ltoutput.dtx, W=lthyphen.dtx, X=ltfinal.dtx