The LaTeX 2ε Sources

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

2020-02-02 Patch level 5

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

Contents

File a
ltdirchk.dtx

1 LaTeX System Dependent Initialisations

This file implements the semi-automatic determination of various system dependent parts of the initialisation. 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, 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, `/`.

`\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 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 TeX 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 initialisation tests in this file. You should not need to set it in `texsys.cfg`, however the following documentation

is left for information. LaTeX 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 LaTeX to take appropriate action for these old TeXs.

If your TeX is older than version 3.141, then you should define `\@TeXversion` (using `\def`) to be the version number. If you do not do this[1] , LaTeX will not work around a bug in old TeX 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.
 ...

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

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

```
! LaTeX Error: \rubbish undefined.

See the LaTeX manual or LaTeX Companion for explanation.
Type  H <return>  for immediate help.
! .
 ...

l.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 TeX, but this should not cause any confusion we hope.

2 Initialisation

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

```
⟨*dircheck⟩
⟨*initex⟩
⟨initex⟩\ifnum\catcode`\{=1
⟨initex⟩  \errmessage
⟨initex⟩  {LaTeX must be made using an initex with no format preloaded}
```

[1] Actually if your TeX is really old, version 2, LaTeX can detect this, and sets `\@TeXversion` to 2 if it is not set in the `cfg` file.

```
⟨initex⟩\fi
\catcode‘\{=1
\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!

```
\ifx\directlua\undefined
\else
  \ifx\luatexversion\undefined
```

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

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

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

```
⟨/initex⟩
⟨/dircheck⟩
⟨*initex, latexrelease⟩
⟨latexrelease⟩\ifx\directlua\undefined\else
⟨latexrelease⟩\IncludeInRelease{2015/10/01}{\luatexluafunction}
⟨latexrelease⟩                            {LuaTeX (prefixed names)}%
    \directlua{tex.enableprimitives("",%
                 tex.extraprimitives("omega", "aleph", "luatex"))}
⟨latexrelease⟩\EndIncludeInRelease
⟨latexrelease⟩\IncludeInRelease{0000/00/00}{\luatexluafunction}
⟨latexrelease⟩                            {LuaTeX (prefixed names)}%
⟨latexrelease⟩\directlua{
⟨latexrelease⟩  tex.enableprimitives(
⟨latexrelease⟩    "luatex",
⟨latexrelease⟩    tex.extraprimitives("core","omega", "aleph", "luatex")
⟨latexrelease⟩  )
⟨latexrelease⟩  local i
⟨latexrelease⟩  local t = { }
⟨latexrelease⟩  for _,i in pairs(tex.extraprimitives("luatex")) do
⟨latexrelease⟩    if not string.match(i,"^U") then
⟨latexrelease⟩      if not string.match(i, "^luatex") then
⟨latexrelease⟩        table.insert(t,i)
⟨latexrelease⟩      end
⟨latexrelease⟩    else
⟨latexrelease⟩      if string.match(i,"^Uchar$") then
⟨latexrelease⟩        table.insert(t,i)
⟨latexrelease⟩      end
⟨latexrelease⟩    end
⟨latexrelease⟩  end
⟨latexrelease⟩  for _,i in pairs(t) do
⟨latexrelease⟩    tex.print(
⟨latexrelease⟩      "\noexpand\\let\noexpand\\" .. i
⟨latexrelease⟩        .. "\noexpand\\undefined"
⟨latexrelease⟩    )
⟨latexrelease⟩  end
⟨latexrelease⟩}
⟨latexrelease⟩\EndIncludeInRelease
```

```
⟨latexrelease⟩\fi
⟨/initex, latexrelease⟩
⟨*dircheck⟩
⟨*initex⟩
  \fi
\fi
```

A test can now be made for eTEX.

```
⟨initex⟩\ifx\eTeXversion\undefined
⟨initex⟩  \errmessage
⟨initex⟩    {LaTeX requires e-TeX}
⟨initex⟩  \expandafter\endinput
⟨initex⟩\fi
```

That distraction over, back to the basics of a format.

```
\catcode`\#=6
\catcode`\^=7
\chardef\active=13
\catcode`\@=11
\countdef\count@=255
\let\bgroup={ \let\egroup=}
\ifx\@@input\@undefined\let\@@input\input\fi
\ifx\@@end\@undefined\let\@@end\end\fi
\chardef\@inputcheck0
\chardef\sixt@@n=16
\newlinechar`\^^J
\def\typeout{\immediate\write17}
\def\dospecials{\do\ \do\\\do\{\do\}\do\$\do\&%
  \do\#\do\^\do\_\do\%\do\~}
\def\@makeother#1{\catcode`#1=12\relax}
\def\space{ }
\def\@tempswafalse{\let\if@tempswa\iffalse}
\def\@tempswatrue{\let\if@tempswa\iftrue}
\let\if@tempswa\iffalse
\def\loop#1\repeat{\def\iterate{#1\relax\expandafter\iterate\fi}%
  \iterate \let\iterate\relax}
\let\repeat\fi
⟨/initex⟩
```

2.2 Some bits of 2e

```
⟨*2ekernel⟩
\def\two@digits#1{\ifnum#1<10 0\fi\number#1}
\long\def\@firstoftwo#1#2{#1}
\long\def\@secondoftwo#1#2{#2}
```

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

```
\def\ProvidesFile#1{%
  \begingroup
    \catcode`\ 10 %
    \ifnum \endlinechar<256 %
      \ifnum \endlinechar>\m@ne
        \catcode\endlinechar 10 %
      \fi
    \fi
    \@makeother\/%
    \@ifnextchar[{\@providesfile{#1}}{\@providesfile{#1}[]}}
\def\@providesfile#1[#2]{%
    \wlog{File: #1 #2}%
    \@addtofilelist{ #2}%
    \endgroup}
\long\def\@addtofilelist#1{}
\def\@empty{}
\catcode`\%=12
\def\@percentchar{%}
\catcode`\%=14
\let\@currdir\@undefined
\let\input@path\@undefined
\let\filename@parse\@undefined
```

\strip@prefix

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

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.

```
<*docstrip>
\openin15=texsys.cfg
\ifeof15
\typeout{** Writing a default texsys.cfg}
\immediate\openout15=texsys.cfg
\begingroup
\catcode`\^^M\active%
\let^^M\par%
\def\reserved@a#1^^M{%
 \def\reserved@b{#1}%
 \ifx\reserved@b\reserved@c\endgroup\else%
     \immediate\write15{#1}%
     \expandafter\reserved@a\fi}%
\def\reserved@d#1START^^M{\let\do\@makeother\dospecials\reserved@a}%
\catcode`\%=12
\def\reserved@c{%END}
\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, TeXwill 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 willcover 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.

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

```
% \def\@currdir{./}
% \def\input@path{%
%   {/usr/local/lib/tex/inputs/distrib/}%
%   {/usr/local/lib/tex/inputs/contrib/}%
%   {/usr/local/lib/tex/inputs/local/}%
% }
```

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.

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

```
% \def\@currdir{./}
% \def\input@path{%
%   {c:/tex/inputs/distrib/}%
%   {c:/tex/inputs/contrib/}%
%   {c:/tex/inputs/local/}%
% }
```

3.6 VMS (DECUS TeX, 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.

```
% \def\@currdir{[]}
% \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:

```
% \def\@currdir{[]}
% \def\input@path{%
%   {tex_inputs:}%
%   {SOMEDISK:[SOME.TEX.DIRECTORY]}%
% }
```

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.

```
% \def\@currdir{:}
% \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.

```
% \def\@currdir{:}
% \def\input@path{%
%   {Hard-Disk:Applications:TeX:TeX-inputs:}%
%   {Hard-Disk:Applications:TeX:My-inputs:}%
% }
```

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:

```
% \def\@dir#1#2 {%
%   \@d@r{#1}#2..\@nil}
% \def\@d@r#1#2.#3.#4\@nil{%
%   <\ifx\@dir#1\@dir\else#1\ifx\@dir#3\@dir\else.\fi\fi#3>#2 }
%
% \def\@currdir{\@dir{}}
% \def\input@path{%
%   {\@dir{area.one}}%
%   {\@dir{area.two}}%
% }
```

END

```
\immediate\closeout15
```

If texsys.cfg did exist, then input it.

```
\else
\typeout{** Using the existing texsys.cfg}
\closein15
\input texsys.cfg
\fi
⟨/docstrip⟩
```

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

```
⟨dircheck⟩\input texsys.cfg
```

4 Setting \@currdir

\@currdir
\IfFileExists

This is a local definition of \IfFileExists. It tries to relocate texsxys.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.

```
\begingroup
\count@\time
\divide\count@ 60
\count2=-\count@
\multiply\count2 60
\advance\count2 \time
```

\today

The current date and time stamp.

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

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

```
\immediate\openout15=texsys.aux
\immediate\write15{\today^^J}
\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!

```
\def\IfFileExists#1#2#3{%
  \openin\@inputcheck#1 %
  \ifeof\@inputcheck
     #3\relax
  \else
    \read\@inputcheck to \reserved@a
    \ifx\reserved@a\today
      \typeout{#1 found}#2\relax
    \else
      \typeout{BAD: old file \reserved@a (should be \today)}%
      #3\relax
    \fi
  \fi
  \closein\@inputcheck}
```

```
\endlinechar=-1
```

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

```
\ifx\@currdir\@undefined
  \IfFileExists{./texsys.aux}{\gdef\@currdir{./}}%
    {\IfFileExists{[]texsys.aux}{\gdef\@currdir{[]}}%
      {\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.

```
  \ifx\@currdir\@undefined
    \global\let\@currdir\@empty
    \typeout{^^J^^J%
      !! No syntax for the current directory could be found^^J%
      }%
  \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.

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

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

```
\fi
```

```
\immediate\closeout15 %
\endgroup
```

```
\typeout{^^J^^J%
         \noexpand\@currdir set to:
          \expandafter\strip@prefix\meaning\@currdir.^^J%
         }
```

Stop here if the file is being used unstripped.

```
⟨*docstrip⟩
\relax\endinput
⟨/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 LaTeX 2$_\varepsilon$ 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 LaTeX 2$_\varepsilon$. This will check, amongst 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.

```
  \typeout{^^J%
    Assuming \noexpand\openin and \noexpand\input^^J%
    \ifx\input@path\@undefined
```

`\input@path` has not been pre-defined.

```
      have the same search path.^^J%
    \else
```

`\input@path` has been defined in `texsys.cfg`.

```
      have different  search paths.^^J%
      LaTeX will use the path specified by \noexpand\input@path:^^J%
    \fi
    }
```

6 Filename Parsing

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

```
\ifx\filename@parse\@undefined
  \def\reserved@a{./}\ifx\@currdir\reserved@a
```

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

```
    \typeout{^^JDefining UNIX/DOS style filename parser.^^J}
    \def\filename@parse#1{%
      \let\filename@area\@empty
      \expandafter\filename@path#1/\\}
```

Search for the last `/`.

```
    \def\filename@path#1/#2\\{%
      \ifx\\#2\\%
         \def\reserved@a{\filename@simple#1.\\}%
      \else
         \edef\filename@area{\filename@area#1/}%
         \def\reserved@a{\filename@path#2\\}%
      \fi
      \reserved@a}
```

```
  \else\def\reserved@a{[]}\ifx\@currdir\reserved@a
```

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

```
    \typeout{^^JDefining VMS style filename parser.^^J}
    \def\filename@parse#1{%
      \let\filename@area\@empty
      \expandafter\filename@path#1]\\}
```

Search for the last `]`.

```
    \def\filename@path#1]#2\\{%
      \ifx\\#2\\%
         \def\reserved@a{\filename@simple#1.\\}%
      \else
         \edef\filename@area{\filename@area#1]}%
         \def\reserved@a{\filename@path#2\\}%
      \fi
      \reserved@a}
```

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

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

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

Search for the last `:`.

```
    \def\filename@path#1:#2\\{%
      \ifx\\#2\\%
         \def\reserved@a{\filename@simple#1.\\}%
      \else
         \edef\filename@area{\filename@area#1:}%
         \def\reserved@a{\filename@path#2\\}%
      \fi
      \reserved@a}
```

```
  \else
```

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

```
    \typeout{^^JDefining generic filename parser.^^J}
    \def\filename@parse#1{%
      \let\filename@area\@empty
      \expandafter\filename@simple#1.\\}
  \fi\fi\fi
```

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

```
⟨/dircheck⟩
⟨*dircheck, latexrelease⟩
⟨latexrelease⟩\IncludeInRelease{2019/10/01}{\filename@simple}
⟨latexrelease⟩                              {Final dot for extension}%
\def\filename@simple#1.#2\\{%
  \ifx\\#2\\%
    \let\filename@ext\relax
    \edef\filename@base{#1}%
  \else
    \filename@dots{#1}#2\\%
  \fi}
```

```
\def\filename@dots#1#2.#3\\{%
  \ifx\\#3\\%
    \def\filename@ext{#2}%
    \edef\filename@base{#1}%
  \else
    \filename@dots{#1.#2}#3\\%
  \fi}
⟨latexrelease⟩\EndIncludeInRelease
⟨latexrelease⟩\IncludeInRelease{0000/00/00}{\filename@simple}
⟨latexrelease⟩                                 {Final dot for extension}%
⟨latexrelease⟩  \def\filename@simple#1.#2\\{%
⟨latexrelease⟩    \ifx\\#2\\%
⟨latexrelease⟩      \let\filename@ext\relax
⟨latexrelease⟩    \else
⟨latexrelease⟩      \edef\filename@ext{\filename@dot#2\\}%
⟨latexrelease⟩    \fi
⟨latexrelease⟩    \edef\filename@base{#1}}
⟨latexrelease⟩\EndIncludeInRelease
⟨/dircheck, latexrelease⟩
⟨*dircheck⟩
```

Remove a final dot, added earlier.

```
  \def\filename@dot#1.\\{#1}
```

```
\else
```

Otherwise, `\filename@parse` was specified in `texsys.cfg`.

```
  \typeout{^^J^^J%
    \noexpand\filename@parse was defined in texsys.cfg:^^J%
    \expandafter\strip@prefix\meaning\filename@parse.^^J%
    }
\fi
```

7 TEX Versions

`\@TeXversion` TEX 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 TEX3.14 would be detected as TEX3, but LATEX only needs to take account of TEX's older than 3, or between 3 and 3.14.

```
\ifx\@TeXversion\@undefined
  \ifx\@undefined\inputlineno
    \def\@TeXversion{2}
  \else
   {\catcode`\^^J=\active
    \def\reserved@a#1#2\@@{\if#1\string^3\fi}
    \edef\reserved@a{\expandafter\reserved@a\string^^J\@@}
    \ifx\reserved@a\@empty\else\gdef\@TeXversion{3}\fi}
  \fi
\fi
```

```
⟨/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
ltplain.dtx

9 Plain TeX

LaTeX includes almost all of the functionality of Knuth's original 'Basic Macros' That is, the plain TeX format described in Appendix B of the TeXBook. 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.

LaTeX font definitions are done using NFSS2 so none of PLAIN's font definitions are in LaTeX.

LaTeX has its own tabbing environment, so PLAIN's is disabled.

LaTeX uses its own output routine, so most of the plain one was removed.

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

```
\message{catcodes,}
```

We had to define the `\catcodes` right away, before the message line, since `\message` uses the { and } characters. When INITEX (the TeX 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:

```
\def\dospecials{\do\ \do\\\do\{\do\}\do\$\do\&%
  \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.

```
\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` `\tw@` `\thr@@` `\sixt@@n` `\@cclv`

Small constants are defined using `\chardef`.

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

`\@cclvi` `\@m` `\@M` `\@MM`

Constants above 255 defined using `\mathchardef`.

```
\mathchardef\@cclvi=256
\mathchardef\@m=1000
\mathchardef\@M=10000
\mathchardef\@MM=20000
```

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.

```
\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 LaTeX 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 LaTeX 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, `\box`255 is reserved for `\output`; `\count`255, `\dimen`255, and `\skip`255 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.

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

`\insc@unt`
`\allocationnumber`

The insertion counter and most recent allocation.

```
\countdef\insc@unt=20
\countdef\allocationnumber=21
```

`\m@ne`

The constant −1.

```
\countdef\m@ne=22 \m@ne=-1
```

`\wlog`

Write on log file (only)

```
\def\wlog{\immediate\write\m@ne}
```

`\count@`
`\dimen@`
`\dimen@i`
`\dimen@ii`
`\skip@`
`\toks@`

Here are abbreviations for the names of scratch registers that don't need to be allocated.

```
\countdef\count@=255
\dimendef\dimen@=0
\dimendef\dimen@i=1 % global only
\dimendef\dimen@ii=2
\skipdef\skip@=0
\toksdef\toks@=0
```

`\newcount`
`\newdimen`
`\newskip`
`\newmuskip`
`\newbox`
`\newtoks`
`\newread`
`\newwrite`
`\newfam`
`\newlanguage`

Now, we define `\newcount`, `\newbox`, etc. so that you can say `\newcount\foo` and `\foo` will be defined (with `\countdef`) to be the next counter.

To find out which counter `\foo` is, you can look at `\allocationnumber`.

Since there's no `\boxdef` command, `\chardef` is used to define a `\newbox`, `\newinsert`, `\newfam`, and so on.

LaTeX change: remove `\outer` from `\newcount` and `\newdimen` (FMi) This is necessary to use `\newcount` inside `\if...` later on. Also remove from `\newskip`, `\newbox` `\newwrite` and `\newfam` (DPC) to save later redefinition.

```
⟨/2ekernel⟩
⟨*2ekernel | latexrelease⟩
⟨latexrelease⟩\IncludeInRelease{2015/01/01}%
⟨latexrelease⟩                 {\newcount}{Extended Allocation}%
```

```
\def\newcount {\e@alloc\count \countdef {\count10}\insc@unt\float@count}
\def\newdimen {\e@alloc\dimen \dimendef {\count11}\insc@unt\float@count}
\def\newskip  {\e@alloc\skip  \skipdef  {\count12}\insc@unt\float@count}
\def\newmuskip
          {\e@alloc\muskip\muskipdef{\count13}\m@ne\e@alloc@top}
```

For compatibility use `\chardef` in the classical range.

```
\def\newbox   {\e@alloc\box
                 {\ifnum\allocationnumber<\@cclvi
                    \expandafter\chardef
                  \else
                    \expandafter\e@alloc@chardef
                  \fi}
                                      {\count14}\insc@unt\float@count}
\def\newtoks  {\e@alloc\toks \toksdef{\count15}\m@ne\e@alloc@top}
\def\newread  {\e@alloc\read \chardef{\count16}\m@ne\sixt@@n}
```

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

```
\ifx\directlua\@undefined
  \def\newwrite   {\e@alloc\write \chardef{\count17}\m@ne\sixt@@n}
\else
  \def\newwrite   {\e@alloc\write
                   {\ifnum\allocationnumber=18
                     \advance\count17\@ne
                     \allocationnumber\count17 %
                   \fi
                   \global\chardef}%
                  {\count17}%
                  \m@ne
                  {128}}
\fi
```

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

```
\ifx\directlua\@undefined
  \def\newlanguage  {\e@alloc\language \chardef{\count19}\m@ne\@cclvi}
\else
  \def\newlanguage  {\e@alloc\language \chardef{\count19}\m@ne{16384}}
\fi
⟨/2ekernel | latexrelease⟩
```

```
⟨latexrelease⟩\EndIncludeInRelease
⟨latexrelease⟩\IncludeInRelease{0000/00/00}%
⟨latexrelease⟩                 {\newcount}{Extended Allocation}%
⟨latexrelease⟩\def\newcount{\alloc@0\count\countdef\insc@unt}
⟨latexrelease⟩\def\newdimen{\alloc@1\dimen\dimendef\insc@unt}
```

```
⟨latexrelease⟩\def\newskip{\alloc@2\skip\skipdef\insc@unt}
⟨latexrelease⟩\def\newmuskip{\alloc@3\muskip\muskipdef\@cclvi}
⟨latexrelease⟩\def\newbox{\alloc@4\box\chardef\insc@unt}
⟨latexrelease⟩\def\newtoks{\alloc@5\toks\toksdef\@cclvi}
⟨latexrelease⟩\def\newread{\alloc@6\read\chardef\sixt@@n}
⟨latexrelease⟩\def\newwrite{\alloc@7\write\chardef\sixt@@n}
⟨latexrelease⟩\def\new@mathgroup{\alloc@8\fam\chardef\sixt@@n}
⟨latexrelease⟩\def\newlanguage{\alloc@9\language\chardef\@cclvi}
⟨latexrelease⟩\let\newfam\new@mathgroup
⟨latexrelease⟩\EndIncludeInRelease
```

\e@alloc@chardef
\e@alloc@top

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

```
⟨*2ekernel | latexrelease⟩
⟨latexrelease⟩\IncludeInRelease{2015/01/01}%
⟨latexrelease⟩                 {\e@alloc@chardef}{Extended Allocation}%
```

```
\ifx\directlua\@undefined
  \ifx\widowpenalties\@undefined
```

classic tex has 2^8 registers.

```
    \mathchardef\e@alloc@top=255
    \let\e@alloc@chardef\chardef
  \else
```

etex and xetex have 2^{15} registers.

```
    \mathchardef\e@alloc@top=32767
    \let\e@alloc@chardef\mathchardef
  \fi
\else
```

luatex has 2^{16} registers.

```
  \chardef\e@alloc@top=65535
  \let\e@alloc@chardef\chardef
\fi
```

```
⟨/2ekernel | latexrelease⟩
⟨latexrelease⟩\EndIncludeInRelease
⟨latexrelease⟩\IncludeInRelease{0000/00/00}%
⟨latexrelease⟩                 {\e@alloc@chardef}{Extended Allocation}%
⟨latexrelease⟩\let\e@alloc@top\@undefined
⟨latexrelease⟩\let\e@alloc@chardef\@undefined
⟨latexrelease⟩\EndIncludeInRelease
```

\e@mathgroup@top

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

```
⟨*2ekernel | latexrelease⟩
⟨latexrelease⟩\IncludeInRelease{2015/01/01}%
⟨latexrelease⟩                 {\e@mathgroup@top}{Extended Allocation}%
```

```
\ifx\Umathcode\@undefined
```

classic and e tex have 16 fam (0–15).

```
  \chardef\e@mathgroup@top=16
\else
```

xetex and luatex have 256 fam (0–255).

```
  \chardef\e@mathgroup@top=256
\fi
```

```
⟨/2ekernel | latexrelease⟩
⟨latexrelease⟩\EndIncludeInRelease
⟨latexrelease⟩\IncludeInRelease{0000/00/00}%
⟨latexrelease⟩                 {\e@mathgroup@top}{Extended Allocation}%
⟨latexrelease⟩\let\e@mathgroup@top\@undefined
⟨latexrelease⟩\EndIncludeInRelease
```

\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.

```
⟨*2ekernel | latexrelease⟩
⟨latexrelease⟩\IncludeInRelease{2015/01/01}{\e@alloc}{Extended Allocation}%
```

```
\def\e@alloc#1#2#3#4#5#6{%
  \global\advance#3\@ne
  \e@ch@ck{#3}{#4}{#5}#1%
  \allocationnumber#3\relax
  \global#2#6\allocationnumber
  \wlog{\string#6=\string#1\the\allocationnumber}}%
```

```
⟨/2ekernel | latexrelease⟩
⟨latexrelease⟩\EndIncludeInRelease
⟨latexrelease⟩\IncludeInRelease{0000/00/00}{\e@alloc}{Extended Allocation}%
⟨latexrelease⟩\let\e@alloc\@undefined
⟨latexrelease⟩\EndIncludeInRelease
⟨*2ekernel⟩
```

\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.

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

```
⟨/2ekernel⟩
⟨*2ekernel | latexrelease⟩
⟨latexrelease⟩\IncludeInRelease{2015/10/01}
⟨latexrelease⟩                 {\e@ch@ck}{Extended Allocation (checking)}%
```

```
\gdef\e@ch@ck#1#2#3#4{%
  \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).

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

Check we are below the extended limit.

```
    \ifnum#1<#3\relax
    \else
      \errmessage{No room for a new \string#4}%
    \fi
  \fi}%
⟨latexrelease⟩\EndIncludeInRelease
⟨latexrelease⟩\IncludeInRelease{2015/01/01}%
⟨latexrelease⟩                 {\e@ch@ck}{Extended Allocation (checking)}%
⟨latexrelease⟩\gdef\e@ch@ck#1#2#3#4{%
⟨latexrelease⟩  \ifnum#1<#2\else
⟨latexrelease⟩    \ifnum#1=#2\relax
⟨latexrelease⟩      #1\@cclvi
⟨latexrelease⟩      \ifx\count#4\advance#1 10 \fi
⟨latexrelease⟩    \fi
⟨latexrelease⟩    \ifnum#1<#3\relax
⟨latexrelease⟩    \else
⟨latexrelease⟩      \errmessage{No room for a new #4}%
⟨latexrelease⟩    \fi
⟨latexrelease⟩  \fi}%
⟨latexrelease⟩\EndIncludeInRelease
⟨latexrelease⟩\IncludeInRelease{0000/00/00}%
⟨latexrelease⟩                 {\e@ch@ck}{Extended Allocation (checking)}%
⟨latexrelease⟩\let\e@ch@ck\@undefined
⟨latexrelease⟩\EndIncludeInRelease
```

```
⟨latexrelease⟩\IncludeInRelease{2015/01/01}%
⟨latexrelease⟩                 {\extrafloats}{Extra floats}%
```

```
\let\float@count\e@alloc@top
```

\extrafloats

```
\ifx\numexpr\@undefined
```

In classic TeX use `\newinsert` to allocate float boxes.

```
\def\extrafloats#1{%
\count@#1\relax
\ifnum\count@>\z@
\newinsert\reserved@a
\global\expandafter\chardef
            \csname bx@\the\allocationnumber\endcsname\allocationnumber
\@cons\@freelist{\csname bx@\the\allocationnumber\endcsname}%
\advance\count@\m@ne
\expandafter\extrafloats
\expandafter\count@
\fi
}%
```

```
\else
```

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

```
\def\extrafloats#1{%
\ifnum#1>\z@
\count@\numexpr\float@count-1\relax
  \ch@ck0\count@\count
  \ch@ck1\count@\dimen
  \ch@ck2\count@\skip
  \ch@ck4\count@\box
\global\e@alloc@chardef\float@count\count@
\global\expandafter\e@alloc@chardef
           \csname bx@\the\float@count\endcsname\float@count
\@cons\@freelist{\csname bx@\the\float@count\endcsname}%
\expandafter
\extrafloats\expandafter{\numexpr#1-1\relax}%
\fi}%
```

```
\fi
```

```
⟨/2ekernel | latexrelease⟩
⟨latexrelease⟩\EndIncludeInRelease
⟨latexrelease⟩\IncludeInRelease{0000/00/00}%
⟨latexrelease⟩                 {\extrafloats}{Extra floats}%
⟨latexrelease⟩\let\float@count\@undefined
⟨latexrelease⟩\let\extrafloats\@undefined
⟨latexrelease⟩\EndIncludeInRelease
⟨*2ekernel⟩
```

\alloc@

```
\def\alloc@#1#2#3#4#5{\global\advance\count1#1\@ne
  \ch@ck#1#4#2%
  \allocationnumber\count1#1%
  \global#3#5\allocationnumber
  \wlog{\string#5=\string#2\the\allocationnumber}}
```

\newinsert

```
⟨/2ekernel⟩
⟨*2ekernel | latexrelease⟩
⟨latexrelease⟩\IncludeInRelease{2015/10/01}
⟨latexrelease⟩                 {\newinsert}{Extended \newinsert}%
```

```
\ifx\numexpr\@undefined
```

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

```
\def\newinsert#1{\global\advance\insc@unt \m@ne
  \ch@ck0\insc@unt\count
  \ch@ck1\insc@unt\dimen
  \ch@ck2\insc@unt\skip
  \ch@ck4\insc@unt\box
  \allocationnumber\insc@unt
  \global\chardef#1\allocationnumber
  \wlog{\string#1=\string\insert\the\allocationnumber}}
```

```
\else
```

The highest register allowed with `\insert`.

```
\ifx\directlua\@undefined
  \chardef\e@insert@top255
\else
  \chardef\e@insert@top\e@alloc@top
\fi
```

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

```
\def\newinsert#1{%
\@tempswafalse
\global\advance\insc@unt\m@ne
\ifnum\count10<\insc@unt
\ifnum\count11<\insc@unt
\ifnum\count12<\insc@unt
\ifnum\count14<\insc@unt
  \@tempswatrue
\fi\fi\fi\fi
\if@tempswa
\allocationnumber\insc@unt
\else
\global\advance\insc@unt\@ne
  \extrafloats\@ne
  \@next\@currbox\@freelist
    {\ifnum\@currbox<\e@insert@top
      \allocationnumber\@currbox
     \else
     \ch@ck0\m@ne\insert
     \fi}%
     {\ch@ck0\m@ne\insert}%
\fi
\global\chardef#1\allocationnumber
\wlog{\string#1=\string\insert\the\allocationnumber}%
}
```

```
\fi
⟨/2ekernel | latexrelease⟩
```

```
⟨latexrelease⟩\EndIncludeInRelease
⟨latexrelease⟩\IncludeInRelease{0000/00/00}%
⟨latexrelease⟩                 {\newinsert}{Extended \newinsert}%
⟨latexrelease⟩\let\e@insert@top\@undefined
⟨latexrelease⟩\def\newinsert#1{\global\advance\insc@unt \m@ne
⟨latexrelease⟩  \ch@ck0\insc@unt\count
⟨latexrelease⟩  \ch@ck1\insc@unt\dimen
⟨latexrelease⟩  \ch@ck2\insc@unt\skip
⟨latexrelease⟩  \ch@ck4\insc@unt\box
⟨latexrelease⟩  \allocationnumber\insc@unt
⟨latexrelease⟩  \global\chardef#1\allocationnumber
⟨latexrelease⟩  \wlog{\string#1=\string\insert\the\allocationnumber}}
⟨latexrelease⟩\EndIncludeInRelease
⟨*2ekernel⟩
```

\ch@ck

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

\newhelp

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

\maxdimen
\hideskip

Here are some examples of allocation.

```
\newdimen\maxdimen \maxdimen=16383.99999pt % the largest legal <dimen>
\newskip\hideskip \hideskip=-1000pt plus 1fill % negative but can grow
```

\p@
\z@
\z@skip
\voidb@x

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

Assign initial values to TEX's parameters

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

All of TEX'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 LATEX 2.09 comments (not necessarily accurate any more):

```
\pretolerance=100
\tolerance=200 % INITEX sets this to 10000
\hbadness=1000
\vbadness=1000
\linepenalty=10
\hyphenpenalty=50
\exhyphenpenalty=50
\binoppenalty=700
\relpenalty=500
\clubpenalty=150
\widowpenalty=150
\displaywidowpenalty=50
\brokenpenalty=100
\predisplaypenalty=10000
```

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

```
\doublehyphendemerits=10000
\finalhyphendemerits=5000
\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

```
\tracinglostchars=1
```

\tracingcommands=0
\tracingrestores=0
\language=0

```
\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
```

```
\defaulthyphenchar=`\-
\defaultskewchar=-1
```

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

```
\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.

```
\showboxbreadth=-1
\showboxdepth=-1
\errorcontextlines=-1
```

```
\hfuzz=0.1pt
\vfuzz=0.1pt
\overfullrule=5pt
\maxdepth=4pt
\splitmaxdepth=\maxdimen
\boxmaxdepth=\maxdimen
```

Historical LaTeX 2.09 comments (not necessarily accurate any more):

```
\lineskiplimit=0pt, changed by \normalbaselines
```

```
\delimitershortfall=5pt
\nulldelimiterspace=1.2pt
\scriptspace=0.5pt
```

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

```
\parindent=20pt
```

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

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

```
\parskip=0pt plus 1pt
```

```
\abovedisplayskip=12pt plus 3pt minus 9pt
\abovedisplayshortskip=0pt plus 3pt
\belowdisplayskip=12pt plus 3pt minus 9pt
\belowdisplayshortskip=7pt plus 3pt minus 4pt
```

```
\leftskip=0pt
\rightskip=0pt
```

```
\topskip=10pt
\splittopskip=10pt
```

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

```
\parfillskip=0pt plus 1fil
```

End of historical LaTeX 2.09 comments.

\normalbaselineskip
\normallineskip
\normallineskiplimit

We also define special registers that function like parameters:

```
\newskip\normalbaselineskip \normalbaselineskip=12pt
\newskip\normallineskip \normallineskip=1pt
\newdimen\normallineskiplimit \normallineskiplimit=0pt
```

\interfootlinepenalty

```
\newcount\interfootnotelinepenalty \interfootnotelinepenalty=100
```

Definitions for preloaded fonts

\magstephalf
\magstep

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

Macros for setting ordinary text

\frenchspacing
\nonfrenchspacing

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

\normalbaselines

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

\M
\I

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

```
\def\^^M{\ } % control <return> = control <space>
\let\^^I\^^M % same for <tab>
```

\lq
\rq

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

\lbrack
\rbrack

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

\aa
\AA

These are not from plain.tex but they are similar to other commands found here and nowhere else, being alternate input forms for characters.

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

\endgraf
\endline

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

\space

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

\empty

This probably ought to go altogether, but let it to the LaTeX version to save space.

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

\null

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

\bgroup
\egroup

```
\let\bgroup={
\let\egroup=}
```

\obeylines
\obeyspaces

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

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

\loop
\iterate
\repeat

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

```
\long\def \loop #1\repeat{%
  \def\iterate{#1\relax  % Extra \relax
               \expandafter\iterate\fi
               }%
  \iterate
  \let\iterate\relax
}
```

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

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

LaTeX defines `\smallskip`, etc. in `ltspace.dtx`.

\nointerlineskip
\offinterlineskip

```
\def\nointerlineskip{\prevdepth-\@m\p@}
\def\offinterlineskip{\baselineskip-\@m\p@
  \lineskip\z@ \lineskiplimit\maxdimen}
```

\vglue
\hglue

```
\def\vglue{\afterassignment\vgl@\skip@=}
\def\vgl@{\par \dimen@\prevdepth \hrule \@height\z@
  \nobreak\vskip\skip@ \prevdepth\dimen@}
\def\hglue{\afterassignment\hgl@\skip@=}
\def\hgl@{\leavevmode \count@\spacefactor \vrule \@width\z@
  \nobreak\hskip\skip@ \spacefactor\count@}
```

LaTeX defines `~` in `ltdefns.dtx`.

\slash This generates a `/` acting a bit like `-` but still allows hyphenation in the word part preceding it (but not after).

```
\def\slash{/\penalty\exhyphenpenalty}
```

\break
\nobreak
\allowbreak

```
\def\break{\penalty-\@M}
\def\nobreak{\penalty \@M}
\def\allowbreak{\penalty \z@}
```

\filbreak
\goodbreak

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

\eject Define `\eject` as in plain TeX but define `\supereject` only in the compatibility file.

```
\def\eject{\par\break}
```

\removelastskip

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

\smallbreak
\medbreak
\bigbreak

```
\def\smallbreak{\par\ifdim\lastskip<\smallskipamount
  \removelastskip\penalty-50\smallskip\fi}
\def\medbreak{\par\ifdim\lastskip<\medskipamount
  \removelastskip\penalty-100\medskip\fi}
\def\bigbreak{\par\ifdim\lastskip<\bigskipamount
  \removelastskip\penalty-200\bigskip\fi}
```

\m@th

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

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

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

\strutbox
\strut
LaTeX sets `\strutbox` in `\set@fontsize`.

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

\hidewidth For alignment entries that can stick out.

```
\def\hidewidth{\hskip\hideskip}
```

`\narrower`

```
\def\narrower{%
  \advance\leftskip\parindent
  \advance\rightskip\parindent}
```

LaTeX defines `\ae` and similar commands elsewhere.

```
\chardef\%=`\%
\chardef\&=`\&
\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

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

`\mathhexbox`

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

`\ialign`

```
\def\ialign{\everycr{}\tabskip\z@skip\halign} % initialized \halign
```

`\oalign`
`\o@lign`
`\ooalign`

```
\def\oalign#1{\leavevmode\vtop{\baselineskip\z@skip \lineskip.25ex%
  \ialign{##\crcr#1\crcr}}}
\def\o@lign{\lineskiplimit\z@ \oalign}
\def\ooalign{\lineskiplimit-\maxdimen \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 LaTeX.

Since the version given here has been in use by LaTeX 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.

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

`\ltx@sh@ft` This is the LaTeX 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).

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

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

LaTeX change: Make `\t` work in a moving argument. Now defined elsewhere.

\hrulefill
\dotfill

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

```
\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.

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

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

```
\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 LaTeX.

```
\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:

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

\showoutput
\loggingoutput

```
\gdef\loggingoutput{\tracingoutput\@ne
    \showboxbreadth\maxdimen\showboxdepth\maxdimen\errorstopmode}
\gdef\showoutput{\loggingoutput\showoverfull}
⟨/2ekernel⟩
```

\tracingall
\loggingall

```
⟨latexrelease⟩\IncludeInRelease{2015/01/01}{\loggingall}{etex tracing}%
⟨*2ekernel | latexrelease⟩
\ifx\tracingscantokens\@undefined
\gdef\loggingall{%
  \tracingstats\tw@
  \tracingpages\@ne
  \tracinglostchars\@ne
  \tracingparagraphs\@ne
  \errorcontextlines\maxdimen
  \loggingoutput
  \tracingmacros\tw@
  \tracingcommands\tw@
  \tracingrestores\@ne
  }%
\else
\gdef\loggingall{%
  \tracingstats\tw@
  \tracingpages\@ne
  \tracinglostchars\tw@
  \tracingparagraphs\@ne
  \tracinggroups\@ne
  \tracingifs\@ne
  \tracingscantokens\@ne
  \tracingnesting\@ne
  \errorcontextlines\maxdimen
  \loggingoutput
  \tracingmacros\tw@
  \tracingcommands\thr@@
  \tracingrestores\@ne
  \tracingassigns\@ne
}%
\fi
\gdef\tracingall{\showoverfull\loggingall}
⟨/2ekernel | latexrelease⟩
⟨latexrelease⟩\EndIncludeInRelease
⟨latexrelease⟩\IncludeInRelease{0000/00/00}{\loggingall}{etex tracing}%
⟨latexrelease⟩\gdef\loggingall{\tracingcommands\tw@\tracingstats\tw@
⟨latexrelease⟩  \tracingpages\@ne\tracinglostchars\@ne
⟨latexrelease⟩  \tracingmacros\tw@\tracingparagraphs\@ne\tracingrestores\@ne
⟨latexrelease⟩  \errorcontextlines\maxdimen\loggingoutput}
⟨latexrelease⟩  \gdef\tracingall{\loggingall\showoverfull}
⟨latexrelease⟩\EndIncludeInRelease
```

\tracingnone
\hideoutput

```
⟨latexrelease⟩\IncludeInRelease{2015/01/01}{\tracingnone}%
⟨latexrelease⟩                             {turn off etex tracing}%
⟨*2ekernel | latexrelease⟩
\ifx\tracingscantokens\@undefined
\def\tracingnone{%
  \tracingonline\z@
  \tracingcommands\z@
  \showboxdepth\m@ne
  \showboxbreadth\m@ne
  \tracingoutput\z@
  \errorcontextlines\m@ne
  \tracingrestores\z@
  \tracingparagraphs\z@
  \tracingmacros\z@
  \tracinglostchars\@ne
  \tracingpages\z@
  \tracingstats\z@
}%
\else
\def\tracingnone{%
  \tracingassigns\z@
  \tracingrestores\z@
  \tracingonline\z@
  \tracingcommands\z@
  \showboxdepth\m@ne
  \showboxbreadth\m@ne
  \tracingoutput\z@
  \errorcontextlines\m@ne
  \tracingnesting\z@
  \tracingscantokens\z@
  \tracingifs\z@
  \tracinggroups\z@
  \tracingparagraphs\z@
  \tracingmacros\z@
  \tracinglostchars\@ne
  \tracingpages\z@
  \tracingstats\z@
}%
\fi
```

```
\def\hideoutput{%
  \tracingoutput\z@
  \showboxbreadth\m@ne
  \showboxdepth\m@ne
  \tracingonline\m@ne
}%
```

```
⟨/2ekernel | latexrelease⟩
⟨latexrelease⟩\EndIncludeInRelease
⟨latexrelease⟩\IncludeInRelease{0000/00/00}{\tracingnone}%
⟨latexrelease⟩                                {turn off etex tracing}%
⟨latexrelease⟩\let\tracingnone\@undefined
⟨latexrelease⟩\let\hideoutput\@undefined
⟨latexrelease⟩\EndIncludeInRelease
```

LaTeX change: `\showhyphens` Defined later.

Punctuation affects the spacing.

```
⟨*2ekernel⟩
\nonfrenchspacing
⟨/2ekernel⟩
```

File c
ltvers.dtx

10 Version Identification

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

`\fmtname`
`\fmtversion`
`\latexreleaseversion`
`\patch@level`

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

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).

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.

```
⟨*2ekernel⟩
\def\fmtname{LaTeX2e}
\edef\fmtversion
⟨/2ekernel⟩
⟨latexrelease⟩\edef\latexreleaseversion
⟨*2ekernel | latexrelease⟩
   {2020-02-02}
⟨/2ekernel | latexrelease⟩
⟨*2ekernel⟩
\def\patch@level{5}
```

`\development@branch@name`

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.

```
\edef\development@branch@name{}
```

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:-)).

```
\iffalse
\def\reserved@a#1/#2/#3\@nil{%
  \count@\year
  \advance\count@-#1\relax
  \multiply\count@ by 12\relax
  \advance\count@\month
  \advance\count@-#2\relax}
\expandafter\reserved@a\fmtversion\@nil
```

`\count@` is now the age of this file in months. Take a generous definition of ‘year’ so this message is not generated too often.

```
\ifnum\count@>65
  \typeout{^^J%
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!^^J%
!  You are attempting to make a LaTeX format from a source file^^J%
!  That is more than five years old.^^J%
!^^J%
!  If you enter <return> to scroll past this message then the format^^J%
!  will be built, but please consider obtaining newer source files^^J%
!  before continuing to build LaTeX.^^J%
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!^^J%
}
    \errhelp{To avoid this error message, obtain new LaTeX sources.}
    \errmessage{LaTeX source files more than 5 years old!}
\fi
\let\reserved@a\relax
\fi
```

```
  \ifnum0\ifnum\patch@level=0 \ifx\development@branch@name\@empty 1\fi\fi>0 %
    \everyjob\expandafter{\the\everyjob
      \typeout{\fmtname \space<\fmtversion>}}
    \immediate
    \write16{\fmtname \space<\fmtversion>}
  \else\ifnum\patch@level>0
    \everyjob\expandafter{\the\everyjob
      \typeout{\fmtname \space<\fmtversion> patch level \patch@level}}
    \immediate
    \write16{\fmtname \space<\fmtversion> patch level \patch@level}
  \else
    \everyjob\expandafter{\the\everyjob
      \typeout{\fmtname \space<\fmtversion> pre-release-\number-\patch@level\space
        \ifx\development@branch@name\@undefined  \else
          \ifx\development@branch@name\@empty  \else
            \space (\development@branch@name\space branch)%
          \fi
        \fi
    }}
    \immediate
    \write16{\fmtname \space<\fmtversion> pre-release-\number-\patch@level\space
        \ifx\development@branch@name\@undefined  \else
          \ifx\development@branch@name\@empty  \else
            \space (\development@branch@name\space branch)%
          \fi
        \fi
    }
    \fi
  \fi
⟨/2ekernel⟩
```

`\IncludeInRelease`

```
⟨2ekernel⟩\let\@currname\@empty
```

```
⟨*2ekernel | latexrelease⟩
```

```
⟨latexrelease⟩\newif\if@includeinrelease
⟨latexrelease⟩\@includeinreleasefalse
```

```
\def\IncludeInRelease#1{%
 \if@includeinrelease
   \PackageError{latexrelease}{mis-matched IncludeInRelease}%
                {There is an \string\EndIncludeRelease\space missing}%
  \@includeinreleasefalse
  \fi
  \kernel@ifnextchar[%
  {\@IncludeInRelease{#1}}
  {\@IncludeInRelease{#1}[#1]}}
```

If a specific date has not been specified in `latexrelease` use '`#1`'.

```
\def\@IncludeInRelease#1[#2]{\@IncludeInRele@se{#2}}
```

```
\def\@IncludeInRele@se#1#2#3{%
  \toks@{[#1] #3}%
  \expandafter\ifx\csname\string#2+\@currname+IIR\endcsname\relax
    \ifnum\expandafter\@parse@version#1//00\@nil
          >\expandafter\@parse@version\fmtversion//00\@nil
      \GenericInfo{}{Skipping: \the\toks@}%
      \expandafter\expandafter\expandafter\@gobble@IncludeInRelease
    \else
      \GenericInfo{}{Applying: \the\toks@}%
      \@includeinreleasetrue
      \expandafter\let\csname\string#2+\@currname+IIR\endcsname\@empty
    \fi
  \else
    \GenericInfo{}{Already applied: \the\toks@}%
    \expandafter\@gobble@IncludeInRelease
  \fi
}
```

```
\def\EndIncludeInRelease{%
\if@includeinrelease
  \@includeinreleasefalse
\else
  \PackageError{latexrelease}{mis-matched EndIncludeInRelease}{}%
\fi}
```

```
\long\def\@gobble@IncludeInRelease#1\EndIncludeInRelease{%
  \@includeinreleasefalse
  \@check@IncludeInRelease#1\IncludeInRelease\@check@IncludeInRelease
  \@end@check@IncludeInRelease}
```

```
\long\def\@check@IncludeInRelease#1\IncludeInRelease
                                   #2#3\@end@check@IncludeInRelease{%
  \ifx\@check@IncludeInRelease#2\else
    \PackageError{latexrelease}{skipped IncludeInRelease for tag \string#2}{}%
  \fi}
```

```
⟨/2ekernel | latexrelease⟩
```

File d
ltdefns.dtx

11 Definitions

This section contains commands used in defining other macros.

```
⟨*2ekernel⟩
```

11.1 Initex initialisations

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

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

`\typeout` Display something on the terminal.

```
\def\typeout#1{\begingroup\set@display@protect
    \immediate\write\@unused{#1}\endgroup}
```

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

```
\newlinechar`\^^J
```

11.2 Saved versions of TEX primitives

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

`\@@par`

```
\let\@@par=\par
%\let\@@input=\input    %%% moved earlier
%\let\@@end=\end        %%%
```

`\@@hyph` Save original primitive definition.

```
\let\@@hyph=\-
```

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

```
\let\@@italiccorr=\/
```

`\@height` `\@depth` `\@width` `\@minus` `\@plus` The following definitions save token space. E.g., using `\@height` instead of height saves 5 tokens at the cost in time of one macro expansion.

```
\def\@height{height} \def\@depth{depth} \def\@width{width}
\def\@minus{minus}
\def\@plus{plus}
```

`\hb@xt@` The next one is another 100 tokens worth.

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

```
\message{hacks,}
```

11.3 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 *{⟨*FOO*⟩}[⟨*i*⟩]{⟨*TEXT*⟩}
User command to define \FOO to be a macro with i arguments (i = 0 if missing) having the definition ⟨*TEXT*⟩. Produces an error if \FOO 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 *{⟨*FOO*⟩}[⟨*i*⟩]{⟨*TEXT*⟩}
Same as \newcommand, except it checks if \FOO already defined.

\newenvironment *{⟨*FOO*⟩}[⟨*i*⟩]{⟨*DEF1*⟩}{⟨*DEF2*⟩}
equivalent to:
\newcommand{\FOO}[i]{DEF1} \def{\endFOO}{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 {⟨*message*⟩}
Produces a warning message on the terminal.

\typein {⟨*message*⟩}
Types message, asks the user to type in a command, then executes it

\typein [⟨*CS*⟩]{⟨*MSG*⟩}
Same as above, except defines \CS to be the input instead of executing it.

\typein

```
\def\typein{%
```

```
  \let\@typein\relax
  \@testopt\@xtypein\@typein}
\ifx\directlua\@undefined
\def\@xtypein[#1]#2{%
 \typeout{#2}%
 \advance\endlinechar\@M
 \read\@inputcheck to#1%
 \advance\endlinechar-\@M
 \@typein}%
\else
\def\@xtypein[#1]#2{%
  \typeout{#2}%
  \begingroup \endlinechar\m@ne
  \read\@inputcheck to#1%
  \expandafter\endgroup
  \expandafter\def\expandafter#1\expandafter{#1}%
  \@typein}%
\fi
```

\@namedef

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

\@nameuse

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

\@cons

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

\@car
\@cdr

```
\def\@car#1#2\@nil{#1}
\def\@cdr#1#2\@nil{#2}
```

\@carcube

`\@carcube T1 ... Tn\@nil` = `T1 T2 T3` , $n > 3$

```
\def\@carcube#1#2#3#4\@nil{#1#2#3}
```

\@onlypreamble
\@preamblecmds

This macro adds its argument to the list of commands stored in `\@preamblecmds` to be disabled after `\begin{document}`. These commands are redefined to generate `\@notprerr` at this point.

```
\def\@preamblecmds{}
\def\@onlypreamble#1{%
  \expandafter\gdef\expandafter\@preamblecmds\expandafter{%
        \@preamblecmds\do#1}}
\@onlypreamble\@onlypreamble
\@onlypreamble\@preamblecmds
```

\@star@or@long

Look ahead for a `*`. If present reset `\l@ngrel@x` so that the next definition, `#1`, will be non-long.

```
\def\@star@or@long#1{%
  \@ifstar
   {\let\l@ngrel@x\relax#1}%
   {\let\l@ngrel@x\long#1}}
```

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

```
\let\l@ngrel@x\relax
```

`\newcommand` User level `\newcommand`.

```
\def\newcommand{\@star@or@long\new@command}
```

`\new@command`

```
\def\new@command#1{%
  \@testopt{\@newcommand#1}0}
```

`\@newcommand` `\@argdef` `\@xargdef` Handling arguments for `\newcommand`.

```
\def\@newcommand#1[#2]{%
  \kernel@ifnextchar [{\@xargdef#1[#2]}%
                {\@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.

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

Handle the second optional argument.

```
\long\def\@xargdef#1[#2][#3]#4{%
  \@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`.

```
     \expandafter\def\expandafter#1\expandafter{%
          \expandafter
          \@protected@testopt
          \expandafter
          #1%
          \csname\string#1\endcsname
          {#3}}%
```

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

```
       \expandafter\@yargdef
          \csname\string#1\endcsname
           \tw@
           {#2}%
           {#4}}}
```

`\@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.

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

`\@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.

```
\def\@protected@testopt#1{%
  \ifx\protect\@typeset@protect
    \expandafter\@testopt
  \else
    \@x@protect#1%
  \fi}
```

`\@yargdef`
`\@yargd@f`

These generate a primitive argument specification, from a LaTeX `[`⟨*digit*⟩`]` form; in fact ⟨*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 LaTeX2.09 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`.

```
\long \def \@yargdef #1#2#3{%
  \ifx#2\tw@
    \def\reserved@b##11{[####1]}%
  \else
    \let\reserved@b\@gobble
  \fi
  \expandafter
    \@yargd@f \expandafter{\number #3}#1%
}
```

```
\long \def \@yargd@f#1#2{%
  \def \reserved@a ##1#1##2##{%
    \expandafter\def\expandafter#2\reserved@b ##1#1%
    }%
  \l@ngrel@x \reserved@a 0##1##2##3##4##5##6##7##8##9###1%
}
```

\@reargdef

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

\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 `\@temp`a–e.)

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

\renew@command

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

\@ifdefinable
\@@ifdefinable
\@rc@ifdefinable

Test is user is allowed to define a command.

```
\long\def\@ifdefinable #1#2{%
      \edef\reserved@a{\expandafter\@gobble\string #1}%
     \@ifundefined\reserved@a
         {\edef\reserved@b{\expandafter\@carcube \reserved@a xxx\@nil}%
          \ifx \reserved@b\@qend \@notdefinable\else
            \ifx \reserved@a\@qrelax \@notdefinable\else
              #2%
            \fi
          \fi}%
         \@notdefinable}
```

Saved definition of `\@ifdefinable`.

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

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

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

\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.

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

\new@environment

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

\@newenva

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

\@newenvb

```
\def\@newenvb#1[#2][#3]{\@newenv{#1}{[#2][{#3}]}}
```

\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.

```
\def\renewenvironment{\@star@or@long\renew@environment}
```

\renew@environment

```
\def\renew@environment#1{%
  \@ifundefined{#1}%
     {\@latex@error{Environment #1 undefined}\@ehc
     }\relax
  \expandafter\let\csname#1\endcsname\relax
  \expandafter\let\csname end#1\endcsname\relax
  \new@environment{#1}}
```

\@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 `\par`s)

Make sure that an attempt to define a 'graf' or 'group' environment fails.

```
\long\def\@newenv#1#2#3#4{%
  \@ifundefined{#1}%
    {\expandafter\let\csname#1\expandafter\endcsname
                         \csname end#1\endcsname}%
    \relax
  \expandafter\new@command
     \csname #1\endcsname#2{#3}%
     \l@ngrel@x\expandafter\def\csname end#1\endcsname{#4}}
```

\newif

And here's a different sort of allocation: For example, `\newif\iffoo` creates `\footrue`, `\foofalse` to go with `\iffoo`.

```
\def\newif#1{%
  \count@\escapechar \escapechar\m@ne
    \let#1\iffalse
    \@if#1\iftrue
    \@if#1\iffalse
  \escapechar\count@}
```

\@if

```
\def\@if#1#2{%
  \expandafter\def\csname\expandafter\@gobbletwo\string#1%
                    \expandafter\@gobbletwo\string#2\endcsname
                       {\let#1#2}}
```

\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.

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

\provide@command

```
\def\provide@command#1{%
  \begingroup
    \escapechar\m@ne\xdef\@gtempa{{\string#1}}%
  \endgroup
  \expandafter\@ifundefined\@gtempa
    {\def\reserved@a{\new@command#1}}%
    {\def\reserved@a{\renew@command\reserved@a}}%
   \reserved@a}%
```

\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 befor 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.

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

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

```
\@onlypreamble\CheckCommand
```

\check@command

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

```
\long\def\@check@c#1#2#3{%
  \expandafter\let\csname\string\reserved@a\endcsname\relax
  \renew@command\reserved@a#2{#3}%
  \@ifundefined{\string\reserved@a}%
   {\@check@eq#1\reserved@a}%
   {\expandafter\@check@eq
          \csname\string#1\expandafter\endcsname
          \csname\string\reserved@a\endcsname}}
\@onlypreamble\@check@c
```

\@check@eq

Complain if #1 and #2 are not \ifx equal.

```
\def\@check@eq#1#2{%
  \ifx#1#2\else
     \@latex@warning@no@line
                {Command \noexpand#1 has
                 changed.\MessageBreak
                 Check if current package is valid}%
  \fi}
\@onlypreamble\@check@eq
```

`\@gobble` `\@gobbletwo` `\@gobblethree` `\@gobblefour`

The `\@gobble` macro is used to get rid of its argument.

```
\long\def \@gobble #1{}
\long\def \@gobbletwo #1#2{}
\long\def \@gobblethree #1#2#3{}
\long\def \@gobblefour #1#2#3#4{}
```

`\@firstofone` `\@firstoftwo` `\@secondoftwo`

Some argument-grabbers.

```
\long\def\@firstofone#1{#1}
\long\def\@firstoftwo#1#2{#1}
\long\def\@secondoftwo#1#2{#2}
```

`\@iden`

`\@iden` is another name for `\@firstofone` for compatibility reasons.

```
\let\@iden\@firstofone
```

`\@thirdofthree`

Another grabber now used in the encoding specific section.

```
\long\def\@thirdofthree#1#2#3{#3}
```

`\@expandtwoargs`

A macro to totally expand two arguments to another macro

```
\def\@expandtwoargs#1#2#3{%
\edef\reserved@a{\noexpand#1{#2}{#3}}\reserved@a}
```

`\@backslashchar`

A category code 12 backslash.

```
\edef\@backslashchar{\expandafter\@gobble\string\\}
```

11.4 Robust commands and protect

Fragile and robust commands are one of the thornier issues in LaTeX's commands. Whilst typesetting documents, LaTeX makes use of many of TeX'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 LaTeX, 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 LaTeX uses for making fragile commands robust is to precede them with `\protect`. This can have one of five 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 `\edef`s, `\mark`s 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 `\proetected@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`

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

`\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\_`.

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 `\ifmmode`s inside `\halign`s. 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`.

```
\def\DeclareRobustCommand{\@star@or@long\declare@robustcommand}
```

```
\def\declare@robustcommand#1{%
   \ifx#1\@undefined\else\ifx#1\relax\else
      \@latex@info{Redefining \string#1}%
   \fi\fi
   \edef\reserved@a{\string#1}%
   \def\reserved@b{#1}%
   \edef\reserved@b{\expandafter\strip@prefix\meaning\reserved@b}%
   \edef#1{%
```

```
      \ifx\reserved@a\reserved@b
         \noexpand\x@protect
         \noexpand#1%
      \fi
      \noexpand\protect
      \expandafter\noexpand\csname
         \expandafter\@gobble\string#1 \endcsname
   }%
   \let\@ifdefinable\@rc@ifdefinable
   \expandafter\new@command\csname
      \expandafter\@gobble\string#1 \endcsname
}
```

\@x@protect
\x@protect

```
\def\x@protect#1{%
   \ifx\protect\@typeset@protect\else
      \@x@protect#1%
   \fi
}
\def\@x@protect#1\fi#2#3{%
   \fi\protect#1%
}
```

\@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.

```
\let\@typeset@protect\relax
```

\set@display@protect
\set@typeset@protect

These macros set `\protect` appropriately for typesetting or displaying.

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

\protected@edef
\protected@xdef
\unrestored@protected@xdef
\restore@protect

The commands `\protected@edef` and `\protected@xdef` perform 'safe' `\edef`s and `\xdef`s, saving and restoring `\protect` appropriately. For cases where restoring `\protect` doesn't matter, there's an 'unsafe' `\unrestored@protected@xdef`, useful if you know what you're doing!

```
\def\protected@edef{%
   \let\@@protect\protect
   \let\protect\@unexpandable@protect
   \afterassignment\restore@protect
   \edef
}
\def\protected@xdef{%
   \let\@@protect\protect
   \let\protect\@unexpandable@protect
   \afterassignment\restore@protect
   \xdef
}
\def\unrestored@protected@xdef{%
   \let\protect\@unexpandable@protect
   \xdef
}
\def\restore@protect{\let\protect\@@protect}
```

`\protect` The normal meaning of `\protect`

```
\set@typeset@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 dones manually (like with `\begin`).

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

```
⟨/2ekernel⟩
⟨latexrelease⟩\IncludeInRelease{2019/10/01}{\MakeRobust}{\MakeRobust}%
⟨*2ekernel | latexrelease⟩
\def\MakeRobust#1{%
  \@ifundefined{\expandafter\@gobble\string#1}{%
    \@latex@error{The control sequence `\string#1' is undefined!%
      \MessageBreak There is nothing here to make robust}%
    \@eha
  }%
```

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`.

```
  {%
    \@ifundefined{\expandafter\@gobble\string#1\space}%
    {%
      \global\expandafter\let\csname
      \expandafter\@gobble\string#1\space\endcsname=#1%
      \edef\reserved@a{\string#1}%
      \def\reserved@b{#1}%
      \edef\reserved@b{\expandafter\strip@prefix\meaning\reserved@b}%
      \xdef#1{%
        \ifx\reserved@a\reserved@b
          \noexpand\x@protect\noexpand#1%
        \fi
        \noexpand\protect\expandafter\noexpand
        \csname\expandafter\@gobble\string#1\space\endcsname}%
    }%
    {\@latex@info{The control sequence `\string#1' is already robust}}%
  }%
}%
⟨/2ekernel | latexrelease⟩
⟨latexrelease⟩\EndIncludeInRelease
⟨latexrelease⟩\IncludeInRelease{2015/01/01}{\MakeRobust}{\MakeRobust}%
⟨latexrelease⟩\def\MakeRobust#1{%
⟨latexrelease⟩  \@ifundefined{\expandafter\@gobble\string#1}{%
⟨latexrelease⟩    \@latex@error{The control sequence `\string#1' is undefined!%
⟨latexrelease⟩      \MessageBreak There is nothing here to make robust}%
⟨latexrelease⟩    \@eha
⟨latexrelease⟩  }%
```

```
⟨latexrelease⟩  {%
⟨latexrelease⟩    \@ifundefined{\expandafter\@gobble\string#1\space}%
⟨latexrelease⟩    {%
⟨latexrelease⟩      \expandafter\let\csname
⟨latexrelease⟩      \expandafter\@gobble\string#1\space\endcsname=#1%
⟨latexrelease⟩      \edef\reserved@a{\string#1}%
⟨latexrelease⟩      \def\reserved@b{#1}%
⟨latexrelease⟩      \edef\reserved@b{\expandafter\strip@prefix\meaning\reserved@b}%
⟨latexrelease⟩      \edef#1{%
⟨latexrelease⟩        \ifx\reserved@a\reserved@b
⟨latexrelease⟩          \noexpand\x@protect\noexpand#1%
⟨latexrelease⟩        \fi
⟨latexrelease⟩        \noexpand\protect\expandafter\noexpand
⟨latexrelease⟩        \csname\expandafter\@gobble\string#1\space\endcsname}%
⟨latexrelease⟩    }%
⟨latexrelease⟩    {\@latex@info{The control sequence `\string#1' is already robust}}%
⟨latexrelease⟩  }%
⟨latexrelease⟩}%
⟨latexrelease⟩\EndIncludeInRelease
⟨latexrelease⟩\IncludeInRelease{0000/00/00}{\MakeRobust}{\MakeRobust}%
⟨latexrelease⟩\let\MakeRobust\@undefined
⟨latexrelease⟩\EndIncludeInRelease
⟨*2ekernel⟩
```

\kernel@make@fragile The opposite of \MakeRobust execpt 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 earler versions of LaTeX 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.

```
⟨/2ekernel⟩
⟨*2ekernel | latexrelease⟩
⟨latexrelease⟩\IncludeInRelease{0000/00/00}%
⟨latexrelease⟩                 {\kernel@make@fragile}{Undo robustness}%
\def\kernel@make@fragile#1{%
  \@ifundefined{\expandafter\@gobble\string#1\space}%
```

If not robust do nothing.

```
      {}%
```

Otherwise copy \foo␣ back to \foo and then undefine the payload command.

```
      {%
       \global\expandafter\let\expandafter #1\csname
       \expandafter\@gobble\string#1\space\endcsname
       \global\expandafter\let\csname
       \expandafter\@gobble\string#1\space\endcsname\@undefined
      }%
}
⟨latexrelease⟩\EndIncludeInRelease
⟨/2ekernel | latexrelease⟩
⟨*2ekernel⟩
```

11.5 Internal defining commands

These commands are used internally to define other LaTeX commands.

`\@ifundefined` Check if first arg is undefined or `\relax` and execute second or third arg depending,

```
⟨/2ekernel⟩
⟨latexrelease⟩\IncludeInRelease{2018-04-01}{\@ifundefined}
⟨latexrelease⟩{Leave commands undefined in \@ifundefined}%
⟨*2ekernel | latexrelease⟩
```

Version using `\ifcsname` to avoid defining undefined tokens to `\relax`. Defined here to simplify using unmatched `\fi`.

```
\def\@ifundefined#1{%
  \ifcsname#1\endcsname\@ifundefin@d@i\else\@ifundefin@d@ii\fi{#1}}
```

```
\long\def\@ifundefin@d@i#1\fi#2{\fi
  \expandafter\ifx\csname #2\endcsname\relax
    \@ifundefin@d@ii
  \fi
  \@secondoftwo}
```

```
\long\def\@ifundefin@d@ii\fi#1#2#3{\fi #2}
```

Now test of engine.

```
\ifx\numexpr\@undefined
```

Classic version (should not be needed as etex is assumed).

```
\def\@ifundefined#1{%
  \expandafter\ifx\csname#1\endcsname\relax
    \expandafter\@firstoftwo
  \else
    \expandafter\@secondoftwo
  \fi}
\else\ifx\directlua\@undefined
```

Use the `\ifcsname` defined above.

```
\else
```

Optimised version for LuaTEX, using `\lastnamedcs`

```
\def\@ifundefined#1{%
  \ifcsname#1\endcsname
    \expandafter\ifx\lastnamedcs\relax\else\@ifundefin@d@i\fi
  \fi
  \@firstoftwo}
```

```
\long\def\@ifundefin@d@i#1#2#3#4#5{#1#2#5}
```

```
\fi
\fi
⟨/2ekernel | latexrelease⟩
⟨latexrelease⟩\EndIncludeInRelease
⟨latexrelease⟩\IncludeInRelease{0000-00-00}{\@ifundefined}
⟨latexrelease⟩{Leave commands undefined in \@ifundefined}%
⟨latexrelease⟩\def\@ifundefined#1{%
⟨latexrelease⟩  \expandafter\ifx\csname#1\endcsname\relax
⟨latexrelease⟩    \expandafter\@firstoftwo
⟨latexrelease⟩  \else
⟨latexrelease⟩    \expandafter\@secondoftwo
⟨latexrelease⟩  \fi}
⟨latexrelease⟩\EndIncludeInRelease
⟨*2ekernel⟩
```

\@qend
\@qrelax

The following define \@qend and \@qrelax to be the strings ‘end’ and ‘relax’ with the characters \catcoded 12.

```
\edef\@qend{\expandafter\@cdr\string\end\@nil}
\edef\@qrelax{\expandafter\@cdr\string\relax\@nil}
```

\@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.

```
\long\def\@ifnextchar#1#2#3{%
  \let\reserved@d=#1%
  \def\reserved@a{#2}%
  \def\reserved@b{#3}%
  \futurelet\@let@token\@ifnch}
```

\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.

```
\let\kernel@ifnextchar\@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.

```
\def\@ifnch{%
  \ifx\@let@token\@sptoken
    \let\reserved@c\@xifnch
  \else
    \ifx\@let@token\reserved@d
      \let\reserved@c\reserved@a
    \else
      \let\reserved@c\reserved@b
    \fi
  \fi
  \reserved@c}
```

\@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.

```
\def\:{\let\@sptoken= } \:  % this makes \@sptoken a space token
```

\@xifnch

In the following definition of \@xifnch, \: is again used to get a space token as delimiter into the definition.

```
\def\:{\@xifnch} \expandafter\def\: {\futurelet\@let@token\@ifnch}
```

\@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.

```
\def\@ifstar#1{\@ifnextchar *{\@firstoftwo{#1}}}
```

`\@dblarg`
`\@xdblarg`

```
\long\def\@dblarg#1{\kernel@ifnextchar[{#1}{\@xdblarg{#1}}}
\long\def\@xdblarg#1#2{#1[{#2}]{#2}}
```

`\@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.

```
\def\@sanitize{\@makeother\ \@makeother\\\@makeother\$\@makeother\&%
\@makeother\#\@makeother\^\@makeother\_\@makeother\%\@makeother\~}
```

`\@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.

```
\def \@onelevel@sanitize #1{%
  \edef #1{\expandafter\strip@prefix
            \meaning #1}%
}
```

`\makeatletter`
`\makeatother` Make internal control sequences accessible or inaccessible.

```
\DeclareRobustCommand\makeatletter{\catcode`\@11\relax}
\DeclareRobustCommand\makeatother{\catcode`\@12\relax}
```

12 Discretionary Hyphenation

`\-`
`\@dischyph` 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 hyhenation.

LaTeX, from LaTeX2.09 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 TeX definition is saved as `\@@hyph` just in case anyone needs it.

LaTeX 2$_\varepsilon$, 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 LaTeX definition accessible via latexrelease as usual.

```
⟨/2ekernel⟩
⟨latexrelease⟩\IncludeInRelease{2017/04/15}{\-}{Use \hyphenchar in \-}%
```

Temporary definition of `\@latex@info`, final definition is later.

```
⟨*2ekernel⟩
\def\@latex@info#1{}
⟨/2ekernel⟩
```

```
⟨*2ekernel | latexrelease⟩
\DeclareRobustCommand{\-}{%
  \discretionary{%
    \char \ifnum\hyphenchar\font<\z@
            \defaulthyphenchar
          \else
            \hyphenchar\font
          \fi
               }{}{}%
}
\let\@dischyph=\-
⟨/2ekernel | latexrelease⟩
⟨latexrelease⟩\EndIncludeInRelease
⟨latexrelease⟩\IncludeInRelease{0000/00/00}{\-}{Use \hyphenchar in \-}%
⟨latexrelease⟩\def\-{\discretionary{-}{}{}}
⟨latexrelease⟩\let\@dischyph=\-
⟨latexrelease⟩\EndIncludeInRelease
⟨*2ekernel⟩
```

Delayed from `ltvers.dtx`

```
\newif\if@includeinrelease
\@includeinreleasefalse
```

Delayed from `ltplain.dtx`

```
⟨/2ekernel⟩
⟨*2ekernel | latexrelease⟩
⟨latexrelease⟩\IncludeInRelease{2019/10/01}%
⟨latexrelease⟩                  {\allowbreak}{Make various commands robust}%
\MakeRobust\allowbreak
\MakeRobust\bigbreak
\MakeRobust\break
\MakeRobust\dotfill
\MakeRobust\frenchspacing
\MakeRobust\goodbreak
\MakeRobust\hrulefill
\MakeRobust\medbreak
\MakeRobust\nobreak
\MakeRobust\nonfrenchspacing
\MakeRobust\obeylines
\MakeRobust\obeyspaces
\MakeRobust\slash
\MakeRobust\smallbreak
\MakeRobust\strut
\MakeRobust\underbar
⟨/2ekernel | latexrelease⟩
⟨latexrelease⟩\EndIncludeInRelease
⟨latexrelease⟩\IncludeInRelease{0000/00/00}%
⟨latexrelease⟩                  {\allowbreak}{Make various commands robust}%
⟨latexrelease⟩
⟨latexrelease⟩\kernel@make@fragile\allowbreak
```

```
⟨latexrelease⟩\kernel@make@fragile\bigbreak
⟨latexrelease⟩\kernel@make@fragile\break
⟨latexrelease⟩\kernel@make@fragile\dotfill
⟨latexrelease⟩\kernel@make@fragile\frenchspacing
⟨latexrelease⟩\kernel@make@fragile\goodbreak
⟨latexrelease⟩\kernel@make@fragile\hrulefill
⟨latexrelease⟩\kernel@make@fragile\medbreak
⟨latexrelease⟩\kernel@make@fragile\nobreak
⟨latexrelease⟩\kernel@make@fragile\nonfrenchspacing
⟨latexrelease⟩\kernel@make@fragile\obeylines
⟨latexrelease⟩\kernel@make@fragile\obeyspaces
⟨latexrelease⟩\kernel@make@fragile\slash
⟨latexrelease⟩\kernel@make@fragile\smallbreak
⟨latexrelease⟩\kernel@make@fragile\strut
⟨latexrelease⟩\kernel@make@fragile\underbar
⟨latexrelease⟩
⟨latexrelease⟩\EndIncludeInRelease
⟨*2ekernel⟩

⟨/2ekernel⟩
```

File e
ltalloc.dtx

13 Counters

This section deals with counter and other variable allocation.

```
⟨*2ekernel⟩
```

The following are from plain TeX:

`\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.

`\sixt@@n` The number 16.

`\@m` The number 1000.

`\@MM` The number 20000.

`\@xxxii` The constant 32.

```
\chardef\@xxxii=32
```

`\@Mi` `\@Mii` `\@Miii` `\@Miv` Constants 10001–10004.

```
\mathchardef\@Mi=10001
\mathchardef\@Mii=10002
\mathchardef\@Miii=10003
\mathchardef\@Miv=10004
```

`\@tempcnta` `\@tempcntb` Scratch count registers used by LaTeX kernel commands.

```
\newcount\@tempcnta
\newcount\@tempcntb
```

`\if@tempswa` General boolean switch used by LaTeX kernel commands.

```
\newif\if@tempswa
```

`\@tempdima` `\@tempdimb` `\@tempdimc` Scratch dimen registers used by LaTeX kernel commands.

```
\newdimen\@tempdima
\newdimen\@tempdimb
\newdimen\@tempdimc
```

`\@tempboxa` Scratch box register used by LaTeX kernel commands.

```
\newbox\@tempboxa
```

`\@tempskipa` `\@tempskipb` Scratch skip registers used by LaTeX kernel commands.

```
\newskip\@tempskipa
\newskip\@tempskipb
```

`\@temptokena` Scratch token register used by LaTeX kernel commands.

```
\newtoks\@temptokena
```

`\@flushglue` Glue used for `\right-` & `\leftskip` = 0pt plus 1fil

```
\newskip\@flushglue \@flushglue = 0pt plus 1fil
```

```
⟨/2ekernel⟩
```

File f

ltcntrl.dtx

14 Program control structure

This section defines a number of control structure macros, such as while-loops and for-loops.
Historical LaTeX 2.09 comments (not necessarily accurate any more):

```
⟨*2ekernel⟩
\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 LaTeX 2.09 comments.

\@whilenoop
\@whilenum
\@iwhilenum

```
\long\def\@whilenum#1\do #2{\ifnum #1\relax #2\relax\@iwhilenum{#1\relax
     #2\relax}\fi}
\long\def\@iwhilenum#1{\ifnum #1\expandafter\@iwhilenum
         \else\expandafter\@gobble\fi{#1}}
```

\@whiledim
\@iwhiledim

```
\long\def\@whiledim#1\do #2{\ifdim #1\relax#2\@iwhiledim{#1\relax#2}\fi}
\long\def\@iwhiledim#1{\ifdim #1\expandafter\@iwhiledim
        \else\expandafter\@gobble\fi{#1}}
```

\@whileswnoop
\@whilesw
\@iwhilesw

```
\long\def\@whilesw#1\fi#2{#1#2\@iwhilesw{#1#2}\fi\fi}
\long\def\@iwhilesw#1\fi{#1\expandafter\@iwhilesw
         \else\@gobbletwo\fi{#1}\fi}
```

Historical LaTeX 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 LaTeX 2.09 comments.

\@nnil

```
\def\@nnil{\@nil}
```

\@empty

```
\def\@empty{}
```

\@fornoop

```
\long\def\@fornoop#1\@@#2#3{}
```

\@for

```
\long\def\@for#1:=#2\do#3{%
  \expandafter\def\expandafter\@fortmp\expandafter{#2}%
  \ifx\@fortmp\@empty \else
    \expandafter\@forloop#2,\@nil,\@nil\@@#1{#3}\fi}
```

\@forloop

```
\long\def\@forloop#1,#2,#3\@@#4#5{\def#4{#1}\ifx #4\@nnil \else
      #5\def#4{#2}\ifx #4\@nnil \else#5\@iforloop #3\@@#4{#5}\fi\fi}
```

`\@iforloop`

```
\long\def\@iforloop#1,#2\@@#3#4{\def#3{#1}\ifx #3\@nnil
       \expandafter\@fornoop \else
      #4\relax\expandafter\@iforloop\fi#2\@@#3{#4}}
```

`\@tfor`

```
\def\@tfor#1:={\@tf@r#1 }
\long\def\@tf@r#1#2\do#3{\def\@fortmp{#2}\ifx\@fortmp\space\else
    \@tforloop#2\@nil\@nil\@@#1{#3}\fi}
\long\def\@tforloop#1#2\@@#3#4{\def#3{#1}\ifx #3\@nnil
       \expandafter\@fornoop \else
      #4\relax\expandafter\@tforloop\fi#2\@@#3{#4}}
```

`\@break@tfor` Break out of a `\@tfor` loop. This should be called *inside* the scope of an `\if`. See `\@iffileonpath` for an example.

```
\long\def\@break@tfor#1\@@#2#3{\fi\fi}
```

`\@removeelement` Removes an element from a comma-separated list and puts it into a control sequence, called as `\@removeelement{`⟨*element*⟩`}{`⟨*list*⟩`}{`⟨*cs*⟩`}`. Due to the implementation method the ⟨*element*⟩ is not allowed to contain braces.

```
\def\@removeelement#1#2#3{%
  \def\reserved@a##1,#1,##2\reserved@a{##1,##2\reserved@b}%
  \def\reserved@b##1,\reserved@b##2\reserved@b{%
    \ifx,##1\@empty\else##1\fi}%
  \edef#3{%
    \expandafter\reserved@b\reserved@a,#2,\reserved@b,#1,\reserved@a}}
```

```
⟨/2ekernel⟩
```

File g
lterror.dtx

15 Error handling and tracing

This section defines LaTeX's error commands.

```
⟨*2ekernel⟩
```

The '2ekernel' 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.

```
\expandafter\let\csname ver@autoerr.sty\endcsname\fmtversion
```

15.1 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`.

```
\let\MessageBreak\relax
```

`\GenericInfo`

This takes two arguments: a continuation and a message, and sends the result to the log file.

```
\DeclareRobustCommand{\GenericInfo}[2]{%
   \begingroup
      \def\MessageBreak{^^J#1}%
      \set@display@protect
      \immediate\write\m@ne{#2\on@line.}%
   \endgroup
}
```

`\GenericWarning`

This takes two arguments: a continuation and a message, and sends the result to the screen.

```
\DeclareRobustCommand{\GenericWarning}[2]{%
   \begingroup
      \def\MessageBreak{^^J#1}%
      \set@display@protect
      \immediate\write\@unused{^^J#2\on@line.^^J}%
   \endgroup
}
```

`\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.

```
\bgroup
\lccode`\@=`\ %
```

```
\lccode`\~=`\ %
\lccode`\}=`\ %
\lccode`\{=`\ %
\lccode`\T=`\T%
\lccode`\H=`\H%
\catcode`\ =11\relax%
\lowercase{%
\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`.

```
\dimen@\ifx\@TeXversion\@undefined4\else\@TeXversion\fi\p@%
\ifdim\dimen@>3.14\p@%
```

First the 'standard case'.

```
\DeclareRobustCommand{\GenericError}[4]{%
\begingroup%
\immediate\write\@unused{}%
\def\MessageBreak{^^J}%
\set@display@protect%
\edef%
%    %<-------------------do not delete this space!------------------->%
\@err@                                                                 %
{{#4}}%
\errhelp
%    %<-------------------do not delete this space!------------------->%
\@err@                                                                 %
\let
%    %<-------------------do not delete this space!------------------->%
\@err@                                                                 %
\@empty
\def\MessageBreak{^^J#1}%
\def~{\errmessage{%
#2.^^J^^J%
#3^^J%
Type  H <return>  for immediate help%
%    %<-------------------do not delete this space!------------------->%
\@err@                                                                 %
}}%
~%
\endgroup}%
```

```
\else%
```

Secondly the version for old TEX's.

```
\DeclareRobustCommand{\GenericError}[4]{%
\begingroup%
\immediate\write\@unused{}%
\def\MessageBreak{^^J}%
\set@display@protect%
\edef%
%    %<-------------------do not delete this space!------------------->%
\@err@                                                                 %
{{#4}}%
\errhelp
%    %<-------------------do not delete this space!------------------->%
\@err@                                                                 %
\let
%    %<-------------------do not delete this space!------------------->%
\@err@                                                                 %
\errmessage
\def\MessageBreak{^^J#1}%
\def~{\typeout{! %
#2.^^J^^J%
#3^^J%
Type  H <return>  for immediate help.}%
%    %<-------------------do not delete this space!------------------->%
\@err@                                                                 %
{}}%
~%
\endgroup}%
\fi}%
```

\PackageError
\PackageWarning
\PackageWarningNoLine
\PackageInfo
\ClassError
\ClassWarning
\ClassWarningNoLine
\ClassInfo

These commands are intended for use by package and class writers, to give information to authors. The syntax is:

\PackageError{⟨*package*⟩}{⟨*error*⟩}{⟨*help*⟩}
\PackageWarning{⟨*package*⟩}{⟨*warning*⟩}
\PackageWarningNoLine{⟨*package*⟩}{⟨*warning*⟩}
\PackageInfo{⟨*package*⟩}{⟨*info*⟩}

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}{FOO}
\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 FOO on input line 54.
```

```
\gdef\PackageError#1#2#3{%
   \GenericError{%
      (#1)\@spaces\@spaces\@spaces\@spaces
   }{%
      Package #1 Error: #2%
   }{%
      See the #1 package documentation for explanation.%
   }{#3}%
}

\def\PackageWarning#1#2{%
   \GenericWarning{%
      (#1)\@spaces\@spaces\@spaces\@spaces
   }{%
      Package #1 Warning: #2%
   }%
}
\def\PackageWarningNoLine#1#2{%
   \PackageWarning{#1}{#2\@gobble}%
}
\def\PackageInfo#1#2{%
   \GenericInfo{%
      (#1) \@spaces\@spaces\@spaces
   }{%
      Package #1 Info: #2%
   }%
}

\gdef\ClassError#1#2#3{%
   \GenericError{%
      (#1) \space\@spaces\@spaces\@spaces
   }{%
      Class #1 Error: #2%
   }{%
      See the #1 class documentation for explanation.%
   }{#3}%
}

\def\ClassWarning#1#2{%
   \GenericWarning{%
      (#1) \space\@spaces\@spaces\@spaces
   }{%
      Class #1 Warning: #2%
   }%
}
\def\ClassWarningNoLine#1#2{%
   \ClassWarning{#1}{#2\@gobble}%
}
\def\ClassInfo#1#2{%
   \GenericInfo{%
      (#1) \space\space\@spaces\@spaces
   }{%
      Class #1 Info: #2%
   }%
}
```

\@latex@error
\@latex@warning
\@latex@warning@no@line
\@latex@info
\@latex@info@no@line

Errors and other info, for use in the LaTeX core.

```
\gdef\@latex@error#1#2{%
   \GenericError{%
      \space\space\space\@spaces\@spaces\@spaces
   }{%
      LaTeX Error: #1%
   }{%
      See the LaTeX manual or LaTeX Companion for explanation.%
   }{#2}%
}
```

```
\def\@latex@warning#1{%
   \GenericWarning{%
      \space\space\space\@spaces\@spaces\@spaces
   }{%
      LaTeX Warning: #1%
   }%
}
```

```
\def\@latex@warning@no@line#1{%
   \@latex@warning{#1\@gobble}}
```

```
\def\@latex@info#1{%
   \GenericInfo{%
      \@spaces\@spaces\@spaces
   }{%
      LaTeX Info: #1%
   }%
}
```

```
\def\@latex@info@no@line#1{%
  \@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%
   }%
 }
```

\c@errorcontextlines

`\errorcontextlines` as a LaTeX counter, so that it may be manipulated with `\setcounter` (once it is defined :-)

```
\let\c@errorcontextlines\errorcontextlines
\c@errorcontextlines=-1
```

\on@line

The message ‘ on input line *n*’.

```
\def\on@line{ on input line \the\inputlineno}
```

`\@warning` `\@@warning` `\@latexerr` Older LaTeX messages. For the moment, these `\let` to the new message commands. They may be changed later, once only obsolete packages and classes contain them.

```
\let\@warning\@latex@warning
\let\@@warning\@latex@warning@no@line
\global\let\@latexerr\@latex@error
```

`\@spaces` Four spaces.

```
\def\@spaces{\space\space\space\space}
```

15.2 Specific errors

`\@eha` `\@ehb` `\@ehc` `\@ehd` The more common error help messages.

```
\gdef\@eha{%
  Your command was ignored.\MessageBreak
  Type \space I <command> <return> \space to replace it %
  with another command,\MessageBreak
  or \space <return> \space to continue without it.}
\gdef\@ehb{%
  You've lost some text. \space \@ehc}
\gdef\@ehc{%
  Try typing \space <return> %
  \space to proceed.\MessageBreak
  If that doesn't work, type \space X <return> \space to quit.}
\gdef\@ehd{%
  You're in trouble here.  \space\@ehc}
```

`\@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`. . . .

```
\gdef\@notdefinable{%
 \@latex@error{%
   Command \@backslashchar\reserved@a\space
   already defined.\MessageBreak
   Or name \@backslashchar\@qend... illegal,
   see p.192 of the manual}\@eha}
```

`\@nolnerr` Generated by `\newline` and `\\` when called in vertical mode.

```
\gdef\@nolnerr{%
  \@latex@error{There's no line here to end}\@eha}
```

`\@nocounterr` Generated by `\setcounter`, `\addtocounter` or `\newcounter` if applied to an undefined counter ⟨*cnt*⟩.

`\@nocnterr` Obsolete error message generated in LaTeX2.09 by `\setcounter`, `\addtocounter` or `\newcounter` for undefined counter. DO NOT use for LaTeX 2_ε it MIGHT vanish! Use `\@nocounterr{`⟨*cnt*⟩`}` instead.

```
\gdef\@nocounterr#1{%
  \@latex@error{No counter '#1' defined}\@eha}
\gdef\@nocnterr{\@nocounterr?}
```

\@ctrerr Called when trying to print the value of a counter numbered by letters that's greater than 26.

```
\gdef\@ctrerr{%
  \@latex@error{Counter too large}\@ehb}
```

\@nodocument Error produced if paragraphs are typeset in the preamble.

```
\gdef\@nodocument{%
  \@latex@error{Missing \protect\begin{document}}\@ehd}
```

\@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.

```
\gdef\@badend#1{%
  \@latex@error{\protect\begin{\@currenvir}\@currenvline
                     \space ended by \protect\end{#1}}\@eha}
```

\@badmath Called by \[, \], \(or \) when used in wrong mode.

```
\gdef\@badmath{%
  \@latex@error{Bad math environment delimiter}\@eha}
```

\@toodeep Called by a list environment nested more than six levels deep, or an enumerate or itemize nested more than four levels.

```
\gdef\@toodeep{%
  \@latex@error{Too deeply nested}\@ehd}
```

\@badpoptabs Called by \endtabbing when not enough \poptabs have occurred, or by \poptabs when too many have occurred.

```
\gdef\@badpoptabs{%
  \@latex@error{\protect\pushtabs\space and \protect\poptabs
      \space don't match}\@ehd}
```

\@badtab Called by \>, \+ , \- or \< when stepping to an undefined tab.

```
\gdef\@badtab{%
 \@latex@error{Undefined tab position}\@ehd}
```

\@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.

```
\gdef\@preamerr#1{%
  \begingroup
    \let\protect\relax
    \@latex@error{\ifcase #1 Illegal character\or
     Missing @-exp\or Missing p-arg\fi\space
     in array arg}\@ehd
  \endgroup}
```

\@badlinearg Occurs in \line and \vector command when a bad slope argument is encountered.

```
\gdef\@badlinearg{%
  \@latex@error{%
       Bad \protect\line\space or \protect\vector
       \space argument}\@ehb}
```

`\@parmoderr` Occurs in a float environment or a `\marginpar` when encountered in inner vertical mode.

```
\gdef\@parmoderr{%
  \@latex@error{Not in outer par mode}\@ehb}
```

`\@fltovf` Occurs in float environment or `\marginpar` when there are no more free boxes for storing floats.

```
\gdef\@fltovf{%
  \@latex@error{Too many unprocessed floats}\@ehb}
```

`\@latexbug` Occurs in output routine. This is bad news.

```
\gdef\@latexbug{%
  \@latex@error{This may be a LaTeX bug}{Call for help}}
```

`\@badcrerr` This error was removed and replaced by `\@nolnerr`.

```
%\def\@badcrerr {\@latex@error{Bad use of \protect\\}\@ehc}
```

`\@noitemerr` `\addvspace` or `\addpenalty` was called when not in vmode. Probably caused by a missing `\item`.

```
\gdef\@noitemerr{%
  \@latex@error{Something's wrong--perhaps a missing %
      \protect\item}\@ehc}
```

`\@notprerr` A command that can be used only in the preamble appears after the command `\begin{document}`.

```
\gdef\@notprerr{%
  \@latex@error{Can be used only in preamble}\@eha}
```

`\@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.

```
\gdef\@inmatherr#1{%
  \relax
  \ifmmode
   \@latex@error{Command \protect#1 invalid in math mode}\@ehc
  \fi}
```

`\@invalidchar` An error for use with invalid characters. This is commented out, since we decided to use catcode 15 instead.

```
%\def\@invalidchar{\@latex@error{Invalid character in input}\@ehc}
```

As well as the above error commands some error messages are directly coded to save space. The messages already present in LaTeX2.09 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.

15.3 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 `\contionally@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`
`\conditionally@traceoff`

These are only dummy definitions. For details see the `trace` package.

```
\let\conditionally@traceon\@empty
\let\conditionally@traceoff\@empty
```

```
⟨/2ekernel⟩
```

File h
ltpar.dtx

16 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.

16.1 Implementation

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.

`\@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.

```
⟨*2ekernel⟩
\message{par,}
```

`\@setpar` `\@par` Initiate a long-term change to `\par`.

```
\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.

```
\def\@par{\let\par\@@par\par}
```

`\@restorepar` Restore from a short-term change to `\par`.

```
\def\@restorepar{\def\par{\@par}}
⟨/2ekernel⟩
```

File i

ltspace.dtx

17 Spacing

This section deals with spacing, and line- and page-breaking.

17.1 User Commands

\nopagebreak [⟨*i*⟩] : ⟨*i*⟩ = 0,...,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 [⟨*i*⟩] : same as except negatives of its penalty

\linebreak [⟨*i*⟩] : analog of the above

\nolinebreak [⟨*i*⟩] : 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

\\\\[⟨*length*⟩] : initially defined to be \vspace{⟨*length*⟩}\newline
Note: \\* adds a \vadjust{\penalty 10000}

OBSOLETE COMMANDS (which never made it into the manual):

\obeycr : defines ¡CR¿ == \\\\\relax
\restorecr : restores ¡CR¿ to its usual meaning.

17.2 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}
```

17.3 Some immediate actions

- Fix bug in \linebreak.
- Fix bug in \\*.
- Reimplement \\, etc, removing extra \vadjusts 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 TEX itself.

17.4 The code

```
⟨*2ekernel⟩
\message{spacing,}
```

```
⟨/2ekernel⟩
⟨*2ekernel | latexrelease⟩
⟨latexrelease⟩\IncludeInRelease{2019/10/01}%
⟨latexrelease⟩                 {\pagebreak}{Make commands robust}%
```

`\pagebreak`
`\nopagebreak`

```
\DeclareRobustCommand\pagebreak{\@testopt{\@no@pgbk-}4}
\DeclareRobustCommand\nopagebreak{\@testopt\@no@pgbk4}
```

`\linebreak`
`\nolinebreak`

```
\DeclareRobustCommand\linebreak{\@testopt{\@no@lnbk-}4}
\DeclareRobustCommand\nolinebreak{\@testopt\@no@lnbk4}
```

`\samepage`

```
\DeclareRobustCommand\samepage{\interlinepenalty\@M
   \postdisplaypenalty\@M
   \interdisplaylinepenalty\@M
   \@beginparpenalty\@M
   \@endparpenalty\@M
   \@itempenalty\@M
   \@secpenalty\@M
   \interfootnotelinepenalty\@M}
```

```
⟨/2ekernel | latexrelease⟩
⟨latexrelease⟩\EndIncludeInRelease
⟨latexrelease⟩\IncludeInRelease{0000/00/00}%
⟨latexrelease⟩                 {\pagebreak}{Make commands robust}%
⟨latexrelease⟩
⟨latexrelease⟩\kernel@make@fragile\pagebreak
⟨latexrelease⟩\kernel@make@fragile\nopagebreak
⟨latexrelease⟩\kernel@make@fragile\linebreak
⟨latexrelease⟩\kernel@make@fragile\nolinebreak
⟨latexrelease⟩\kernel@make@fragile\samepage
⟨latexrelease⟩
⟨latexrelease⟩\EndIncludeInRelease
⟨*2ekernel⟩
```

\@no@pgbk

```
\def\@no@pgbk #1[#2]{%
  \ifvmode
    \penalty #1\@getpen{#2}%
  \else
    \@bsphack
    \vadjust{\penalty #1\@getpen{#2}}%
    \@esphack
  \fi}
```

\@no@lnbk

```
\def\@no@lnbk #1[#2]{%
  \ifvmode
    \@nolnerr
  \else
    \@tempskipa\lastskip
    \unskip
    \penalty #1\@getpen{#2}%
    \ifdim\@tempskipa>\z@
      \hskip\@tempskipa
      \ignorespaces
    \fi
  \fi}
```

\\\\ 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 `\\`.

\@normalcr The internal definition of the 'normal' definition of `\\`.

```
⟨/2ekernel⟩
⟨*2ekernel | latexrelease⟩
⟨latexrelease⟩\IncludeInRelease{2020/02/02}%
⟨latexrelease⟩                 {\@normalcr}{Make robust}%
\protected\def\@normalcr{%
  \let \reserved@e \relax
  \let \reserved@f \relax
  \@ifstar{\let \reserved@e \vadjust \let \reserved@f \nobreak
           \@xnewline}%
  \@xnewline}
```

```
\let\\\@normalcr
⟨/2ekernel | latexrelease⟩
⟨latexrelease⟩\EndIncludeInRelease
⟨latexrelease⟩\IncludeInRelease{0000/00/00}%
⟨latexrelease⟩                 {\@normalcr}{Make robust}%
⟨latexrelease⟩
⟨latexrelease⟩\DeclareRobustCommand\\{%
⟨latexrelease⟩   \let \reserved@e \relax
⟨latexrelease⟩   \let \reserved@f \relax
⟨latexrelease⟩   \@ifstar{\let \reserved@e \vadjust \let \reserved@f \nobreak
⟨latexrelease⟩            \@xnewline}%
⟨latexrelease⟩  \@xnewline}
⟨latexrelease⟩\expandafter\let\expandafter\@normalcr
⟨latexrelease⟩     \csname\expandafter\@gobble\string\\ \endcsname
⟨latexrelease⟩
⟨latexrelease⟩\EndIncludeInRelease
⟨*2ekernel⟩
```

\newline A simple form of the 'normal' definition of \\.

```
\DeclareRobustCommand\newline{\@normalcr\relax}
```

\@xnewline

```
\def\@xnewline{\@ifnextchar[% ] bracket matching
                  \@newline
                 {\@gnewline\relax}}
```

\@newline

```
\def\@newline[#1]{\let \reserved@e \vadjust
                   \@gnewline {\vskip #1}}
```

\@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

```
\def\@gnewline #1{%
  \ifvmode
    \@nolnerr
  \else
    \unskip \reserved@e {\reserved@f#1}\nobreak \hfil \break
  \fi}
```

\@getpen

```
\def\@getpen#1{\ifcase #1 \z@ \or \@lowpenalty\or
         \@medpenalty \or \@highpenalty
         \else \@M \fi}
```

\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.

```
\def\@nobreakfalse{\global\let\if@nobreak\iffalse}
\def\@nobreaktrue {\global\let\if@nobreak\iftrue}
\@nobreakfalse
```

`\@savsk` `\@savsf` Registers used to save the space factor and last skip.

```
\newdimen\@savsk
\newcount\@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 `\addvspace`, 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
}
```

For the moment we are going to ignore the vertical versions until they are correct.

```
\def\@bsphack{%
 \relax
 \ifhmode
  \@savsk\lastskip
  \@savsf\spacefactor
 \fi}
```

\@esphack Companion to \@bsphack. If this command is not properly paired with \@bsphack one might end up with a low-level TEX 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.

```
⟨/2ekernel⟩
⟨latexrelease⟩\IncludeInRelease{2018/10/10}%
⟨latexrelease⟩                 {\@esphack}{hyphenation and nobreak after space hack}%
⟨*2ekernel | latexrelease⟩
\def\@esphack{%
 \relax
 \ifhmode
  \spacefactor\@savsf
  \ifdim\@savsk>\z@

   \ifdim\lastskip=\z@
    \nobreak \hskip\z@skip
   \fi
   \ignorespaces
  \fi

 \else
  \ifvmode
   \if@nobreak\nobreak\else\if@noskipsec\nobreak\fi\fi
  \fi

 \fi}%
⟨/2ekernel | latexrelease⟩
⟨latexrelease⟩\EndIncludeInRelease
⟨latexrelease⟩\IncludeInRelease{2015/10/01}%
⟨latexrelease⟩                 {\@esphack}{hyphenation and nobreak after space hack}%
⟨latexrelease⟩\def\@esphack{%
⟨latexrelease⟩ \relax
⟨latexrelease⟩ \ifhmode
⟨latexrelease⟩  \spacefactor\@savsf
⟨latexrelease⟩  \ifdim\@savsk>\z@
⟨latexrelease⟩      \ifdim\lastskip=\z@
⟨latexrelease⟩        \nobreak \hskip\z@skip
⟨latexrelease⟩      \fi
⟨latexrelease⟩      \ignorespaces
⟨latexrelease⟩    \fi
⟨latexrelease⟩  \fi}%
⟨latexrelease⟩\EndIncludeInRelease
⟨latexrelease⟩\IncludeInRelease{2015/01/01}%
⟨latexrelease⟩                 {\@esphack}{hyphenation and nobreak after space hack}%
⟨latexrelease⟩\def\@esphack{%
⟨latexrelease⟩  \relax
⟨latexrelease⟩  \ifhmode
⟨latexrelease⟩    \spacefactor\@savsf
⟨latexrelease⟩    \ifdim\@savsk>\z@
⟨latexrelease⟩      \nobreak \hskip\z@skip
⟨latexrelease⟩      \ignorespaces
⟨latexrelease⟩    \fi
⟨latexrelease⟩  \fi}%
⟨latexrelease⟩\EndIncludeInRelease
⟨latexrelease⟩\IncludeInRelease{0000/00/00}%
⟨latexrelease⟩                 {\@esphack}{hyphenation and nobreak after space hack}%
⟨latexrelease⟩\def\@esphack{%
⟨latexrelease⟩  \relax
⟨latexrelease⟩  \ifhmode
⟨latexrelease⟩    \spacefactor\@savsf
⟨latexrelease⟩    \ifdim\@savsk>\z@
⟨latexrelease⟩      \ignorespaces
⟨latexrelease⟩    \fi
⟨latexrelease⟩  \fi}%
⟨latexrelease⟩\EndIncludeInRelease
⟨*2ekernel⟩
```

`\@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

```
⟨/2ekernel⟩
⟨latexrelease⟩\IncludeInRelease{2015/01/01}%
⟨latexrelease⟩                 {\@Esphack}{hyphenation after space hack}%
⟨*2ekernel | latexrelease⟩
\def\@Esphack{%
  \relax
  \ifhmode
    \spacefactor\@savsf
    \ifdim\@savsk>\z@
      \nobreak \hskip\z@skip
      \@ignoretrue
      \ignorespaces
    \fi
  \fi}%
⟨/2ekernel | latexrelease⟩
⟨latexrelease⟩\EndIncludeInRelease
⟨latexrelease⟩\IncludeInRelease{0000/00/00}%
⟨latexrelease⟩                 {\@Esphack}{hyphenation after space hack}%
⟨latexrelease⟩\def\@Esphack{%
⟨latexrelease⟩  \relax
⟨latexrelease⟩  \ifhmode
⟨latexrelease⟩    \spacefactor\@savsf
⟨latexrelease⟩    \ifdim\@savsk>\z@
⟨latexrelease⟩      \@ignoretrue
⟨latexrelease⟩      \ignorespaces
⟨latexrelease⟩    \fi
⟨latexrelease⟩  \fi}%
⟨latexrelease⟩\EndIncludeInRelease
⟨*2ekernel⟩
```

`\@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
}
```

17.5 Vertical spacing

LaTeX supports the plain TeX 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.

```
\def\@xaddvskip{%
  \ifdim\lastskip<\@tempskipb
    \vskip-\lastskip
    \vskip\@tempskipb
  \else
    \ifdim\@tempskipb<\z@
      \ifdim\lastskip<\z@
      \else
        \advance\@tempskipb\lastskip
        \vskip-\lastskip
        \vskip \@tempskipb
      \fi
    \fi
  \fi}
```

\addvspace Add vertical space taking into account space already added, as described above.

```
\def\addvspace#1{%
  \ifvmode
     \if@minipage\else
       \ifdim \lastskip =\z@
         \vskip #1\relax
       \else
       \@tempskipb#1\relax
         \@xaddvskip
       \fi
     \fi
  \else
    \@noitemerr
  \fi}
```

\addpenalty

```
⟨/2ekernel⟩
⟨latexrelease⟩\IncludeInRelease{2015/01/01}%
⟨latexrelease⟩                      {\addpenalty}{\addpenalty}%
⟨*2ekernel | 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.

```
\def\addpenalty#1{%
  \ifvmode
    \if@minipage
    \else
      \if@nobreak
      \else
        \ifdim\lastskip=\z@
          \penalty#1\relax
        \else
          \@tempskipb\lastskip
```

We have to make sure the final `\vskip` seen by TEX 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 TEX.

```
          \begingroup
            \@tempskipa\@tempskipb
            \advance \@tempskipb
              \ifdim\prevdepth>\maxdepth\maxdepth\else
```

If `\prevdepth` is -1000pt due to `\nointerlineskip` we better not add it!

```
                 \ifdim \prevdepth = -\@m\p@ \z@ \else \prevdepth \fi
               \fi
             \vskip -\@tempskipb
             \penalty#1%
             \ifdim\@tempskipa=\@tempskipb
```

Do nothing if the `\prevdepth` check made no adjustment.

```
             \else
```

Combine the prevdepth adjustment into a single skip.

```
               \advance\@tempskipb -\@tempskipa
               \vskip \@tempskipb
             \fi
```

The final skip is always the specified length.

```
             \vskip \@tempskipa
          \endgroup
        \fi
      \fi
    \fi
  \else
    \@noitemerr
  \fi}%
```

```
⟨/2ekernel | latexrelease⟩
⟨latexrelease⟩\EndIncludeInRelease
⟨latexrelease⟩\IncludeInRelease{0000/00/00}%
⟨latexrelease⟩                 {\addpenalty}{\addpenalty}%
⟨latexrelease⟩\def\addpenalty#1{%
⟨latexrelease⟩  \ifvmode
⟨latexrelease⟩    \if@minipage
⟨latexrelease⟩    \else
⟨latexrelease⟩      \if@nobreak
⟨latexrelease⟩      \else
⟨latexrelease⟩        \ifdim\lastskip=\z@
⟨latexrelease⟩          \penalty#1\relax
⟨latexrelease⟩        \else
⟨latexrelease⟩          \@tempskipb\lastskip
⟨latexrelease⟩          \vskip -\lastskip
⟨latexrelease⟩          \penalty#1%
⟨latexrelease⟩          \vskip\@tempskipb
⟨latexrelease⟩        \fi
⟨latexrelease⟩      \fi
⟨latexrelease⟩    \fi
⟨latexrelease⟩  \else
⟨latexrelease⟩    \@noitemerr
⟨latexrelease⟩  \fi}%
⟨latexrelease⟩\EndIncludeInRelease
⟨*2ekernel⟩
```

\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.

```
\DeclareRobustCommand\vspace{\@ifstar\@vspacer\@vspace}
\def\@vspace #1{%
  \ifvmode
    \vskip #1
    \vskip\z@skip
   \else
     \@bsphack
     \vadjust{\@restorepar
              \vskip #1
              \vskip\z@skip
              }%
     \@esphack
   \fi}
```

```
\def\@vspacer#1{%
  \ifvmode
    \dimen@\prevdepth
    \hrule \@height\z@
    \nobreak
    \vskip #1
    \vskip\z@skip
    \prevdepth\dimen@
  \else
    \@bsphack
    \vadjust{\@restorepar
             \hrule \@height\z@
             \nobreak
             \vskip #1
             \vskip\z@skip}%
    \@esphack
  \fi}
```

\smallskip
\medskip
\bigskip

```
\def\smallskip{\vspace\smallskipamount}
\def\medskip{\vspace\medskipamount}
\def\bigskip{\vspace\bigskipamount}
```

\smallskipamount
\medskipamount
\bigskipamount

```
\newskip\smallskipamount \smallskipamount=3pt plus 1pt minus 1pt
\newskip\medskipamount   \medskipamount  =6pt plus 2pt minus 2pt
\newskip\bigskipamount   \bigskipamount =12pt plus 4pt minus 4pt
```

17.6 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!

```
\DeclareRobustCommand{\nobreakdashes}{%
  \leavevmode
  \toks@{}%
  \def\reserved@a##1{\toks@\expandafter{\the\toks@-}%
                    \futurelet\@let@token \reserved@b}%
  \def\reserved@b  {\ifx\@let@token -%
                       \expandafter\reserved@a
                    \else
                      \setbox\z@ \hbox{\the\toks@\nobreak}%
                      \unhbox\z@
                      \spacefactor\sfcode`\-
                    \fi}%
  \futurelet\@let@token \reserved@b
}
```

\nobreakspace
\@xobeysp

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 LaTeX internal commands.

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 .

```
\DeclareRobustCommand{\nobreakspace}{%
   \leavevmode\nobreak\ }
\catcode `\~=13
\def~{\nobreakspace{}}
\expandafter\let\expandafter\@xobeysp\csname nobreakspace \endcsname
```

`\,` Used in paragraph mode produces a `\thinspace`. It has the ordinary definition in math mode. Useful for quotes inside quotes, as in ``` ``\,`Foo', he said.'' ```

```
\DeclareRobustCommand{\,}{%
   \relax\ifmmode\mskip\thinmuskip\else\thinspace\fi
}
```

`\@` 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.

```
⟨/2ekernel⟩
⟨latexrelease⟩\IncludeInRelease{2015/01/01}%
⟨latexrelease⟩                 {\@}{Space after \@}%
⟨*2ekernel | latexrelease⟩

\def\@{\spacefactor\@m{}}%
⟨/2ekernel | latexrelease⟩
⟨latexrelease⟩\EndIncludeInRelease
⟨latexrelease⟩\IncludeInRelease{0000/00/00}%
⟨latexrelease⟩                 {\@}{Space after \@}%
⟨latexrelease⟩\def\@{\spacefactor\@m}%
⟨latexrelease⟩\EndIncludeInRelease
⟨*2ekernel⟩
```

`\hspace`

```
\DeclareRobustCommand\hspace{\@ifstar\@hspacer\@hspace}
```

`\@hspace`

```
\def\@hspace#1{\hskip #1\relax}
```

`\@hspacer` extra `\hskip 0pt` added 1985/17/12 to guard against a following `\unskip` `\relax` added 13 Oct 88 for usual TEX lossage replaced both changes by `\hskip\z@skip` 27 Nov 91

```
\def\@hspacer#1{\vrule \@width\z@\nobreak
                \hskip #1\hskip \z@skip}
```

`\fill`

```
\newskip\fill
\fill = 0pt plus 1fill
```

`\stretch`

```
\def\stretch#1{\z@ \@plus #1fill\relax}

⟨/2ekernel⟩
⟨*2ekernel | latexrelease⟩
⟨latexrelease⟩\IncludeInRelease{2018/12/01}%
⟨latexrelease⟩                 {\thinspace}{Start LR-mode}%
```

`\thinspace`
`\negthinspace`
`\enspace`

```
\DeclareRobustCommand\thinspace{\leavevmode@ifvmode\kern .16667em }
\DeclareRobustCommand\negthinspace{\leavevmode@ifvmode\kern-.16667em }
\DeclareRobustCommand\enspace{\leavevmode@ifvmode\kern.5em }
```

`\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).

```
\protected\def\leavevmode@ifvmode{\ifvmode\expandafter\indent\fi}
```

```
⟨/2ekernel | latexrelease⟩
⟨latexrelease⟩\EndIncludeInRelease
⟨latexrelease⟩\IncludeInRelease{0000/00/00}%
⟨latexrelease⟩                 {\thinspace}{Start LR-mode}%
⟨latexrelease⟩\def\thinspace{\kern .16667em }
⟨latexrelease⟩\def\negthinspace{\kern-.16667em }
⟨latexrelease⟩\def\enspace{\kern.5em }
⟨latexrelease⟩\let\leavevmode@ifvmode\@undefined
⟨latexrelease⟩\EndIncludeInRelease
⟨*2ekernel⟩
```

\enskip
\quad
\qquad

```
\def\enskip{\hskip.5em\relax}
\def\quad{\hskip1em\relax}
\def\qquad{\hskip2em\relax}
```

\obeycr
\restorecr

The following definitions will probably get deleted or moved to compatibility mode soon.

```
{\catcode`\^^M=13 \gdef\obeycr{\catcode`\^^M13 \def^^M{\\\relax}%
    \@gobblecr}%
{\catcode`\^^M=13 \gdef\@gobblecr{\@ifnextchar
\@gobble\ignorespaces}}
\gdef\restorecr{\catcode`\^^M5 }}
```

```
⟨/2ekernel⟩
```

File j
ltlogos.dtx

18 Logos

Various logos are defined here.

\TeX The TeX logo, adjusted so that a full stop after the logo counts as ending a sentence.

```
⟨*2ekernel⟩
\DeclareRobustCommand\TeX{T\kern-.1667em\lower.5ex\hbox{E}\kern-.125emX\@}
```

\LaTeX The LaTeX logo.

```
\DeclareRobustCommand{\LaTeX}{L\kern-.36em%
        {\sbox\z@ T%
         \vbox to\ht\z@{\hbox{\check@mathfonts
                              \fontsize\sf@size\z@
                              \math@fontsfalse\selectfont
                              A}%
                        \vss}%
        }%
        \kern-.15em%
        \TeX}
```

\LaTeXe The LaTeX 2$_\varepsilon$ logo as proposed by A-W designers.

```
\DeclareRobustCommand{\LaTeXe}{\mbox{\m@th
  \if b\expandafter\@car\f@series\@nil\boldmath\fi
  \LaTeX\kern.15em2$_{\textstyle\varepsilon}$}}
⟨/2ekernel⟩
```

File k
ltfiles.dtx

19 File Handling

The following user commands are defined in this part:

`\document` (ie `\begin{document}`)
Reads in the .AUX files and `\catcode`'s `@` to 12.

`\nofiles`
Suppresses all file output by setting `\@filesw` false.

`\includeonly` {⟨*NAME1, ... ,NAMEn*⟩}
Causes only parts NAME1, ... ,NAMEn to be read by their `\include` commands. Works by setting partsw true and setting `\@partlist` to NAME1, ... ,NAMEn.

`\include` {⟨*NAME*⟩}
Does an `\input` NAME unless `\@partsw` is true and NAME is not in `\@partlist`. If `\@filesw` is true, then it directs .AUX output to NAME.AUX, including a checkpoint at the end.

`\input` {⟨*NAME*⟩}
The same as TeX's `\input`, except it allows optional braces around the file name. In LaTeX 2$_\varepsilon$, it also avoids the primitive 'missing file' error, if the file can not be found.

`\IfFileExists` {⟨*NAME*⟩}{⟨*then*⟩}{⟨*else*⟩}
If the file exists on the system, execute *then* otherwise execute *else*.

`\InputIfFileExists` {⟨*NAME*⟩}{⟨*then*⟩}{⟨*else*⟩}
If the file exists on the system, execute *then* and input *NAME* otherwise execute *else*.

Historical LaTeX 2.09 comments (not necessarily accurate any more):

```
⟨*2ekernel⟩
\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.
  @filesw      : 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@FOO      : 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 \@filesw = T
     then  \immediate\write\@mainaux{\string\@input{FILE.AUX}}
   fi
   if  \@partsw = T
     then \@tempswa := F
          \reserved@b == FILE
          for \reserved@a := \@partlist
              do if eval(\reserved@a) = eval(\reserved@b)
                   then \@tempswa := T          fi
              od
   fi

   if \@tempswa = T
      then \@auxout := \@partaux
           if \@filesw = T
             then  \immediate\openout\@partaux{FILE.AUX}
                   \immediate\write\@partaux{\relax}
           fi
           \@input{FILE.TEX}
           \clearpage
           \@writeckpt{FILE}
           if @filesw then \closeout \@partaux fi
           \@auxout := \@mainaux
      else \cp@FILE
   fi
  END

 \@writeckpt{FILE} ==
  BEGIN
    if \@filesw = 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 LaTeX 2.09 comments.

\@inputcheck
\@unused

Allocate read stream for testing and output stream.

```
\newread\@inputcheck
\newwrite\@unused
```

\@mainaux
\@partaux

```
\newwrite\@mainaux
\newwrite\@partaux
```

\if@filesw
\if@partsw

```
\newif\if@filesw \@fileswtrue
\newif\if@partsw \@partswfalse
```

\@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).

```
\newcount\@clubpenalty
\@clubpenalty \clubpenalty
```

\document

```
⟨/2ekernel⟩
⟨latexrelease⟩\IncludeInRelease{2017/04/15}%
⟨latexrelease⟩  {\document}{Save language for hyphenation}%
⟨*2ekernel | latexrelease⟩
```

Cancel the `\begingroup` from `\begin`.

```
\def\document{\endgroup
```

If some options on `\documentclass` haven't been used by any package we will now give a warning since this is most certainly a misspelling.

```
  \ifx\@unusedoptionlist\@empty\else
    \@latex@warning@no@line{Unused global option(s):^^J%
            \@spaces[\@unusedoptionlist]}%
  \fi
  \@colht\textheight
  \@colroom\textheight \vsize\textheight
  \columnwidth\textwidth
  \@clubpenalty\clubpenalty
  \if@twocolumn
    \advance\columnwidth -\columnsep
    \divide\columnwidth\tw@ \hsize\columnwidth \@firstcolumntrue
  \fi
  \hsize\columnwidth \linewidth\hsize
  \begingroup\@floatplacement\@dblfloatplacement
    \makeatletter\let\@writefile\@gobbletwo
    \global \let \@multiplelabels \relax
    \@input{\jobname.aux}%
  \endgroup
  \if@filesw
    \immediate\openout\@mainaux\jobname.aux
    \immediate\write\@mainaux{\relax}%
  \fi
```

Dateline 1991/03/26: FMi added `\process@table` to support NFSS; This will also work with old lfonts if no other style defines `\process@table`. The following line forces the initialization of the math fonts.

```
\process@table
\let\glb@currsize\@empty  % Force math initialization.
```

```
\normalsize
\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 LaTeX2.0x and plain TeX.)

```
\ifx\normalsfcodes\@empty
  \ifnum\sfcode`\.=\@m
    \let\normalsfcodes\frenchspacing
  \else
    \let\normalsfcodes\nonfrenchspacing
  \fi
\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 spoint.

```
\ifx\document@default@language\m@ne
  \chardef\document@default@language\language
\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.

```
\@noskipsecfalse
```

```
\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…

```
\let\AtBeginDocument\@firstofone
\@begindocumenthook
```

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.

```
\ifdim\topskip<1sp\global\topskip 1sp\relax\fi
\global\@maxdepth\maxdepth
\global\let\@begindocumenthook\@undefined
\ifx\@listfiles\@undefined
  \global\let\@filelist\relax
  \global\let\@addtofilelist\@gobble
\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.

```
\gdef\do##1{\global\let ##1\@notprerr}%
\@preamblecmds
```

The next line saves tokens and also allows `\@nodocument` to be used directly to trap preamble errors.

```
\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.

```
\global\let\do\noexpand
```

Use of `\AtBeginDocument` hook might mean that we are already in horizontal mode, so ignore the space after `\begin{document}`.

```
\ignorespaces}
⟨/2ekernel | latexrelease⟩
⟨latexrelease⟩\EndIncludeInRelease
⟨latexrelease⟩\IncludeInRelease{0000/00/00}%
⟨latexrelease⟩  {\document}{Save language for hyphenation}
⟨latexrelease⟩\def\document{\endgroup
⟨latexrelease⟩  \ifx\@unusedoptionlist\@empty\else
⟨latexrelease⟩    \@latex@warning@no@line{Unused global option(s):^^J%
⟨latexrelease⟩            \@spaces[\@unusedoptionlist]}%
⟨latexrelease⟩  \fi
⟨latexrelease⟩  \@colht\textheight
⟨latexrelease⟩  \@colroom\textheight \vsize\textheight
⟨latexrelease⟩  \columnwidth\textwidth
⟨latexrelease⟩  \@clubpenalty\clubpenalty
⟨latexrelease⟩  \if@twocolumn
⟨latexrelease⟩    \advance\columnwidth -\columnsep
⟨latexrelease⟩    \divide\columnwidth\tw@ \hsize\columnwidth
⟨latexrelease⟩    \@firstcolumntrue
⟨latexrelease⟩  \fi
⟨latexrelease⟩  \hsize\columnwidth \linewidth\hsize
⟨latexrelease⟩  \begingroup\@floatplacement\@dblfloatplacement
⟨latexrelease⟩    \makeatletter\let\@writefile\@gobbletwo
⟨latexrelease⟩    \global \let \@multiplelabels \relax
⟨latexrelease⟩    \@input{\jobname.aux}%
⟨latexrelease⟩  \endgroup
⟨latexrelease⟩  \if@filesw
⟨latexrelease⟩    \immediate\openout\@mainaux\jobname.aux
⟨latexrelease⟩    \immediate\write\@mainaux{\relax}%
⟨latexrelease⟩  \fi
```

```
⟨latexrelease⟩  \process@table
⟨latexrelease⟩  \let\glb@currsize\@empty
⟨latexrelease⟩  \normalsize
⟨latexrelease⟩  \everypar{}%
⟨latexrelease⟩  \ifx\normalsfcodes\@empty
⟨latexrelease⟩    \ifnum\sfcode`\.=\@m
⟨latexrelease⟩      \let\normalsfcodes\frenchspacing
⟨latexrelease⟩    \else
⟨latexrelease⟩      \let\normalsfcodes\nonfrenchspacing
⟨latexrelease⟩    \fi
⟨latexrelease⟩  \fi
⟨latexrelease⟩  \@noskipsecfalse
⟨latexrelease⟩  \let \@refundefined \relax
⟨latexrelease⟩  \let\AtBeginDocument\@firstofone
⟨latexrelease⟩  \@begindocumenthook
⟨latexrelease⟩  \ifdim\topskip<1sp\global\topskip 1sp\relax\fi
⟨latexrelease⟩  \global\@maxdepth\maxdepth
⟨latexrelease⟩  \global\let\@begindocumenthook\@undefined
⟨latexrelease⟩  \ifx\@listfiles\@undefined
⟨latexrelease⟩    \global\let\@filelist\relax
⟨latexrelease⟩    \global\let\@addtofilelist\@gobble
⟨latexrelease⟩  \fi
⟨latexrelease⟩  \gdef\do##1{\global\let ##1\@notprerr}%
⟨latexrelease⟩  \@preamblecmds
⟨latexrelease⟩  \global\let \@nodocument \relax
⟨latexrelease⟩  \global\let\do\noexpand
⟨latexrelease⟩  \ignorespaces}
⟨latexrelease⟩\EndIncludeInRelease
⟨*2ekernel⟩
```

```
\@onlypreamble\document
```

`\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.

```
\let\normalsfcodes\@empty
```

`\nofiles` Set `\@fileswfalse` which suppresses the places where LaTeX 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.

```
\def\nofiles{%
  \@fileswfalse
  \typeout{No auxiliary output files.^^J}%
  \long\def\protected@write##1##2##3%
    {\write\m@ne{}\if@nobreak\ifvmode\nobreak\fi\fi}%
  \let\makeindex\relax
  \let\makeglossary\relax}
\@onlypreamble\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.

```
\long\def \protected@write#1#2#3{%
      \begingroup
       \let\thepage\relax
       #2%
       \let\protect\@unexpandable@protect
       \edef\reserved@a{\write#1{#3}}%
       \reserved@a
      \endgroup
      \if@nobreak\ifvmode\nobreak\fi\fi
}
```

```
\let\@auxout=\@mainaux
```

\includeonly

```
⟨/2ekernel⟩
⟨*2ekernel | latexrelease⟩
⟨latexrelease⟩\IncludeInRelease{2019/10/01}%
⟨latexrelease⟩                 {\includeonly}{Spaces in file names}%
\def\includeonly#1{%
  \@partswtrue
  \set@curr@file{\zap@space#1 \@empty}%
  \let\@partlist\@curr@file
  }
\@onlypreamble\includeonly
```

\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 LaTeX 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.

```
\def\include#1{\relax
  \ifnum\@auxout=\@partaux
    \@latex@error{\string\include\space cannot be nested}\@eha
  \else
  \set@curr@file{#1 }%
  \expandafter\@include\@curr@file
  \fi}
```

```
⟨/2ekernel | latexrelease⟩
⟨latexrelease⟩\EndIncludeInRelease
⟨latexrelease⟩\IncludeInRelease{0000/00/00}%
⟨latexrelease⟩                 {\includeonly}{Spaces in file names}%
⟨latexrelease⟩\def\includeonly#1{%
⟨latexrelease⟩  \@partswtrue
⟨latexrelease⟩  \edef\@partlist{\zap@space#1 \@empty}}
⟨latexrelease⟩
⟨latexrelease⟩\def\include#1{\relax
⟨latexrelease⟩  \ifnum\@auxout=\@partaux
⟨latexrelease⟩     \@latex@error{\string\include\space cannot be nested}\@eha
⟨latexrelease⟩   \else \@include#1 \fi}
⟨latexrelease⟩
⟨latexrelease⟩\EndIncludeInRelease
⟨*2ekernel⟩
```

\@include

```
\def\@include#1 {%
  \clearpage
  \if@filesw
    \immediate\write\@mainaux{\string\@input{#1.aux}}%
  \fi
  \@tempswatrue
  \if@partsw
    \@tempswafalse
    \edef\reserved@b{#1}%
    \@for\reserved@a:=\@partlist\do
      {\ifx\reserved@a\reserved@b\@tempswatrue\fi}%
  \fi
  \if@tempswa
    \let\@auxout\@partaux
    \if@filesw
      \immediate\openout\@partaux #1.aux
      \immediate\write\@partaux{\relax}%
    \fi
    \@input@{#1.tex}%
    \clearpage
    \@writeckpt{#1}%
    \if@filesw
      \immediate\closeout\@partaux
    \fi
  \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.

```
    \deadcycles\z@
    \@nameuse{cp@#1}%
  \fi
  \let\@auxout\@mainaux}
```

\@writeckpt

```
\def\@writeckpt#1{%
  \if@filesw
    \immediate\write\@partaux{\string\@setckpt{#1}\@charlb}%
    {\let\@elt\@wckptelt \cl@@ckpt}%
    \immediate\write\@partaux{\@charrb}%
  \fi}
```

\@wckptelt

```
\def\@wckptelt#1{%
  \immediate\write\@partaux{%
    \string\setcounter{#1}{\the\@nameuse{c@#1}}}}
```

\@setckpt RmS 93/08/31: introduced \@setckpt

```
\def\@setckpt#1{\global\@namedef{cp@#1}}
```

\@charlb \@charrb The following defines \@charlb and \@charrb to be { and }, respectively with \catcode 11.

```
{\catcode`[=1 \catcode`]=2
\catcode`{=11 \catcode`}=11
\gdef\@charlb[{]
\gdef\@charrb[}]
]% }brace matching
```

19.1 Safe Input Macros

\@curr@file \set@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 \escapchar 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 contruct (removing any existing inner quotes. Somes 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 compatibilty 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.

```
⟨/2ekernel⟩
⟨*2ekernel | latexrelease⟩
⟨latexrelease⟩\IncludeInRelease{2019/10/01}%
⟨latexrelease⟩                  {\set@curr@file}{Quote file names}%
\def\set@curr@file#1{%
  \begingroup
    \escapechar\m@ne
    \xdef\@curr@file{%
      \expandafter\expandafter\expandafter\unquote@name
      \expandafter\expandafter\expandafter{%
      \expandafter\string
        \csname\@firstofone#1\@empty\endcsname}}%
  \endgroup
}
```

\quote@name \quote@@name \unquote@name Quoting spaces

```
 a b c     -> "a b c"
 "a b c"   -> "a b c"
 a" "b" "c -> "a b c"
           -> ""
```

```
\def\quote@name#1{"\quote@@name#1\@gobble""}
\def\quote@@name#1"{#1\quote@@name}
```

and removing quotes . . .

```
\def\unquote@name#1{\quote@@name#1\@gobble"}
```

\IfFileExists

```
\DeclareRobustCommand\IfFileExists[1]{%
  \set@curr@file{#1}%
  \expandafter\IfFileExists@\expandafter{\@curr@file}}
```

\IfFileExists@ Argument #1 is \@curr@file so catcode 12 string with no quotes.

```
\long\def \IfFileExists@#1#2#3{%
  \openin\@inputcheck"#1" %
  \ifeof\@inputcheck
    \ifx\input@path\@undefined
      \def\reserved@a{#3}%
    \else
      \def\reserved@a{\@iffileonpath{#1}{#2}{#3}}%
    \fi
  \else
    \closein\@inputcheck
    \edef\@filef@und{"#1" }%
    \def\reserved@a{#2}%
  \fi
  \reserved@a}
```

\@iffileonpath If the file is not found by \openin, and \input@path is defined, look in all the directories specified in \input@path.

```
\long\def\@iffileonpath#1{%
  \let\reserved@a\@secondoftwo
  \expandafter\@tfor\expandafter\reserved@b\expandafter
             :\expandafter=\input@path\do{%
    \openin\@inputcheck\expandafter\quote@name\expandafter{\reserved@b#1} %
    \ifeof\@inputcheck\else
      \edef\@filef@und{\expandafter\quote@name\expandafter{\reserved@b#1} }%
      \let\reserved@a\@firstoftwo%
      \closein\@inputcheck
      \@break@tfor
    \fi}%
  \reserved@a}
```

```
⟨/2ekernel | latexrelease⟩
⟨latexrelease⟩\EndIncludeInRelease
⟨latexrelease⟩\IncludeInRelease{0000/00/00}%
⟨latexrelease⟩                 {\set@curr@file}{Quote file names}%
⟨latexrelease⟩
⟨latexrelease⟩\let\quote@name\@undefined
⟨latexrelease⟩\let\quote@@name\@undefined
⟨latexrelease⟩\let\unquote@name\@undefined
⟨latexrelease⟩\let\set@curr@file\@undefined
⟨latexrelease⟩
⟨latexrelease⟩\let\IfFileExists@\@undefined
⟨latexrelease⟩
⟨latexrelease⟩\long\def \IfFileExists#1#2#3{%
⟨latexrelease⟩  \openin\@inputcheck#1 %
⟨latexrelease⟩  \ifeof\@inputcheck
⟨latexrelease⟩    \ifx\input@path\@undefined
⟨latexrelease⟩      \def\reserved@a{#3}%
⟨latexrelease⟩    \else
⟨latexrelease⟩      \def\reserved@a{\@iffileonpath{#1}{#2}{#3}}%
⟨latexrelease⟩    \fi
⟨latexrelease⟩  \else
⟨latexrelease⟩    \closein\@inputcheck
⟨latexrelease⟩    \edef\@filef@und{#1 }%
⟨latexrelease⟩    \def\reserved@a{#2}%
⟨latexrelease⟩  \fi
⟨latexrelease⟩  \reserved@a}
⟨latexrelease⟩
⟨latexrelease⟩\long\def\@iffileonpath#1{%
⟨latexrelease⟩  \let\reserved@a\@secondoftwo
⟨latexrelease⟩  \expandafter\@tfor\expandafter\reserved@b\expandafter
⟨latexrelease⟩             :\expandafter=\input@path\do{%
⟨latexrelease⟩    \openin\@inputcheck\reserved@b#1 %
⟨latexrelease⟩    \ifeof\@inputcheck\else
⟨latexrelease⟩      \edef\@filef@und{\reserved@b#1 }%
⟨latexrelease⟩      \let\reserved@a\@firstoftwo%
⟨latexrelease⟩      \closein\@inputcheck
⟨latexrelease⟩      \@break@tfor
⟨latexrelease⟩    \fi}%
⟨latexrelease⟩  \reserved@a}
⟨latexrelease⟩
⟨latexrelease⟩\EndIncludeInRelease
⟨*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`'.

```
⟨/2ekernel⟩
⟨latexrelease⟩\IncludeInRelease{2019/10/01}%
⟨latexrelease⟩  {\InputIfFileExists}{Don't lose the file name}%
⟨*2ekernel | latexrelease⟩
%    \begin{macrocode}
\DeclareRobustCommand \InputIfFileExists[2]{%
  \IfFileExists{#1}%
    {%
  \expandafter\@swaptwoargs\expandafter
      {\@filef@und}{#2\@addtofilelist{#1}\@@input}}}
```

`\@swaptwoargs` Swap two arguments and return them unbraced (like `\@firstoftwo` etc).

```
\long\def\@swaptwoargs#1#2{#2#1}
```

```
⟨/2ekernel | latexrelease⟩
⟨latexrelease⟩\EndIncludeInRelease
⟨latexrelease⟩\IncludeInRelease{0000/00/00}%
⟨latexrelease⟩  {\InputIfFileExists}{Don't lose the file name}%
⟨latexrelease⟩\long\def \InputIfFileExists#1#2{%
⟨latexrelease⟩  \IfFileExists{#1}%
⟨latexrelease⟩    {#2\@addtofilelist{#1}\@@input \@filef@und}}
⟨latexrelease⟩
⟨latexrelease⟩\let\@swaptwoargs\@undefined
⟨latexrelease⟩\EndIncludeInRelease
⟨*2ekernel⟩
```

`\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 LATEX).

```
\def\input{\@ifnextchar\bgroup\@iinput\@@input}
```

\@iinput Define \@iinput (i.e., \input) in terms of \InputIfIfileExists.

```
⟨/2ekernel⟩
⟨*2ekernel | latexrelease⟩
⟨latexrelease⟩\IncludeInRelease{2019/10/01}%
⟨latexrelease⟩                 {\@iinput}{Quote file names}%
\def\@iinput#1{%
  \InputIfFileExists{#1}{}%
  {\filename@parse\@curr@file
   \edef\reserved@a{\noexpand\@missingfileerror
     {\filename@area\filename@base}%
     {\ifx\filename@ext\relax tex\else\filename@ext\fi}}%
   \reserved@a}}
⟨/2ekernel | latexrelease⟩
⟨latexrelease⟩\EndIncludeInRelease
⟨latexrelease⟩\IncludeInRelease{0000/00/00}%
⟨latexrelease⟩                 {\@iinput}{Quote file names}%
⟨latexrelease⟩\def\@iinput#1{%
⟨latexrelease⟩  \InputIfFileExists{#1}{}%
⟨latexrelease⟩  {\filename@parse{#1}%
⟨latexrelease⟩   \edef\reserved@a{\noexpand\@missingfileerror
⟨latexrelease⟩     {\filename@area\filename@base}%
⟨latexrelease⟩     {\ifx\filename@ext\relax tex\else\filename@ext\fi}}%
⟨latexrelease⟩   \reserved@a}}
⟨latexrelease⟩\EndIncludeInRelease
⟨*2ekernel⟩
```

\@input Define \@input in terms of \IfIfileExists. 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.

```
\def\@input#1{%
  \IfFileExists{#1}{\@@input\@filef@und}{\typeout{No file #1.}}}
```

\@input@ Version of \@input that does add the file to \@filelist.

```
\def\@input@#1{\InputIfFileExists{#1}{}{\typeout{No file #1.}}}
```

\@missingfileerror This 'error' command avoids TeX's primitive missing file loop.

Missing file error. Prompt for a new filename, offering a default extension.

```
\gdef\@missingfileerror#1#2{%
     \typeout{^^J! LaTeX Error: File `#1.#2' not found.^^J^^J%
      Type X to quit or <RETURN> to proceed,^^J%
      or enter new name. (Default extension: #2)^^J}%
     \message{Enter file name: }%
      {\endlinechar\m@ne
       \global\read\m@ne to\@gtempa}%
    \ifx\@gtempa\@empty
    \else
      \def\reserved@a{x}\ifx\reserved@a\@gtempa\batchmode\@@end\fi
      \def\reserved@a{X}\ifx\reserved@a\@gtempa\batchmode\@@end\fi
      \filename@parse\@gtempa
      \edef\filename@ext{%
        \ifx\filename@ext\relax#2\else\filename@ext\fi}%
     \edef\reserved@a{%
       \noexpand\InputIfFileExists
         {\filename@area\filename@base.\filename@ext}%
         {}%
         {\noexpand\@missingfileerror
            {\filename@area\filename@base}{\filename@ext}}}%
      \reserved@a
    \fi}
```

\@obsoletefile For compatibility with LaTeX 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.

```
\def\@obsoletefile#1#2{%
   \@latex@warning@no@line{inputting ‘#1’ instead of obsolete ‘#2’}}
\@onlypreamble\@obsoletefile
```

19.2 Listing files

\@filelist A list of files input so far. The initial value of \@gobble eats the comma before the first file name.

```
\let\@filelist\@gobble
```

\@addtofilelist 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.

```
%\def\@addtofilelist#1{\xdef\@filelist{\@filelist,#1}}
```

\listfiles A preamble command to cause \end{document} to list files input from the main file.

```
\def\listfiles{%
  \let\listfiles\relax
  \def\@listfiles##1##2##3##4##5##6##7##8##9\@@{%
     \def\reserved@d{\\}%
     \@tfor\reserved@c:=##1##2##3##4##5##6##7##8\do{%
       \ifx\reserved@c\reserved@d
         \edef\filename@area{ \filename@area}%
       \fi}}%

  \def\@dofilelist{%
     \typeout{^^J *File List*}%
     \@for\@currname:=\@filelist\do{%
       \filename@parse\@currname
       \edef\reserved@a{%
          \filename@base.%
          \ifx\filename@ext\relax tex\else\filename@ext\fi}%
       \expandafter\let\expandafter\reserved@b
                             \csname ver@\reserved@a\endcsname
       \expandafter\expandafter\expandafter\@listfiles\expandafter
            \filename@area\filename@base\\\\\\\\\\\\\\\\\\@@
       \typeout{%
         \filename@area\reserved@a
         \ifx\reserved@b\relax\else\@spaces\reserved@b\fi}}%
     \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}
```

```
\@onlypreamble\listfiles
```

`\@dofilelist`

```
\let\@dofilelist\relax
```

```
⟨/2ekernel⟩
```

File l
ltoutenc.dtx

20 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 {\frac14}}
```

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`.

20.1 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 flacky 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` LaTeX 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 LaTeX 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}
```

20.2 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.

20.3 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 `t1enc.def` for the Cork encoding.
`TS1`	generates `ts1enc.def` for the Text Companion encoding.
`TS1sty`	generates `textcomp.sty`, package that sets up use of the Text Companion encoding.
`OT1`	generates `ot1enc.def` for Knuth's CM encoding.
`OMS`	generates `omsenc.def` for Knuth's math symbol encoding.
`OML`	generates `omlenc.def` for Knuth's math letters encoding.
`OT4`	generates `ot4enc.def` 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 `tuenc.def` for Unicode font encoding.
`package`	generates `fontenc.sty` for selecting encodings.
`2ekernel`	for the kernel commands.

20.4 Definitions for the kernel

20.4.1 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.

```
⟨*2ekernel⟩
\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`.

```
\def\DeclareTextCommand{%
   \@dec@text@cmd\newcommand}
\def\ProvideTextCommand{%
   \@dec@text@cmd\providecommand}
\def\@dec@text@cmd#1#2#3{%
   \expandafter\def\expandafter#2%
      \expandafter{%
         \csname#3-cmd\expandafter\endcsname
         \expandafter#2%
         \csname#3\string#2\endcsname
      }%
   \let\@ifdefinable\@rc@ifdefinable
   \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.

```
\def\chardef@text@cmd{%
   \let\@ifdefinable\@@ifdefinable
   \chardef
  }
\def\DeclareTextSymbol#1#2#3{%
   \@dec@text@cmd\chardef@text@cmd#1{#2}#3\relax
  }
```

The declarations are only available before `\begin{document}`.

```
\@onlypreamble\DeclareTextCommand
\@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.

```
\def\@current@cmd#1{%
   \ifx\protect\@typeset@protect
      \@inmathwarn#1%
   \else
      \noexpand#1\expandafter\@gobble
   \fi}
```

```
\def\@changed@cmd#1#2{%
   \ifx\protect\@typeset@protect
      \@inmathwarn#1%
      \expandafter\ifx\csname\cf@encoding\string#1\endcsname\relax
         \expandafter\ifx\csname ?\string#1\endcsname\relax
            \expandafter\def\csname ?\string#1\endcsname{%
               \TextSymbolUnavailable#1%
            }%
         \fi
         \global\expandafter\let
               \csname\cf@encoding \string#1\expandafter\endcsname
               \csname ?\string#1\endcsname
      \fi
      \csname\cf@encoding\string#1%
         \expandafter\endcsname
   \else
      \noexpand#1%
   \fi}
```

```
\gdef\TextSymbolUnavailable#1{%
```

```
   \@latex@error{%
      Command \protect#1 unavailable in encoding \cf@encoding%
   }\@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).

```
\def\@inmathwarn#1{%
   \ifmmode
      \@latex@warning{Command \protect#1 invalid in math mode}%
   \fi}
```

`\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.

```
\def\DeclareTextCommandDefault#1{%
   \DeclareTextCommand#1?}
```

```
\def\ProvideTextCommandDefault#1{%
   \ProvideTextCommand#1?}
```

```
\@onlypreamble\DeclareTextCommandDefault
%\@onlypreamble\ProvideTextCommandDefault
```

They require `\?-cmd` to be initialized as `\@changed@cmd`.

```
\expandafter\let\csname?-cmd\endcsname\@changed@cmd
```

`\DeclareTextAccent`

This is just a disguise for defining a TeX `\accent` command.

```
\def\DeclareTextAccent#1#2#3{%
   \DeclareTextCommand#1{#2}{\add@accent{#3}}}
```

```
\@onlypreamble\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 reimplementation and its deficiencies, see pr/3160.

```
\def\add@accent#1#2{\hmode@bgroup
```

Turn off the group in `\UseTextSymbol` in case this is used inside the argument of `\add@accent`.

```
  \let\hmode@start@before@group\@firstofone
  \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.

```
      \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 temporaily. We also ignore `\ignorespaces` which has the same issue and may show up as part of `\normalfont` if that is used.

```
  \let\maybe@load@fontshape\relax
  \let\ignorespaces\relax
  \accent#1 #2\egroup\spacefactor\accent@spacefactor}
```

Default definition for `\accent@spacefactor` prevents a horrible death of the above macro inside an unprotected `\edef`.

```
\let\accent@spacefactor\relax
```

`\hmode@bgroup`

```
\def\hmode@bgroup{\leavevmode\bgroup}
```

`\DeclareTextCompositeCommand`
`\DeclareTextComposite`
`\@text@composite`
`\@text@composite@x`
`\@strip@args`

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:

```
#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`.

```
⟨/2ekernel⟩
⟨latexrelease⟩\IncludeInRelease{2017/04/15}{\DeclareTextCompositeCommand}
⟨latexrelease⟩                              {test for undeclared accent}%
⟨*2ekernel | latexrelease⟩
\def\DeclareTextCompositeCommand#1#2#3#4{%
  \expandafter\let\expandafter\reserved@a\csname#2\string#1\endcsname
  \ifx\reserved@a\relax
   \DeclareTextCommand#1{#2}{%
     \@latex@error{\string#1 undeclared in encoding #2}\@eha}%
   \@latex@info{Composite with undeclared \string#1 in encoding #2}%
   \expandafter\let\expandafter\reserved@a\csname#2\string#1\endcsname
  \fi
  \expandafter\expandafter\expandafter\ifx
  \expandafter\@car\reserved@a\relax\relax\@nil \@text@composite \else
      \edef\reserved@b##1{%
         \def\expandafter\noexpand
            \csname#2\string#1\endcsname####1{%
            \noexpand\@text@composite
```

```
               \expandafter\noexpand\csname#2\string#1\endcsname
               ####1\noexpand\@empty\noexpand\@text@composite
               {##1}}}%
      \expandafter\reserved@b\expandafter{\reserved@a{##1}}%
   \fi
   \expandafter\def\csname\expandafter\string\csname
      #2\endcsname\string#1-\string#3\@empty\endcsname{#4}%
  }
⟨/2ekernel | latexrelease⟩
⟨latexrelease⟩\EndIncludeInRelease
⟨latexrelease⟩\IncludeInRelease{0000/00/00}{\DeclareTextCompositeCommand}
⟨latexrelease⟩                            {test for undeclared accent}%
⟨latexrelease⟩\def\DeclareTextCompositeCommand#1#2#3#4{%
⟨latexrelease⟩  \expandafter\let\expandafter\reserved@a
⟨latexrelease⟩                      \csname#2\string#1\endcsname
⟨latexrelease⟩  \expandafter\expandafter\expandafter\ifx
⟨latexrelease⟩  \expandafter\@car\reserved@a\relax\relax\@nil
⟨latexrelease⟩                               \@text@composite \else
⟨latexrelease⟩      \edef\reserved@b##1{%
⟨latexrelease⟩         \def\expandafter\noexpand
⟨latexrelease⟩           \csname#2\string#1\endcsname####1{%
⟨latexrelease⟩           \noexpand\@text@composite
⟨latexrelease⟩             \expandafter\noexpand\csname#2\string#1\endcsname
⟨latexrelease⟩             ####1\noexpand\@empty\noexpand\@text@composite
⟨latexrelease⟩             {##1}}}%
⟨latexrelease⟩      \expandafter\reserved@b\expandafter{\reserved@a{##1}}%
⟨latexrelease⟩   \fi
⟨latexrelease⟩   \expandafter\def\csname\expandafter\string\csname
⟨latexrelease⟩      #2\endcsname\string#1-\string#3\@empty\endcsname{#4}}
⟨latexrelease⟩\EndIncludeInRelease
⟨*2ekernel⟩
```

```
\@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 áa. 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.

```
\def\@text@composite#1#2#3\@text@composite{%
   \expandafter\@text@composite@x
      \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 form the input stream. This has now being 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.

```
\def\@text@composite@x#1{%
   \ifx#1\relax
      \expandafter\@secondoftwo
   \else
      \expandafter\@firstoftwo
   \fi
   #1}
```

The command \DeclareTextComposite uses \DeclareTextCompositeCommand to declare a command which expands out to a single glyph.

```
\catcode\z@=11\relax

\def\DeclareTextComposite#1#2#3#4{%
   \def\reserved@a{\DeclareTextCompositeCommand#1{#2}{#3}}%
   \bgroup
      \lccode\z@#4%
      \lowercase{%
   \egroup
      \reserved@a ^^@}}

\catcode\z@=15\relax

\@onlypreamble\DeclareTextComposite
```

```
⟨/2ekernel⟩
⟨*2ekernel | latexrelease⟩
⟨latexrelease⟩\IncludeInRelease{2019/10/01}%
⟨latexrelease⟩                     {\UseTextAccent}{Make commands robust}%
```

\UseTextAccent
\UseTextSymbol
\@use@text@encoding

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.

For a detailed discussion of this reimplementation and its deficiencies, see pr/3160.

```
\DeclareRobustCommand\UseTextAccent[3]{%
  \hmode@start@before@group
  {%
```

Turn off the group in \UseTextSymbol in case this is used inside the arguments of \UseTextAccent.

```
    \let\hmode@start@before@group\@firstofone
    \let\@curr@enc\cf@encoding
    \@use@text@encoding{#1}%
    #2{\@use@text@encoding\@curr@enc#3}%
   }}
```

```
\DeclareRobustCommand\UseTextSymbol[2]{%
       \hmode@start@before@group
       {%
          \def\@wrong@font@char{\MessageBreak
             for \noexpand\symbol`\string#2'}%
          \@use@text@encoding{#1}%
          #2%
       }%
    }
⟨/2ekernel | latexrelease⟩
⟨latexrelease⟩\EndIncludeInRelease
⟨latexrelease⟩\IncludeInRelease{0000/00/00}%
⟨latexrelease⟩                 {\UseTextAccent}{Make commands robust}%
⟨latexrelease⟩
⟨latexrelease⟩\kernel@make@fragile\UseTextAccent
⟨latexrelease⟩\kernel@make@fragile\UseTextSymbol
⟨latexrelease⟩
⟨latexrelease⟩\EndIncludeInRelease
⟨*2ekernel⟩
```

Switch to a different text encoding without any grouping for use in \UseTextAccent or \UseTextSymbol (and for \oldstylenums).

```
\def\@use@text@encoding#1{%
   \edef\f@encoding{#1}%
   \xdef\font@name{%
      \csname\curr@fontshape/\f@size\endcsname}%
   \pickup@font
   \font@name
   \@@enc@update}
```

\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.

```
\let\hmode@start@before@group\leavevmode
```

\DeclareTextSymbolDefault
\DeclareTextAccentDefault

Some syntactic sugar. Again, these should probably be optimized for speed.

```
\def\DeclareTextSymbolDefault#1#2{%
   \DeclareTextCommandDefault#1{\UseTextSymbol{#2}#1}}
```

```
\def\DeclareTextAccentDefault#1#2{%
   \DeclareTextCommandDefault#1{\UseTextAccent{#2}#1}}
```

```
\@onlypreamble\DeclareTextSymbolDefault
\@onlypreamble\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.

```
\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).

```
  \expandafter\ifx\csname#2\string#1\endcsname\relax
  \else
```

Else: throw away that declaration.

```
     \global\expandafter\let\csname#2\string#1\endcsname
            \@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:

```
     \expandafter\expandafter\expandafter
     \ifx\expandafter\@thirdofthree#1\@undefined
       \expandafter\gdef\expandafter#1\expandafter
          {\csname ?-cmd\expandafter\endcsname\expandafter
           #1\csname?\string#1\endcsname}%
     \fi
 \fi
}
```

```
\@onlypreamble\UndeclareTextCommand
```

20.4.2 Hyphenation

\patterns
\@@patterns
\hyphenation
\@@hyphenation

We redefine \patterns and \hyphenation to allow the use of commands declared with \DeclareText* to be used inside them.

```
%\let\@@patterns\patterns
%\let\@@hyphenation\hyphenation
%\def\patterns{%
%   \bgroup
%      \let\protect\@empty
%      \let\@typeset@protect\@empty
%      \let\@changed@x\@changed@x@mouth
%   \afterassignment\egroup
%   \@@patterns
%}
%\def\hyphenation{%
%   \bgroup
%      \let\protect\@empty
%      \let\@typeset@protect\@empty
%      \let\@changed@x\@changed@x@mouth
%   \afterassignment\egroup
%   \@@hyphenation
%}
```

20.4.3 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.

```
\def\@tabacckludge#1{\expandafter\@changed@cmd
                                   \csname\string#1\endcsname\relax}
\let\a=\@tabacckludge
```

20.4.4 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 TEX 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:

```
\DeclareTextAccentDefault{\"}{OT1}
\DeclareTextAccentDefault{\'}{OT1}
\DeclareTextAccentDefault{\.}{OT1}
\DeclareTextAccentDefault{\=}{OT1}
\DeclareTextAccentDefault{\H}{OT1}
\DeclareTextAccentDefault{\^}{OT1}
\DeclareTextAccentDefault{\`}{OT1}
\DeclareTextAccentDefault{\b}{OT1}
\DeclareTextAccentDefault{\c}{OT1}
\DeclareTextAccentDefault{\d}{OT1}
\DeclareTextAccentDefault{\r}{OT1}
\DeclareTextAccentDefault{\u}{OT1}
\DeclareTextAccentDefault{\v}{OT1}
```

```
\DeclareTextAccentDefault{\~}{OT1}
```

Some symbols from OT1:

```
%\DeclareTextSymbolDefault{\AA}{OT1}
\DeclareTextSymbolDefault{\AE}{OT1}
\DeclareTextSymbolDefault{\L}{OT1}
\DeclareTextSymbolDefault{\OE}{OT1}
\DeclareTextSymbolDefault{\O}{OT1}
%\DeclareTextSymbolDefault{\aa}{OT1}
\DeclareTextSymbolDefault{\ae}{OT1}
\DeclareTextSymbolDefault{\i}{OT1}
\DeclareTextSymbolDefault{\j}{OT1}
```

```
\DeclareTextSymbolDefault{\ij}{OT1}
\DeclareTextSymbolDefault{\IJ}{OT1}
```

```
\DeclareTextSymbolDefault{\l}{OT1}
\DeclareTextSymbolDefault{\oe}{OT1}
\DeclareTextSymbolDefault{\o}{OT1}
\DeclareTextSymbolDefault{\ss}{OT1}
\DeclareTextSymbolDefault{\textdollar}{OT1}
\DeclareTextSymbolDefault{\textemdash}{OT1}
\DeclareTextSymbolDefault{\textendash}{OT1}
\DeclareTextSymbolDefault{\textexclamdown}{OT1}
%\DeclareTextSymbolDefault{\texthyphenchar}{OT1}
%\DeclareTextSymbolDefault{\texthyphen}{OT1}
\DeclareTextSymbolDefault{\textquestiondown}{OT1}
\DeclareTextSymbolDefault{\textquotedblleft}{OT1}
\DeclareTextSymbolDefault{\textquotedblright}{OT1}
\DeclareTextSymbolDefault{\textquoteleft}{OT1}
\DeclareTextSymbolDefault{\textquoteright}{OT1}
\DeclareTextSymbolDefault{\textsterling}{OT1}
```

Some symbols from OMS:

```
\DeclareTextSymbolDefault{\textasteriskcentered}{OMS}
\DeclareTextSymbolDefault{\textbackslash}{OMS}
\DeclareTextSymbolDefault{\textbar}{OMS}
\DeclareTextSymbolDefault{\textbardbl}{OMS}
\DeclareTextSymbolDefault{\textbraceleft}{OMS}
\DeclareTextSymbolDefault{\textbraceright}{OMS}
\DeclareTextSymbolDefault{\textbullet}{OMS}
\DeclareTextSymbolDefault{\textdaggerdbl}{OMS}
\DeclareTextSymbolDefault{\textdagger}{OMS}
\DeclareTextSymbolDefault{\textparagraph}{OMS}
\DeclareTextSymbolDefault{\textperiodcentered}{OMS}
\DeclareTextSymbolDefault{\textsection}{OMS}
\DeclareTextAccentDefault{\textcircled}{OMS}
```

Some symbols from OML:

```
\DeclareTextSymbolDefault{\textless}{OML}
\DeclareTextSymbolDefault{\textgreater}{OML}
\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.

```
\DeclareTextCommandDefault{\textcopyright}{\textcircled{c}}
% \expandafter\def\expandafter
%                 \copyright\expandafter{\expandafter{\copyright}}

\DeclareTextCommandDefault{\textasciicircum}{\^{}}
\DeclareTextCommandDefault{\textasciitilde}{\~{}}
\DeclareTextCommandDefault{\textunderscore}{%
  \leavevmode \kern.06em\vbox{\hrule\@width.3em}}
```

There is no good reason anymore to fake `\textcompwordmark`.

```
%\DeclareTextCommandDefault{\textcompwordmark}{\leavevmode\kern\z@}
\DeclareTextSymbolDefault{\textcompwordmark}{T1}

\DeclareTextCommandDefault{\textvisiblespace}{%
   \mbox{\kern.06em\vrule \@height.3ex}%
   \vbox{\hrule \@width.3em}%
   \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).

```
\DeclareTextCommandDefault{\textellipsis}{%
   .\kern\fontdimen3\font
   .\kern\fontdimen3\font
   .\kern\fontdimen3\font}

%\DeclareTextCommandDefault{\textregistered}{\textcircled{\scshape r}}
\DeclareTextCommandDefault{\textregistered}{\textcircled{%
     \check@mathfonts\fontsize\sf@size\z@\math@fontsfalse\selectfont R}}
\DeclareTextCommandDefault{\texttrademark}{\textsuperscript{TM}}
\DeclareTextCommandDefault{\SS}{SS}

\DeclareTextCommandDefault{\textordfeminine}{\textsuperscript{a}}
\DeclareTextCommandDefault{\textordmasculine}{\textsuperscript{o}}
```

20.4.5 Math material

Some commands can be used in both text and math mode:

```
\DeclareRobustCommand{\$}{\ifmmode\mathdollar\else\textdollar\fi}
```

We use `\protected` not `\DeclareRobustCommand` so that `\bigl\{` etc. works inside `\protected@edef`.

```
\protected\def\{{\ifmmode\lbrace\else\textbraceleft\fi}
\protected\def\}{\ifmmode\rbrace\else\textbraceright\fi}

\DeclareRobustCommand{\P}{\ifmmode\mathparagraph\else\textparagraph\fi}
\DeclareRobustCommand{\S}{\ifmmode\mathsection\else\textsection\fi}
\DeclareRobustCommand{\dag}{\ifmmode{\dagger}\else\textdagger\fi}
\DeclareRobustCommand{\ddag}{\ifmmode{\ddagger}\else\textdaggerdbl\fi}
```

For historical reasons `\copyright` needs `{}` around the definition in maths.

```
\DeclareRobustCommand{\_}{%
   \ifmmode\nfss@text{\textunderscore}\else\textunderscore\fi}
\DeclareRobustCommand{\copyright}{%
   \ifmmode{\nfss@text{\textcopyright}}\else\textcopyright\fi}
\DeclareRobustCommand{\pounds}{%
   \ifmmode\mathsterling\else\textsterling\fi}
```

```
\DeclareRobustCommand{\dots}{%
   \ifmmode\mathellipsis\else\textellipsis\fi}
```

```
\let\ldots\dots
```

Default definition of the commabelow accent.

```
⟨/2ekernel⟩
⟨latexrelease⟩\IncludeInRelease{2015/10/01}{\textcommabelow}{comma accent}%
⟨*2ekernel | latexrelease⟩
\DeclareTextCommandDefault\textcommabelow[1]
  {\hmode@bgroup\ooalign{\null#1\crcr\hidewidth\raise-.31ex
   \hbox{\check@mathfonts\fontsize\ssf@size\z@
   \math@fontsfalse\selectfont,}\hidewidth}\egroup}
⟨latexrelease⟩\EndIncludeInRelease
⟨/2ekernel | latexrelease⟩
⟨latexrelease⟩\IncludeInRelease{0000/00/00}{\textcommabelow}{comma accent}%
⟨latexrelease⟩\let\textcommabelow\@undefined
⟨latexrelease⟩\expandafter
⟨latexrelease⟩  \let\csname\string\T1\string\c-G\endcsname\@undefined
⟨latexrelease⟩\expandafter
⟨latexrelease⟩  \let\csname\string\T1\string\c-K\endcsname\@undefined
⟨latexrelease⟩\expandafter
⟨latexrelease⟩  \let\csname\string\T1\string\c-k\endcsname\@undefined
⟨latexrelease⟩\expandafter
⟨latexrelease⟩  \let\csname\string\T1\string\c-L\endcsname\@undefined
⟨latexrelease⟩\expandafter
⟨latexrelease⟩  \let\csname\string\T1\string\c-l\endcsname\@undefined
⟨latexrelease⟩\expandafter
⟨latexrelease⟩  \let\csname\string\T1\string\c-N\endcsname\@undefined
⟨latexrelease⟩\expandafter
⟨latexrelease⟩  \let\csname\string\T1\string\c-n\endcsname\@undefined
⟨latexrelease⟩\expandafter
⟨latexrelease⟩  \let\csname\string\T1\string\c-R\endcsname\@undefined
⟨latexrelease⟩\expandafter
⟨latexrelease⟩  \let\csname\string\T1\string\c-r\endcsname\@undefined
⟨latexrelease⟩\EndIncludeInRelease
```

Default definition of the commaabove accent(E.G.).

```
⟨latexrelease⟩\IncludeInRelease{2016/02/01}{\textcommaabove}{comma above}%
⟨*2ekernel | latexrelease⟩
\DeclareTextCommandDefault\textcommaabove[1]{%
  \hmode@bgroup
  \ooalign{%
    \hidewidth
    \raise.7ex\hbox{%
      \check@mathfonts\fontsize\ssf@size\z@\math@fontsfalse\selectfont`%
    }%
   \hidewidth\crcr
   \null#1\crcr
  }%
  \egroup
}
⟨latexrelease⟩\EndIncludeInRelease
⟨/2ekernel | latexrelease⟩
⟨latexrelease⟩\IncludeInRelease{0000/00/00}{\textcommaabove}{comma above}%
```

```
⟨latexrelease⟩\let\textcommaabove\@undefined
⟨latexrelease⟩\expandafter
⟨latexrelease⟩  \let\csname\string\OT1\string\c-g\endcsname\@undefined
⟨latexrelease⟩\expandafter
⟨latexrelease⟩  \let\csname\string\T1\string\c-g\endcsname\@undefined
⟨latexrelease⟩\EndIncludeInRelease
```

20.5 Definitions for the OT1 encoding

The definitions for the 'TEX text' (OT1) encoding.

Declare the encoding.

```
⟨*OT1⟩
\DeclareFontEncoding{OT1}{}{}
```

Declare the accents.

```
\DeclareTextAccent{\"}{OT1}{127}
\DeclareTextAccent{\'}{OT1}{19}
\DeclareTextAccent{\.}{OT1}{95}
\DeclareTextAccent{\=}{OT1}{22}
\DeclareTextAccent{\^}{OT1}{94}
\DeclareTextAccent{\`}{OT1}{18}
\DeclareTextAccent{\~}{OT1}{126}
\DeclareTextAccent{\H}{OT1}{125}
\DeclareTextAccent{\u}{OT1}{21}
\DeclareTextAccent{\v}{OT1}{20}
\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.

```
\DeclareTextCommand{\b}{OT1}[1]
  {\hmode@bgroup\o@lign{\relax#1\crcr\hidewidth\ltx@sh@ft{-3ex}%
     \vbox to.2ex{\hbox{\char22}\vss}\hidewidth}\egroup}
\DeclareTextCommand{\c}{OT1}[1]
  {\leavevmode\setbox\z@\hbox{#1}\ifdim\ht\z@=1ex\accent24 #1%
    \else{\ooalign{\unhbox\z@\crcr\hidewidth\char24\hidewidth}}\fi}
\DeclareTextCommand{\d}{OT1}[1]
  {\hmode@bgroup
    \o@lign{\relax#1\crcr\hidewidth\ltx@sh@ft{-1ex}.\hidewidth}\egroup}
```

Declare the text symbols.

```
\DeclareTextSymbol{\AE}{OT1}{29}
\DeclareTextSymbol{\OE}{OT1}{30}
\DeclareTextSymbol{\O}{OT1}{31}
\DeclareTextSymbol{\ae}{OT1}{26}
\DeclareTextSymbol{\i}{OT1}{16}
\DeclareTextSymbol{\j}{OT1}{17}
\DeclareTextSymbol{\oe}{OT1}{27}
\DeclareTextSymbol{\o}{OT1}{28}
\DeclareTextSymbol{\ss}{OT1}{25}
\DeclareTextSymbol{\textemdash}{OT1}{124}
\DeclareTextSymbol{\textendash}{OT1}{123}
```

Using the ligatures helps with OT1 fonts that have `\textexclamdown` and `\textquestiondown` in unusual positions.

```
%\DeclareTextSymbol{\textexclamdown}{OT1}{60}
%\DeclareTextSymbol{\textquestiondown}{OT1}{62}
\DeclareTextCommand{\textexclamdown}{OT1}{!`}
\DeclareTextCommand{\textquestiondown}{OT1}{?`}
%\DeclareTextSymbol{\texthyphenchar}{OT1}{`\-}
%\DeclareTextSymbol{\texthyphen}{OT1}{`\-}
\DeclareTextSymbol{\textquotedblleft}{OT1}{92}
\DeclareTextSymbol{\textquotedblright}{OT1}{`\"}
\DeclareTextSymbol{\textquoteleft}{OT1}{`\`}
\DeclareTextSymbol{\textquoteright}{OT1}{`\'}
```

Some symbols which are faked from others:

```
% \DeclareTextCommand{\aa}{OT1}
%    {{\accent23a}}
\DeclareTextCommand{\L}{OT1}
   {\leavevmode\setbox\z@\hbox{L}\hb@xt@\wd\z@{\hss\@xxxii L}}
\DeclareTextCommand{\l}{OT1}
   {\hmode@bgroup\@xxxii l\egroup}
% \DeclareTextCommand{\AA}{OT1}
%    {\leavevmode\setbox\z@\hbox{h}\dimen@\ht\z@\advance\dimen@-1ex%
%     \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.

```
\DeclareTextCompositeCommand{\r}{OT1}{A}
   {\leavevmode\setbox\z@\hbox{!}\dimen@\ht\z@\advance\dimen@-1ex%
    \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.

```
\DeclareTextCommand{\ij}{OT1}{%
  \nobreak\hskip\z@skip i\kern-0.02em j\nobreak\hskip\z@skip}
\DeclareTextCommand{\IJ}{OT1}{%
  \nobreak\hskip\z@skip I\kern-0.02em J\nobreak\hskip\z@skip}
```

In the OT1 encoding, £ and $ share a slot.

```
\DeclareTextCommand{\textdollar}{OT1}{\hmode@bgroup
   \ifdim \fontdimen\@ne\font >\z@
      \slshape
   \else
      \upshape
   \fi
   \char`\$\egroup}
```

```
\DeclareTextCommand{\textsterling}{OT1}{\hmode@bgroup
   \ifdim \fontdimen\@ne\font >\z@
      \itshape
   \else
      \fontshape{ui}\selectfont
   \fi
   \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 LaTeX 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.

```
\DeclareTextComposite{\.}{OT1}{i}{`\i}
\DeclareTextComposite{\.}{OT1}{\i}{`\i}
\DeclareTextCompositeCommand{\`}{OT1}{i}{\@tabacckludge`\i}
\DeclareTextCompositeCommand{\'}{OT1}{i}{\@tabacckludge'\i}
\DeclareTextCompositeCommand{\^}{OT1}{i}{\^\i}
\DeclareTextCompositeCommand{\"}{OT1}{i}{\"\i}
```

T1 encoding is given more extensive set of overloads for `\c` But here we just adjust `\c{g}`.

```
\ifx\textcommaabove\@undefined\else
\DeclareTextCompositeCommand{\c}{OT1}{g}{\textcommaabove{g}}
\fi
```

```
⟨/OT1⟩
```

20.6 Definitions for the T1 encoding

The definitions for the 'Extended TeX text' (T1) encoding.

Declare the encoding.

```
⟨*T1⟩
\DeclareFontEncoding{T1}{}{}
```

Declare the accents.

```
\DeclareTextAccent{\`}{T1}{0}
\DeclareTextAccent{\'}{T1}{1}
\DeclareTextAccent{\^}{T1}{2}
\DeclareTextAccent{\~}{T1}{3}
\DeclareTextAccent{\"}{T1}{4}
\DeclareTextAccent{\H}{T1}{5}
\DeclareTextAccent{\r}{T1}{6}
\DeclareTextAccent{\v}{T1}{7}
\DeclareTextAccent{\u}{T1}{8}
\DeclareTextAccent{\=}{T1}{9}
\DeclareTextAccent{\.}{T1}{10}
```

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.

```
\DeclareTextCommand{\b}{T1}[1]
  {\hmode@bgroup\o@lign{\relax#1\crcr\hidewidth\ltx@sh@ft{-3ex}%
    \vbox to.2ex{\hbox{\char9}\vss}\hidewidth}\egroup}
\DeclareTextCommand{\c}{T1}[1]
  {\leavevmode\setbox\z@\hbox{#1}\ifdim\ht\z@=1ex\accent11 #1%
    \else{\ooalign{\unhbox\z@\crcr
      \hidewidth\char11\hidewidth}}\fi}
\DeclareTextCommand{\d}{T1}[1]
  {\hmode@bgroup
   \o@lign{\relax#1\crcr\hidewidth\ltx@sh@ft{-1ex}.\hidewidth}\egroup}
\DeclareTextCommand{\k}{T1}[1]
  {\hmode@bgroup\ooalign{\null#1\crcr\hidewidth\char12}\egroup}
\DeclareTextCommand{\textogonekcentered}{T1}[1]
  {\hmode@bgroup\ooalign{%
               \null#1\crcr\hidewidth\char12\hidewidth}\egroup}
```

Some symbols are constructed.

Slot 24 contains a small circle intended for construction of these two glyphs.

```
\DeclareTextCommand{\textperthousand}{T1}
   {\%\char 24 }          % space or ‘relax as delimiter?
\DeclareTextCommand{\textpertenthousand}{T1}
   {\%\char 24\char 24 }  % space or ‘relax as delimiter?
```

For Maltese, `\Hwithstroke` and `\hwithstroke` are needed.

```
\DeclareTextCommand{\Hwithstroke}{T1}
   {%
    \hmode@bgroup
    \vphantom{H}%
    \sbox\z@{H}%
    \ooalign{%
      H\cr
      \hidewidth
      \vrule
        height \dimexpr 0.7\ht\z@+0.1ex\relax
        depth  -0.7\ht\z@
        width  0.8\wd\z@
      \hidewidth\cr
    }%
    \egroup
   }
\DeclareTextCommand{\hwithstroke}{T1}
   {%
    \hmode@bgroup
    \vphantom{h}%
    \sbox\z@{h}%
    \ooalign{%
      h\cr
      \kern0.075\wd\z@
      \vrule
        height \dimexpr 0.7\ht\z@+0.1ex\relax
        depth  -0.7\ht\z@
        width  0.4\wd\z@
      \hidewidth\cr
    }%
    \egroup
   }
```

Declare the text symbols.

```
%\DeclareTextSymbol{\AA}{T1}{197}
\DeclareTextSymbol{\AE}{T1}{198}
\DeclareTextSymbol{\DH}{T1}{208}
\DeclareTextSymbol{\DJ}{T1}{208}
\DeclareTextSymbol{\L}{T1}{138}
\DeclareTextSymbol{\NG}{T1}{141}
\DeclareTextSymbol{\OE}{T1}{215}
\DeclareTextSymbol{\O}{T1}{216}
\DeclareTextSymbol{\SS}{T1}{223}
\DeclareTextSymbol{\TH}{T1}{222}
%\DeclareTextSymbol{\aa}{T1}{229}
\DeclareTextSymbol{\ae}{T1}{230}
```

```
\DeclareTextSymbol{\dh}{T1}{240}
\DeclareTextSymbol{\dj}{T1}{158}
```

```
\DeclareTextSymbol{\guillemetleft}{T1}{19}
\DeclareTextSymbol{\guillemetright}{T1}{20}
% old Adobe names
\DeclareTextSymbol{\guillemotleft}{T1}{19}
\DeclareTextSymbol{\guillemotright}{T1}{20}
```

```
\DeclareTextSymbol{\guilsinglleft}{T1}{14}
\DeclareTextSymbol{\guilsinglright}{T1}{15}
\DeclareTextSymbol{\i}{T1}{25}
\DeclareTextSymbol{\j}{T1}{26}
\DeclareTextSymbol{\ij}{T1}{188}
\DeclareTextSymbol{\IJ}{T1}{156}
\DeclareTextSymbol{\l}{T1}{170}
\DeclareTextSymbol{\ng}{T1}{173}
\DeclareTextSymbol{\oe}{T1}{247}
\DeclareTextSymbol{\o}{T1}{248}
\DeclareTextSymbol{\quotedblbase}{T1}{18}
\DeclareTextSymbol{\quotesinglbase}{T1}{13}
\DeclareTextSymbol{\ss}{T1}{255}
\DeclareTextSymbol{\textasciicircum}{T1}{`\^}
\DeclareTextSymbol{\textasciitilde}{T1}{`\~}
\DeclareTextSymbol{\textbackslash}{T1}{`\\}
\DeclareTextSymbol{\textbar}{T1}{`\|}
\DeclareTextSymbol{\textbraceleft}{T1}{`\{}
\DeclareTextSymbol{\textbraceright}{T1}{`\}}
\DeclareTextSymbol{\textcompwordmark}{T1}{23}
\DeclareTextSymbol{\textdollar}{T1}{`\$}
\DeclareTextSymbol{\textemdash}{T1}{22}
\DeclareTextSymbol{\textendash}{T1}{21}
\DeclareTextSymbol{\textexclamdown}{T1}{189}
\DeclareTextSymbol{\textgreater}{T1}{`\>}
%\DeclareTextSymbol{\texthyphenchar}{T1}{127}
%\DeclareTextSymbol{\texthyphen}{T1}{`\-}
\DeclareTextSymbol{\textless}{T1}{`\<}
\DeclareTextSymbol{\textquestiondown}{T1}{190}
\DeclareTextSymbol{\textquotedblleft}{T1}{16}
\DeclareTextSymbol{\textquotedblright}{T1}{17}
\DeclareTextSymbol{\textquotedbl}{T1}{`\"}
\DeclareTextSymbol{\textquoteleft}{T1}{`\`}
\DeclareTextSymbol{\textquoteright}{T1}{`\'}
\DeclareTextSymbol{\textsection}{T1}{159}
\DeclareTextSymbol{\textsterling}{T1}{191}
\DeclareTextSymbol{\textunderscore}{T1}{95}
\DeclareTextSymbol{\textvisiblespace}{T1}{32}
\DeclareTextSymbol{\th}{T1}{254}
```

Declare the composites.

```
\DeclareTextComposite{\.}{T1}{i}{`\i}
\DeclareTextComposite{\.}{T1}{\i}{`\i}
```

”80 = 128

```
\DeclareTextComposite{\u}{T1}{A}{128}
\DeclareTextComposite{\k}{T1}{A}{129}
```

```
\DeclareTextComposite{\'}{T1}{C}{130}
\DeclareTextComposite{\v}{T1}{C}{131}
\DeclareTextComposite{\v}{T1}{D}{132}
\DeclareTextComposite{\v}{T1}{E}{133}
\DeclareTextComposite{\k}{T1}{E}{134}
\DeclareTextComposite{\u}{T1}{G}{135}
```

”88 = 136

```
\DeclareTextComposite{\'}{T1}{L}{136}
\DeclareTextComposite{\v}{T1}{L}{137}
\DeclareTextComposite{\'}{T1}{N}{139}
\DeclareTextComposite{\v}{T1}{N}{140}
\DeclareTextComposite{\H}{T1}{O}{142}
\DeclareTextComposite{\'}{T1}{R}{143}
```

”90 = 144

```
\DeclareTextComposite{\v}{T1}{R}{144}
\DeclareTextComposite{\'}{T1}{S}{145}
\DeclareTextComposite{\v}{T1}{S}{146}
\DeclareTextComposite{\c}{T1}{S}{147}
\DeclareTextComposite{\v}{T1}{T}{148}
\DeclareTextComposite{\c}{T1}{T}{149}
\DeclareTextComposite{\H}{T1}{U}{150}
\DeclareTextComposite{\r}{T1}{U}{151}
```

”98 = 152

```
\DeclareTextComposite{\"}{T1}{Y}{152}
\DeclareTextComposite{\'}{T1}{Z}{153}
\DeclareTextComposite{\v}{T1}{Z}{154}
\DeclareTextComposite{\.}{T1}{Z}{155}
\DeclareTextComposite{\.}{T1}{I}{157}
```

”A0 = 160

```
\DeclareTextComposite{\u}{T1}{a}{160}
\DeclareTextComposite{\k}{T1}{a}{161}
\DeclareTextComposite{\'}{T1}{c}{162}
\DeclareTextComposite{\v}{T1}{c}{163}
\DeclareTextComposite{\v}{T1}{d}{164}
\DeclareTextComposite{\v}{T1}{e}{165}
\DeclareTextComposite{\k}{T1}{e}{166}
\DeclareTextComposite{\u}{T1}{g}{167}
```

”A8 = 168

```
\DeclareTextComposite{\'}{T1}{l}{168}
\DeclareTextComposite{\v}{T1}{l}{169}
\DeclareTextComposite{\'}{T1}{n}{171}
\DeclareTextComposite{\v}{T1}{n}{172}
\DeclareTextComposite{\H}{T1}{o}{174}
\DeclareTextComposite{\'}{T1}{r}{175}
```

”B0 = 176

```
\DeclareTextComposite{\v}{T1}{r}{176}
\DeclareTextComposite{\'}{T1}{s}{177}
\DeclareTextComposite{\v}{T1}{s}{178}
\DeclareTextComposite{\c}{T1}{s}{179}
\DeclareTextComposite{\v}{T1}{t}{180}
```

```
\DeclareTextComposite{\c}{T1}{t}{181}
\DeclareTextComposite{\H}{T1}{u}{182}
\DeclareTextComposite{\r}{T1}{u}{183}
```

”B8 = 184

```
\DeclareTextComposite{\"}{T1}{y}{184}
\DeclareTextComposite{\'}{T1}{z}{185}
\DeclareTextComposite{\v}{T1}{z}{186}
\DeclareTextComposite{\.}{T1}{z}{187}
```

”C0 = 192

```
\DeclareTextComposite{\`}{T1}{A}{192}
\DeclareTextComposite{\'}{T1}{A}{193}
\DeclareTextComposite{\^}{T1}{A}{194}
\DeclareTextComposite{\~}{T1}{A}{195}
\DeclareTextComposite{\"}{T1}{A}{196}
\DeclareTextComposite{\r}{T1}{A}{197}
\DeclareTextComposite{\c}{T1}{C}{199}
```

”C8 = 200

```
\DeclareTextComposite{\`}{T1}{E}{200}
\DeclareTextComposite{\'}{T1}{E}{201}
\DeclareTextComposite{\^}{T1}{E}{202}
\DeclareTextComposite{\"}{T1}{E}{203}
\DeclareTextComposite{\`}{T1}{I}{204}
\DeclareTextComposite{\'}{T1}{I}{205}
\DeclareTextComposite{\^}{T1}{I}{206}
\DeclareTextComposite{\"}{T1}{I}{207}
```

”D0 = 208

```
\DeclareTextComposite{\~}{T1}{N}{209}
\DeclareTextComposite{\`}{T1}{O}{210}
\DeclareTextComposite{\'}{T1}{O}{211}
\DeclareTextComposite{\^}{T1}{O}{212}
\DeclareTextComposite{\~}{T1}{O}{213}
\DeclareTextComposite{\"}{T1}{O}{214}
```

”D8 = 216

```
\DeclareTextComposite{\`}{T1}{U}{217}
\DeclareTextComposite{\'}{T1}{U}{218}
\DeclareTextComposite{\^}{T1}{U}{219}
\DeclareTextComposite{\"}{T1}{U}{220}
\DeclareTextComposite{\'}{T1}{Y}{221}
```

”E0 = 224

```
\DeclareTextComposite{\`}{T1}{a}{224}
\DeclareTextComposite{\'}{T1}{a}{225}
\DeclareTextComposite{\^}{T1}{a}{226}
\DeclareTextComposite{\~}{T1}{a}{227}
\DeclareTextComposite{\"}{T1}{a}{228}
\DeclareTextComposite{\r}{T1}{a}{229}
\DeclareTextComposite{\c}{T1}{c}{231}
```

”E8 = 232

```
\DeclareTextComposite{\`}{T1}{e}{232}
\DeclareTextComposite{\'}{T1}{e}{233}
\DeclareTextComposite{\^}{T1}{e}{234}
```

```
\DeclareTextComposite{\"}{T1}{e}{235}
\DeclareTextComposite{\`}{T1}{i}{236}
\DeclareTextComposite{\`}{T1}{\i}{236}
\DeclareTextComposite{\'}{T1}{i}{237}
\DeclareTextComposite{\'}{T1}{\i}{237}
\DeclareTextComposite{\^}{T1}{i}{238}
\DeclareTextComposite{\^}{T1}{\i}{238}
\DeclareTextComposite{\"}{T1}{i}{239}
\DeclareTextComposite{\"}{T1}{\i}{239}
```

"F0 = 240

```
\DeclareTextComposite{\~}{T1}{n}{241}
\DeclareTextComposite{\`}{T1}{o}{242}
\DeclareTextComposite{\'}{T1}{o}{243}
\DeclareTextComposite{\^}{T1}{o}{244}
\DeclareTextComposite{\~}{T1}{o}{245}
\DeclareTextComposite{\"}{T1}{o}{246}
```

"F8 = 248

```
\DeclareTextComposite{\`}{T1}{u}{249}
\DeclareTextComposite{\'}{T1}{u}{250}
\DeclareTextComposite{\^}{T1}{u}{251}
\DeclareTextComposite{\"}{T1}{u}{252}
\DeclareTextComposite{\'}{T1}{y}{253}

\DeclareTextCompositeCommand{\k}{T1}{o}{\textogonekcentered{o}}
\DeclareTextCompositeCommand{\k}{T1}{O}{\textogonekcentered{O}}

\ifx\textcommaabove\@undefined\else
\DeclareTextCompositeCommand{\c}{T1}{g}{\textcommaabove{g}}
\fi
\ifx\textcommabelow\@undefined\else
\DeclareTextCompositeCommand{\c}{T1}{G}{\textcommabelow{G}}
\DeclareTextCompositeCommand{\c}{T1}{K}{\textcommabelow{K}}
\DeclareTextCompositeCommand{\c}{T1}{k}{\textcommabelow{k}}
\DeclareTextCompositeCommand{\c}{T1}{L}{\textcommabelow{L}}
\DeclareTextCompositeCommand{\c}{T1}{l}{\textcommabelow{l}}
\DeclareTextCompositeCommand{\c}{T1}{N}{\textcommabelow{N}}
\DeclareTextCompositeCommand{\c}{T1}{n}{\textcommabelow{n}}
\DeclareTextCompositeCommand{\c}{T1}{R}{\textcommabelow{R}}
\DeclareTextCompositeCommand{\c}{T1}{r}{\textcommabelow{r}}
\fi
⟨/T1⟩
```

20.7 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 LATEX text symbols.

Declare the encoding.

```
⟨*OMS⟩
\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!

```
\DeclareTextSymbol{\textasteriskcentered}{OMS}{3}   % "03
\DeclareTextSymbol{\textbackslash}{OMS}{110}        % "6E
\DeclareTextSymbol{\textbar}{OMS}{106}              % "6A
\DeclareTextSymbol{\textbardbl}{OMS}{107}           % "6B
\DeclareTextSymbol{\textbraceleft}{OMS}{102}        % "66
\DeclareTextSymbol{\textbraceright}{OMS}{103}       % "67
\DeclareTextSymbol{\textbullet}{OMS}{15}            % "0F
\DeclareTextSymbol{\textdaggerdbl}{OMS}{122}        % "7A
\DeclareTextSymbol{\textdagger}{OMS}{121}           % "79
\DeclareTextSymbol{\textparagraph}{OMS}{123}        % "7B
\DeclareTextSymbol{\textperiodcentered}{OMS}{1}     % "01
\DeclareTextSymbol{\textsection}{OMS}{120}          % "78
\DeclareTextSymbol{\textbigcircle}{OMS}{13}         % "0D
\DeclareTextCommand{\textcircled}{OMS}[1]{\hmode@bgroup
   \ooalign{%
      \hfil \raise .07ex\hbox {\upshape#1}\hfil \crcr
      \char 13 % "0D
   }%
 \egroup}
⟨/OMS⟩
```

20.8 Definitions for the OML encoding

The definitions for the 'TeX math italic' (OML) encoding. Even though this is meant to be a math font, it includes some of the standard LaTeX text symbols.

Declare the encoding.

```
⟨*OML⟩
\DeclareFontEncoding{OML}{}{}
```

Declare the symbols.

```
\DeclareTextSymbol{\textless}{OML}{`\<}
\DeclareTextSymbol{\textgreater}{OML}{`\>}
\DeclareTextAccent{\t}{OML}{127}  % "7F
⟨/OML⟩
```

20.9 Definitions for the OT4 encoding

These definitions are for the Polish extension to the 'TeX 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 LaTeX 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.

```
⟨*OT4⟩
\DeclareFontEncoding{OT4}{}{}
\DeclareFontSubstitution{OT4}{cmr}{m}{n}
```

Declare the accents.

```
\DeclareTextAccent{\"}{OT4}{127}
\DeclareTextAccent{\'}{OT4}{19}
```

```
\DeclareTextAccent{\.}{OT4}{95}
\DeclareTextAccent{\=}{OT4}{22}
\DeclareTextAccent{\^}{OT4}{94}
\DeclareTextAccent{\`}{OT4}{18}
\DeclareTextAccent{\~}{OT4}{126}
\DeclareTextAccent{\H}{OT4}{125}
\DeclareTextAccent{\u}{OT4}{21}
\DeclareTextAccent{\v}{OT4}{20}
\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:

```
\DeclareTextCommand{\k}{OT4}[1]{%
    \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.

```
\DeclareTextCommand{\b}{OT4}[1]
   {\hmode@bgroup\o@lign{\relax#1\crcr\hidewidth\ltx@sh@ft{-3ex}%
     \vbox to.2ex{\hbox{\char22}\vss}\hidewidth}\egroup}
\DeclareTextCommand{\c}{OT4}[1]
   {\leavevmode\setbox\z@\hbox{#1}\ifdim\ht\z@=1ex\accent24 #1%
    \else{\ooalign{\unhbox\z@\crcr\hidewidth\char24\hidewidth}}\fi}
\DeclareTextCommand{\d}{OT4}[1]
   {\hmode@bgroup
    \o@lign{\relax#1\crcr\hidewidth\ltx@sh@ft{-1ex}.\hidewidth}\egroup}
```

Declare the text symbols.

```
\DeclareTextSymbol{\AE}{OT4}{29}
\DeclareTextSymbol{\OE}{OT4}{30}
\DeclareTextSymbol{\O}{OT4}{31}
\DeclareTextSymbol{\L}{OT4}{138}
\DeclareTextSymbol{\ae}{OT4}{26}

\DeclareTextSymbol{\guillemetleft}{OT4}{174}
\DeclareTextSymbol{\guillemetright}{OT4}{175}
% old Adobe names
\DeclareTextSymbol{\guillemotleft}{OT4}{174}
\DeclareTextSymbol{\guillemotright}{OT4}{175}

\DeclareTextSymbol{\i}{OT4}{16}
\DeclareTextSymbol{\j}{OT4}{17}
\DeclareTextSymbol{\l}{OT4}{170}
\DeclareTextSymbol{\o}{OT4}{28}
\DeclareTextSymbol{\oe}{OT4}{27}
\DeclareTextSymbol{\quotedblbase}{OT4}{255}
\DeclareTextSymbol{\ss}{OT4}{25}
\DeclareTextSymbol{\textemdash}{OT4}{124}
\DeclareTextSymbol{\textendash}{OT4}{123}
\DeclareTextSymbol{\textexclamdown}{OT4}{60}
%\DeclareTextSymbol{\texthyphenchar}{OT4}{`\-}
%\DeclareTextSymbol{\texthyphen}{OT4}{`\-}
\DeclareTextSymbol{\textquestiondown}{OT4}{62}
\DeclareTextSymbol{\textquotedblleft}{OT4}{92}
\DeclareTextSymbol{\textquotedblright}{OT4}{`\"}
\DeclareTextSymbol{\textquoteleft}{OT4}{`\`}
```

```
\DeclareTextSymbol{\textquoteright}{OT4}{`\'}
```

Definition for Å as in OT1:

```
\DeclareTextCompositeCommand{\r}{OT4}{A}
  {\leavevmode\setbox\z@\hbox{!}\dimen@\ht\z@\advance\dimen@-1ex%
   \rlap{\raise.67\dimen@\hbox{\char23}}A}
```

In the OT4 encoding, £ and $ share a slot.

```
\DeclareTextCommand{\textdollar}{OT4}{\hmode@bgroup
  \ifdim \fontdimen\@ne\font >\z@
    \slshape
  \else
    \upshape
  \fi
  \char`\$\egroup}
\DeclareTextCommand{\textsterling}{OT4}{\hmode@bgroup
  \ifdim \fontdimen\@ne\font >\z@
    \itshape
  \else
    \fontshape{ui}\selectfont
  \fi
  \char`\$\egroup}
```

Declare the composites.

```
\DeclareTextComposite{\k}{OT4}{A}{129}
\DeclareTextComposite{\'}{OT4}{C}{130}
\DeclareTextComposite{\k}{OT4}{E}{134}
\DeclareTextComposite{\'}{OT4}{N}{139}
\DeclareTextComposite{\'}{OT4}{S}{145}
\DeclareTextComposite{\'}{OT4}{Z}{153}
\DeclareTextComposite{\.}{OT4}{Z}{155}
\DeclareTextComposite{\k}{OT4}{a}{161}
\DeclareTextComposite{\'}{OT4}{c}{162}
\DeclareTextComposite{\k}{OT4}{e}{166}
\DeclareTextComposite{\'}{OT4}{n}{171}
\DeclareTextComposite{\'}{OT4}{s}{177}
\DeclareTextComposite{\'}{OT4}{z}{185}
\DeclareTextComposite{\.}{OT4}{z}{187}
\DeclareTextComposite{\'}{OT4}{O}{211}
\DeclareTextComposite{\'}{OT4}{o}{243}
⟨/OT4⟩
```

20.10 Definitions for the TS1 encoding

```
⟨*TS1⟩
\DeclareFontEncoding{TS1}{}{}
\DeclareFontSubstitution{TS1}{cmr}{m}{n}
```

Some accents have to be built by hand. Note that `\ooalign` and `\o@lign` must be inside a group.

```
\DeclareTextCommand{\capitalcedilla}{TS1}[1]
  {\hmode@bgroup
   \ooalign{\null#1\crcr\hidewidth\char11\hidewidth}\egroup}
\DeclareTextCommand{\capitalogonek}{TS1}[1]
  {\hmode@bgroup
```

```
    \ooalign{\null#1\crcr\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

```
\DeclareTextAccent{\capitalgrave}{TS1}{0}
\DeclareTextAccent{\capitalacute}{TS1}{1}
\DeclareTextAccent{\capitalcircumflex}{TS1}{2}
\DeclareTextAccent{\capitaltilde}{TS1}{3}
\DeclareTextAccent{\capitaldieresis}{TS1}{4}
\DeclareTextAccent{\capitalhungarumlaut}{TS1}{5}
\DeclareTextAccent{\capitalring}{TS1}{6}
\DeclareTextAccent{\capitalcaron}{TS1}{7}
```

"08 = 8

```
\DeclareTextAccent{\capitalbreve}{TS1}{8}
\DeclareTextAccent{\capitalmacron}{TS1}{9}
\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 assymetric 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.

" =

```
\DeclareTextAccent{\t}{TS1}{26}
\DeclareTextAccent{\capitaltie}{TS1}{27}
\DeclareTextAccent{\newtie}{TS1}{28}
\DeclareTextAccent{\capitalnewtie}{TS1}{29}
```

Compund word marks.

The text companion fonts contain two compound word marks of different heights, one has `cap_height`, the other `asc_height`.

```
\DeclareTextSymbol{\textcapitalcompwordmark}{TS1}{23}
\DeclareTextSymbol{\textascendercompwordmark}{TS1}{31}
```

The text companion symbols.

```
\DeclareTextSymbol{\textquotestraightbase}{TS1}{13}
```

"10 = 16

```
\DeclareTextSymbol{\textquotestraightdblbase}{TS1}{18}
\DeclareTextSymbol{\texttwelveudash}{TS1}{21}
\DeclareTextSymbol{\textthreequartersemdash}{TS1}{22}
```

"18 = 24

```
\DeclareTextSymbol{\textleftarrow}{TS1}{24}
\DeclareTextSymbol{\textrightarrow}{TS1}{25}
```

”20 = 32

```
\DeclareTextSymbol{\textblank}{TS1}{32}
\DeclareTextSymbol{\textdollar}{TS1}{36}
\DeclareTextSymbol{\textquotesingle}{TS1}{39}
```

”28 = 40

```
\DeclareTextSymbol{\textasteriskcentered}{TS1}{42}
```

Note that ’054 is a comma and ’056 is a full stop: these make numbers using oldstyle digits easier to input.

```
\DeclareTextSymbol{\textdblhyphen}{TS1}{45}
\DeclareTextSymbol{\textfractionsolidus}{TS1}{47}
```

Oldstyle digits.
”30 = 48

```
\DeclareTextSymbol{\textzerooldstyle}{TS1}{48}
\DeclareTextSymbol{\textoneoldstyle}{TS1}{49}
\DeclareTextSymbol{\texttwooldstyle}{TS1}{50}
\DeclareTextSymbol{\textthreeoldstyle}{TS1}{51}
\DeclareTextSymbol{\textfouroldstyle}{TS1}{52}
\DeclareTextSymbol{\textfiveoldstyle}{TS1}{53}
\DeclareTextSymbol{\textsixoldstyle}{TS1}{54}
\DeclareTextSymbol{\textsevenoldstyle}{TS1}{55}
```

”38 = 56

```
\DeclareTextSymbol{\texteightoldstyle}{TS1}{56}
\DeclareTextSymbol{\textnineoldstyle}{TS1}{57}
```

More text companion symbols.

```
\DeclareTextSymbol{\textlangle}{TS1}{60}
\DeclareTextSymbol{\textminus}{TS1}{61}
\DeclareTextSymbol{\textrangle}{TS1}{62}
```

”48 = 72

```
\DeclareTextSymbol{\textmho}{TS1}{77}
```

The big circle is here to define the command `\textcircled`. Formerly it was taken from the `cmsy` font.

```
\DeclareTextSymbol{\textbigcircle}{TS1}{79}
\DeclareTextCommand{\textcircled}{TS1}[1]{\hmode@bgroup
  \ooalign{%
     \hfil \raise .07ex\hbox {\upshape#1}\hfil \crcr
     \char 79   % '117 = "4F
  }%
 \egroup}
```

More text companion symbols.
”50 = 80

```
\DeclareTextSymbol{\textohm}{TS1}{87}
```

”58 = 88

```
\DeclareTextSymbol{\textlbrackdbl}{TS1}{91}
\DeclareTextSymbol{\textrbrackdbl}{TS1}{93}
\DeclareTextSymbol{\textuparrow}{TS1}{94}
\DeclareTextSymbol{\textdownarrow}{TS1}{95}
```

”60 = 96

```
\DeclareTextSymbol{\textasciigrave}{TS1}{96}
\DeclareTextSymbol{\textborn}{TS1}{98}
\DeclareTextSymbol{\textdivorced}{TS1}{99}
\DeclareTextSymbol{\textdied}{TS1}{100}
```

”68 = 104

```
\DeclareTextSymbol{\textleaf}{TS1}{108}
\DeclareTextSymbol{\textmarried}{TS1}{109}
\DeclareTextSymbol{\textmusicalnote}{TS1}{110}
```

”78 = 120

```
\DeclareTextSymbol{\texttildelow}{TS1}{126}
```

This glyph, `\textdblhyphenchar` is hanging, like the hyphenchar of the ec fonts.

```
\DeclareTextSymbol{\textdblhyphenchar}{TS1}{127}
```

”80 = 128

```
\DeclareTextSymbol{\textasciibreve}{TS1}{128}
\DeclareTextSymbol{\textasciicaron}{TS1}{129}
```

This next glyph is *not* the same as `\textquotedbl`.

```
\DeclareTextSymbol{\textacutedbl}{TS1}{130}
\DeclareTextSymbol{\textgravedbl}{TS1}{131}
\DeclareTextSymbol{\textdagger}{TS1}{132}
\DeclareTextSymbol{\textdaggerdbl}{TS1}{133}
\DeclareTextSymbol{\textbardbl}{TS1}{134}
\DeclareTextSymbol{\textperthousand}{TS1}{135}
```

”88 = 136

```
\DeclareTextSymbol{\textbullet}{TS1}{136}
\DeclareTextSymbol{\textcelsius}{TS1}{137}
\DeclareTextSymbol{\textdollaroldstyle}{TS1}{138}
\DeclareTextSymbol{\textcentoldstyle}{TS1}{139}
\DeclareTextSymbol{\textflorin}{TS1}{140}
\DeclareTextSymbol{\textcolonmonetary}{TS1}{141}
\DeclareTextSymbol{\textwon}{TS1}{142}
\DeclareTextSymbol{\textnaira}{TS1}{143}
```

”90 = 144

```
\DeclareTextSymbol{\textguarani}{TS1}{144}
\DeclareTextSymbol{\textpeso}{TS1}{145}
\DeclareTextSymbol{\textlira}{TS1}{146}
\DeclareTextSymbol{\textrecipe}{TS1}{147}
\DeclareTextSymbol{\textinterrobang}{TS1}{148}
\DeclareTextSymbol{\textinterrobangdown}{TS1}{149}
\DeclareTextSymbol{\textdong}{TS1}{150}
\DeclareTextSymbol{\texttrademark}{TS1}{151}
```

”98 = 152

```
\DeclareTextSymbol{\textpertenthousand}{TS1}{152}
\DeclareTextSymbol{\textpilcrow}{TS1}{153}
\DeclareTextSymbol{\textbaht}{TS1}{154}
\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`.

```
\DeclareTextSymbol{\textdiscount}{TS1}{156}
\DeclareTextSymbol{\textestimated}{TS1}{157}
\DeclareTextSymbol{\textopenbullet}{TS1}{158}
\DeclareTextSymbol{\textservicemark}{TS1}{159}
```

”A0 = 160

```
\DeclareTextSymbol{\textlquill}{TS1}{160}
\DeclareTextSymbol{\textrquill}{TS1}{161}
\DeclareTextSymbol{\textcent}{TS1}{162}
\DeclareTextSymbol{\textsterling}{TS1}{163}
\DeclareTextSymbol{\textcurrency}{TS1}{164}
\DeclareTextSymbol{\textyen}{TS1}{165}
\DeclareTextSymbol{\textbrokenbar}{TS1}{166}
\DeclareTextSymbol{\textsection}{TS1}{167}
```

”A8 = 168

```
\DeclareTextSymbol{\textasciidieresis}{TS1}{168}
\DeclareTextSymbol{\textcopyright}{TS1}{169}
\DeclareTextSymbol{\textordfeminine}{TS1}{170}
\DeclareTextSymbol{\textcopyleft}{TS1}{171}
\DeclareTextSymbol{\textlnot}{TS1}{172}
```

The meaning of the circled-P is “sound recording copyright”.

```
\DeclareTextSymbol{\textcircledP}{TS1}{173}
\DeclareTextSymbol{\textregistered}{TS1}{174}
\DeclareTextSymbol{\textasciimacron}{TS1}{175}
```

”B0 = 176

```
\DeclareTextSymbol{\textdegree}{TS1}{176}
\DeclareTextSymbol{\textpm}{TS1}{177}
\DeclareTextSymbol{\texttwosuperior}{TS1}{178}
\DeclareTextSymbol{\textthreesuperior}{TS1}{179}
\DeclareTextSymbol{\textasciiacute}{TS1}{180}
\DeclareTextSymbol{\textmu}{TS1}{181} % micro sign
\DeclareTextSymbol{\textparagraph}{TS1}{182}
\DeclareTextSymbol{\textperiodcentered}{TS1}{183}
```

”B8 = 184

```
\DeclareTextSymbol{\textreferencemark}{TS1}{184}
\DeclareTextSymbol{\textonesuperior}{TS1}{185}
\DeclareTextSymbol{\textordmasculine}{TS1}{186}
\DeclareTextSymbol{\textsurd}{TS1}{187}
\DeclareTextSymbol{\textonequarter}{TS1}{188}
\DeclareTextSymbol{\textonehalf}{TS1}{189}
\DeclareTextSymbol{\textthreequarters}{TS1}{190}
\DeclareTextSymbol{\texteuro}{TS1}{191}
```

”E0 = 208

```
\DeclareTextSymbol{\texttimes}{TS1}{214}
```

”F0 = 240

```
\DeclareTextSymbol{\textdiv}{TS1}{246}
⟨/TS1⟩
```

20.11 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 form the `inputenc` UTF-8 support, which is essentially doing a reverse mapping from UTF-8 data to LaTeX 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.

```
⟨*TU⟩
```

In the base interface the Unicode encoding is always known as TU But we parameterise the encoding name to allow for modelling differences in Unicode support by different fonts.

```
\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.

```
\begingroup\expandafter\expandafter\expandafter\endgroup
\expandafter\ifx\csname XeTeXrevision\endcsname\relax
```

```
  \begingroup\expandafter\expandafter\expandafter\endgroup
  \expandafter\ifx\csname directlua\endcsname\relax
```

Not LuaTeX or XeTeX, abort with a warning.

```
    \PackageWarningNoLine{fontenc}
      {\UnicodeEncodingName\space
       encoding is only available with XeTeX and LuaTeX.\MessageBreak
       Defaulting to T1 encoding}
      \def\encodingdefault{T1}
    \expandafter\expandafter\expandafter\endinput
```

```
  \else
```

LuaTeX.

```
    \def\UnicodeFontTeXLigatures{+tlig;}
```

```
    \def\reserved@a#1{%
      \def\@remove@tlig##1{\@remove@tlig@##1\@nil#1\@nil\relax}
      \def\@remove@tlig@##1#1{\@remove@tlig@@##1}}
    \edef\reserved@b{\detokenize{+tlig;}}
    \expandafter\reserved@a\expandafter{\reserved@b}
    \def\@remove@tlig@@#1\@nil#2\relax{#1}
```

```
    \def\remove@tlig#1{%
      \begingroup
      \font\remove@tlig
      \expandafter\@remove@tlig\expandafter{\fontname\font}%
      \remove@tlig
      \char#1\relax
      \endgroup
    }

  \fi
\else
```

XeTeX

```
  \def\UnicodeFontTeXLigatures{mapping=tex-text;}

  \def\remove@tlig#1{\XeTeXglyph\numexpr\XeTeXcharglyph#1\relax}

\fi

\def\UnicodeFontFile#1#2{"[#1]:#2"}
\def\UnicodeFontName#1#2{"#1:#2"}
```

Declare the encoding

```
\DeclareFontEncoding\UnicodeEncodingName{}{}
```

Declare accent command to use a postpended combining character rather than the TeX `\accent` primitive

```
\def\add@unicode@accent#1#2{%
  \if\relax\detokenize{#2}\relax^^a0\else#2\fi
  \char#1\relax}

\def\DeclareUnicodeAccent#1#2#3{%
  \DeclareTextCommand{#1}{#2}{\add@unicode@accent{#3}}%
}
```

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.

```
{
\catcode\z@=11\relax
\gdef\DeclareUnicodeComposite#1#2#3{%
   \def\reserved@a##1##2{%
     \DeclareTextCompositeCommand#1\UnicodeEncodingName{#2}{%
   \iffontchar\font#3 ##2%
      \else ##1\fi}}%
    \expandafter\expandafter\expandafter\extract@default@composite
    \csname\UnicodeEncodingName\string#1\endcsname{#2}\@nil
   \bgroup
      \lccode\z@#3 %
      \lowercase{\egroup
      \expandafter\reserved@a\expandafter{\reserved@b}{^^@}}}%
}

\def\extract@default@composite#1{%
 \ifx\@text@composite#1%
   \expandafter\extract@default@composite@a
 \else
   \expandafter\extract@default@composite@b\expandafter#1%
 \fi}
```

```
\def\extract@default@composite@a#1\@text@composite#2\@nil{%
  \def\reserved@b{#2}}
\def\extract@default@composite@b#1#2\@nil{%
  \def\reserved@b{#1#2}}
```

```
\DeclareTextCommand\textquotesingle \UnicodeEncodingName{%
                                                \remove@tlig{"0027}}
\DeclareTextCommand\textasciigrave  \UnicodeEncodingName{%
                                                \remove@tlig{"0060}}
\DeclareTextCommand\textquotedbl    \UnicodeEncodingName{%
                                                \remove@tlig{"0022}}
```

```
\DeclareTextSymbol{\textdollar}          \UnicodeEncodingName{"0024}
\DeclareTextSymbol{\textless}            \UnicodeEncodingName{"003C}
\DeclareTextSymbol{\textgreater}         \UnicodeEncodingName{"003E}
\DeclareTextSymbol{\textbackslash}       \UnicodeEncodingName{"005C}
\DeclareTextSymbol{\textasciicircum}     \UnicodeEncodingName{"005E}
\DeclareTextSymbol{\textunderscore}      \UnicodeEncodingName{"005F}
\DeclareTextSymbol{\textbraceleft}       \UnicodeEncodingName{"007B}
\DeclareTextSymbol{\textbar}             \UnicodeEncodingName{"007C}
\DeclareTextSymbol{\textbraceright}      \UnicodeEncodingName{"007D}
\DeclareTextSymbol{\textasciitilde}      \UnicodeEncodingName{"007E}
\DeclareTextSymbol{\textexclamdown}      \UnicodeEncodingName{"00A1}
\DeclareTextSymbol{\textcent}            \UnicodeEncodingName{"00A2}
\DeclareTextSymbol{\textsterling}        \UnicodeEncodingName{"00A3}
\DeclareTextSymbol{\textcurrency}        \UnicodeEncodingName{"00A4}
\DeclareTextSymbol{\textyen}             \UnicodeEncodingName{"00A5}
\DeclareTextSymbol{\textbrokenbar}       \UnicodeEncodingName{"00A6}
\DeclareTextSymbol{\textsection}         \UnicodeEncodingName{"00A7}
\DeclareTextSymbol{\textasciidieresis}   \UnicodeEncodingName{"00A8}
\DeclareTextSymbol{\textcopyright}       \UnicodeEncodingName{"00A9}
\DeclareTextSymbol{\textordfeminine}     \UnicodeEncodingName{"00AA}
```

```
\DeclareTextSymbol{\guillemetleft}       \UnicodeEncodingName{"00AB}
% old Adobe name
\DeclareTextSymbol{\guillemotleft}       \UnicodeEncodingName{"00AB}
```

```
\DeclareTextSymbol{\textlnot}            \UnicodeEncodingName{"00AC}
\DeclareTextSymbol{\textregistered}      \UnicodeEncodingName{"00AE}
\DeclareTextSymbol{\textasciimacron}     \UnicodeEncodingName{"00AF}
\DeclareTextSymbol{\textdegree}          \UnicodeEncodingName{"00B0}
\DeclareTextSymbol{\textpm}              \UnicodeEncodingName{"00B1}
\DeclareTextSymbol{\texttwosuperior}     \UnicodeEncodingName{"00B2}
\DeclareTextSymbol{\textthreesuperior}   \UnicodeEncodingName{"00B3}
\DeclareTextSymbol{\textasciiacute}      \UnicodeEncodingName{"00B4}
\DeclareTextSymbol{\textmu}              \UnicodeEncodingName{"00B5}
\DeclareTextSymbol{\textparagraph}       \UnicodeEncodingName{"00B6}
\DeclareTextSymbol{\textperiodcentered}  \UnicodeEncodingName{"00B7}
\DeclareTextSymbol{\textonesuperior}     \UnicodeEncodingName{"00B9}
\DeclareTextSymbol{\textordmasculine}    \UnicodeEncodingName{"00BA}
```

```
\DeclareTextSymbol{\guillemetright}      \UnicodeEncodingName{"00BB}
% old Adobe name
\DeclareTextSymbol{\guillemotright}      \UnicodeEncodingName{"00BB}
```

```
\DeclareTextSymbol{\textonequarter}      \UnicodeEncodingName{"00BC}
\DeclareTextSymbol{\textonehalf}         \UnicodeEncodingName{"00BD}
```

```
\DeclareTextSymbol{\textthreequarters}   \UnicodeEncodingName{"00BE}
\DeclareTextSymbol{\textquestiondown}    \UnicodeEncodingName{"00BF}
\DeclareTextSymbol{\AE}                  \UnicodeEncodingName{"00C6}
\DeclareTextSymbol{\DH}                  \UnicodeEncodingName{"00D0}
\DeclareTextSymbol{\texttimes}           \UnicodeEncodingName{"00D7}
\DeclareTextSymbol{\O}                   \UnicodeEncodingName{"00D8}
\DeclareTextSymbol{\TH}                  \UnicodeEncodingName{"00DE}
\DeclareTextSymbol{\ss}                  \UnicodeEncodingName{"00DF}
\DeclareTextSymbol{\ae}                  \UnicodeEncodingName{"00E6}
\DeclareTextSymbol{\dh}                  \UnicodeEncodingName{"00F0}
\DeclareTextSymbol{\textdiv}             \UnicodeEncodingName{"00F7}
\DeclareTextSymbol{\o}                   \UnicodeEncodingName{"00F8}
\DeclareTextSymbol{\th}                  \UnicodeEncodingName{"00FE}
\DeclareTextSymbol{\DJ}                  \UnicodeEncodingName{"0110}
\DeclareTextSymbol{\dj}                  \UnicodeEncodingName{"0111}
\DeclareTextSymbol{\i}                   \UnicodeEncodingName{"0131}
\DeclareTextSymbol{\IJ}                  \UnicodeEncodingName{"0132}
\DeclareTextSymbol{\ij}                  \UnicodeEncodingName{"0133}
\DeclareTextSymbol{\L}                   \UnicodeEncodingName{"0141}
\DeclareTextSymbol{\l}                   \UnicodeEncodingName{"0142}
\DeclareTextSymbol{\NG}                  \UnicodeEncodingName{"014A}
\DeclareTextSymbol{\ng}                  \UnicodeEncodingName{"014B}
\DeclareTextSymbol{\OE}                  \UnicodeEncodingName{"0152}
\DeclareTextSymbol{\oe}                  \UnicodeEncodingName{"0153}
\DeclareTextSymbol{\textflorin}          \UnicodeEncodingName{"0192}
\DeclareTextSymbol{\j}                   \UnicodeEncodingName{"0237}
\DeclareTextSymbol{\textasciicaron}      \UnicodeEncodingName{"02C7}
\DeclareTextSymbol{\textasciibreve}      \UnicodeEncodingName{"02D8}
\DeclareTextSymbol{\textacutedbl}        \UnicodeEncodingName{"02DD}
\DeclareTextSymbol{\textgravedbl}        \UnicodeEncodingName{"02F5}
\DeclareTextSymbol{\texttildelow}        \UnicodeEncodingName{"02F7}
\DeclareTextSymbol{\textbaht}            \UnicodeEncodingName{"0E3F}
\DeclareTextSymbol{\SS}                  \UnicodeEncodingName{"1E9E}
\DeclareTextSymbol{\textcompwordmark}    \UnicodeEncodingName{"200C}
\DeclareTextSymbol{\textendash}          \UnicodeEncodingName{"2013}
\DeclareTextSymbol{\textemdash}          \UnicodeEncodingName{"2014}
\DeclareTextSymbol{\textbardbl}          \UnicodeEncodingName{"2016}
\DeclareTextSymbol{\textquoteleft}       \UnicodeEncodingName{"2018}
\DeclareTextSymbol{\textquoteright}      \UnicodeEncodingName{"2019}
\DeclareTextSymbol{\quotesinglbase}      \UnicodeEncodingName{"201A}
\DeclareTextSymbol{\textquotedblleft}    \UnicodeEncodingName{"201C}
\DeclareTextSymbol{\textquotedblright}   \UnicodeEncodingName{"201D}
\DeclareTextSymbol{\quotedblbase}        \UnicodeEncodingName{"201E}
\DeclareTextSymbol{\textdagger}          \UnicodeEncodingName{"2020}
\DeclareTextSymbol{\textdaggerdbl}       \UnicodeEncodingName{"2021}
\DeclareTextSymbol{\textbullet}          \UnicodeEncodingName{"2022}
\DeclareTextSymbol{\textellipsis}        \UnicodeEncodingName{"2026}
\DeclareTextSymbol{\textperthousand}     \UnicodeEncodingName{"2030}
\DeclareTextSymbol{\textpertenthousand}  \UnicodeEncodingName{"2031}
\DeclareTextSymbol{\guilsinglleft}       \UnicodeEncodingName{"2039}
\DeclareTextSymbol{\guilsinglright}      \UnicodeEncodingName{"203A}
\DeclareTextSymbol{\textreferencemark}   \UnicodeEncodingName{"203B}
\DeclareTextSymbol{\textinterrobang}     \UnicodeEncodingName{"203D}
\DeclareTextSymbol{\textfractionsolidus} \UnicodeEncodingName{"2044}
```

```
\DeclareTextSymbol{\textlquill}          \UnicodeEncodingName{"2045}
\DeclareTextSymbol{\textrquill}          \UnicodeEncodingName{"2046}
\DeclareTextSymbol{\textdiscount}        \UnicodeEncodingName{"2052}
\DeclareTextSymbol{\textcolonmonetary}   \UnicodeEncodingName{"20A1}
\DeclareTextSymbol{\textlira}            \UnicodeEncodingName{"20A4}
\DeclareTextSymbol{\textnaira}           \UnicodeEncodingName{"20A6}
\DeclareTextSymbol{\textwon}             \UnicodeEncodingName{"20A9}
\DeclareTextSymbol{\textdong}            \UnicodeEncodingName{"20AB}
\DeclareTextSymbol{\texteuro}            \UnicodeEncodingName{"20AC}
\DeclareTextSymbol{\textpeso}            \UnicodeEncodingName{"20B1}
\DeclareTextSymbol{\textcelsius}         \UnicodeEncodingName{"2103}
\DeclareTextSymbol{\textnumero}          \UnicodeEncodingName{"2116}
\DeclareTextSymbol{\textcircledP}        \UnicodeEncodingName{"2117}
\DeclareTextSymbol{\textrecipe}          \UnicodeEncodingName{"211E}
\DeclareTextSymbol{\textservicemark}     \UnicodeEncodingName{"2120}
\DeclareTextSymbol{\texttrademark}       \UnicodeEncodingName{"2122}
\DeclareTextSymbol{\textohm}             \UnicodeEncodingName{"2126}
\DeclareTextSymbol{\textmho}             \UnicodeEncodingName{"2127}
\DeclareTextSymbol{\textestimated}       \UnicodeEncodingName{"212E}
\DeclareTextSymbol{\textleftarrow}       \UnicodeEncodingName{"2190}
\DeclareTextSymbol{\textuparrow}         \UnicodeEncodingName{"2191}
\DeclareTextSymbol{\textrightarrow}      \UnicodeEncodingName{"2192}
\DeclareTextSymbol{\textdownarrow}       \UnicodeEncodingName{"2193}
\DeclareTextSymbol{\textminus}           \UnicodeEncodingName{"2212}

```

```
\DeclareTextSymbol{\Hwithstroke}         \UnicodeEncodingName{"0126}
\DeclareTextSymbol{\hwithstroke}         \UnicodeEncodingName{"0127}
```

Not all fonts have U+2217 but using U+002A requires some adjustment.

```
\DeclareTextCommand{\textasteriskcentered}\UnicodeEncodingName{%
  \iffontchar\font"2217 \char"2217 \else
    \begingroup
      \fontsize
       {\the\dimexpr1.2\dimexpr\f@size pt\relax}%
       {\f@baselineskip}%
      \selectfont
      \raisebox{-0.6ex}[\dimexpr\height-0.6ex][0pt]{*}%
    \endgroup
  \fi
}
```

```
\DeclareTextSymbol{\textsurd}            \UnicodeEncodingName{"221A}
\DeclareTextSymbol{\textlangle}          \UnicodeEncodingName{"2329}
\DeclareTextSymbol{\textrangle}          \UnicodeEncodingName{"232A}
\DeclareTextSymbol{\textblank}           \UnicodeEncodingName{"2422}
\DeclareTextSymbol{\textvisiblespace}    \UnicodeEncodingName{"2423}
\DeclareTextSymbol{\textopenbullet}      \UnicodeEncodingName{"25E6}
\DeclareTextSymbol{\textbigcircle}       \UnicodeEncodingName{"25EF}
\DeclareTextSymbol{\textmusicalnote}     \UnicodeEncodingName{"266A}
\DeclareTextSymbol{\textmarried}         \UnicodeEncodingName{"26AD}
\DeclareTextSymbol{\textdivorced}        \UnicodeEncodingName{"26AE}
\DeclareTextSymbol{\textinterrobangdown} \UnicodeEncodingName{"2E18}
```

Accents must be declared before the composites that use them.

```
\DeclareUnicodeAccent{\`}                \UnicodeEncodingName{"0300}
\DeclareUnicodeAccent{\'}                \UnicodeEncodingName{"0301}
\DeclareUnicodeAccent{\^}                \UnicodeEncodingName{"0302}
\DeclareUnicodeAccent{\~}                \UnicodeEncodingName{"0303}
\DeclareUnicodeAccent{\=}                \UnicodeEncodingName{"0304}
\DeclareUnicodeAccent{\u}                \UnicodeEncodingName{"0306}
\DeclareUnicodeAccent{\.}                \UnicodeEncodingName{"0307}
\DeclareUnicodeAccent{\"}                \UnicodeEncodingName{"0308}
\DeclareUnicodeAccent{\r}                \UnicodeEncodingName{"030A}
\DeclareUnicodeAccent{\H}                \UnicodeEncodingName{"030B}
\DeclareUnicodeAccent{\v}                \UnicodeEncodingName{"030C}
\DeclareUnicodeAccent{\b}                \UnicodeEncodingName{"0332}
\DeclareUnicodeAccent{\d}                \UnicodeEncodingName{"0323}
\DeclareUnicodeAccent{\c}                \UnicodeEncodingName{"0327}
\DeclareUnicodeAccent{\k}                \UnicodeEncodingName{"0328}
\DeclareTextCommand\textcommabelow       \UnicodeEncodingName[1]
  {\hmode@bgroup\ooalign{\null#1\crcr\hidewidth\raise-.31ex
   \hbox{\check@mathfonts\fontsize\ssf@size\z@
   \math@fontsfalse\selectfont,}\hidewidth}\egroup}

\DeclareUnicodeComposite{\^}              {}{"005E}
\DeclareUnicodeComposite{\~}              {}{"007E}

\DeclareUnicodeComposite{\`}             {A}{"00C0}
\DeclareUnicodeComposite{\'}             {A}{"00C1}
\DeclareUnicodeComposite{\^}             {A}{"00C2}
\DeclareUnicodeComposite{\~}             {A}{"00C3}
\DeclareUnicodeComposite{\"}             {A}{"00C4}
\DeclareUnicodeComposite{\r}             {A}{"00C5}
\DeclareUnicodeComposite{\c}             {C}{"00C7}
\DeclareUnicodeComposite{\`}             {E}{"00C8}
\DeclareUnicodeComposite{\'}             {E}{"00C9}
\DeclareUnicodeComposite{\^}             {E}{"00CA}
\DeclareUnicodeComposite{\"}             {E}{"00CB}
\DeclareUnicodeComposite{\`}             {I}{"00CC}
\DeclareUnicodeComposite{\'}             {I}{"00CD}
\DeclareUnicodeComposite{\^}             {I}{"00CE}
\DeclareUnicodeComposite{\"}             {I}{"00CF}
\DeclareUnicodeComposite{\~}             {N}{"00D1}
\DeclareUnicodeComposite{\`}             {O}{"00D2}
\DeclareUnicodeComposite{\'}             {O}{"00D3}
\DeclareUnicodeComposite{\^}             {O}{"00D4}
\DeclareUnicodeComposite{\~}             {O}{"00D5}
\DeclareUnicodeComposite{\"}             {O}{"00D6}
\DeclareUnicodeComposite{\`}             {U}{"00D9}
\DeclareUnicodeComposite{\'}             {U}{"00DA}
\DeclareUnicodeComposite{\^}             {U}{"00DB}
\DeclareUnicodeComposite{\"}             {U}{"00DC}
\DeclareUnicodeComposite{\'}             {Y}{"00DD}
\DeclareUnicodeComposite{\`}             {a}{"00E0}
\DeclareUnicodeComposite{\'}             {a}{"00E1}
\DeclareUnicodeComposite{\^}             {a}{"00E2}
\DeclareUnicodeComposite{\~}             {a}{"00E3}
\DeclareUnicodeComposite{\"}             {a}{"00E4}
\DeclareUnicodeComposite{\r}             {a}{"00E5}
```

```
\DeclareUnicodeComposite{\c}        {c}{"00E7}
\DeclareUnicodeComposite{\`}        {e}{"00E8}
\DeclareUnicodeComposite{\'}        {e}{"00E9}
\DeclareUnicodeComposite{\^}        {e}{"00EA}
\DeclareUnicodeComposite{\"}        {e}{"00EB}
\DeclareUnicodeComposite{\`}        \i {"00EC}
\DeclareUnicodeComposite{\`}        {i}{"00EC}
\DeclareUnicodeComposite{\'}        \i {"00ED}
\DeclareUnicodeComposite{\'}        {i}{"00ED}
\DeclareUnicodeComposite{\^}        \i {"00EE}
\DeclareUnicodeComposite{\^}        {i}{"00EE}
\DeclareUnicodeComposite{\"}        \i {"00EF}
\DeclareUnicodeComposite{\"}        {i}{"00EF}
\DeclareUnicodeComposite{\~}        {n}{"00F1}
\DeclareUnicodeComposite{\`}        {o}{"00F2}
\DeclareUnicodeComposite{\'}        {o}{"00F3}
\DeclareUnicodeComposite{\^}        {o}{"00F4}
\DeclareUnicodeComposite{\~}        {o}{"00F5}
\DeclareUnicodeComposite{\"}        {o}{"00F6}
\DeclareUnicodeComposite{\`}        {u}{"00F9}
\DeclareUnicodeComposite{\'}        {u}{"00FA}
\DeclareUnicodeComposite{\^}        {u}{"00FB}
\DeclareUnicodeComposite{\"}        {u}{"00FC}
\DeclareUnicodeComposite{\'}        {y}{"00FD}
\DeclareUnicodeComposite{\"}        {y}{"00FF}
\DeclareUnicodeComposite{\=}        {A}{"0100}
\DeclareUnicodeComposite{\=}        {a}{"0101}
\DeclareUnicodeComposite{\u}        {A}{"0102}
\DeclareUnicodeComposite{\u}        {a}{"0103}
\DeclareUnicodeComposite{\k}        {A}{"0104}
\DeclareUnicodeComposite{\k}        {a}{"0105}
\DeclareUnicodeComposite{\'}        {C}{"0106}
\DeclareUnicodeComposite{\'}        {c}{"0107}
\DeclareUnicodeComposite{\^}        {C}{"0108}
\DeclareUnicodeComposite{\^}        {c}{"0109}
\DeclareUnicodeComposite{\.}        {C}{"010A}
\DeclareUnicodeComposite{\.}        {c}{"010B}
\DeclareUnicodeComposite{\v}        {C}{"010C}
\DeclareUnicodeComposite{\v}        {c}{"010D}
\DeclareUnicodeComposite{\v}        {D}{"010E}
\DeclareUnicodeComposite{\v}        {d}{"010F}
\DeclareUnicodeComposite{\=}        {E}{"0112}
\DeclareUnicodeComposite{\=}        {e}{"0113}
\DeclareUnicodeComposite{\u}        {E}{"0114}
\DeclareUnicodeComposite{\u}        {e}{"0115}
\DeclareUnicodeComposite{\.}        {E}{"0116}
\DeclareUnicodeComposite{\.}        {e}{"0117}
\DeclareUnicodeComposite{\k}        {E}{"0118}
\DeclareUnicodeComposite{\k}        {e}{"0119}
\DeclareUnicodeComposite{\v}        {E}{"011A}
\DeclareUnicodeComposite{\v}        {e}{"011B}
\DeclareUnicodeComposite{\^}        {G}{"011C}
\DeclareUnicodeComposite{\^}        {g}{"011D}
\DeclareUnicodeComposite{\u}        {G}{"011E}
```

```
\DeclareUnicodeComposite{\u}             {g}{"011F}
\DeclareUnicodeComposite{\.}             {G}{"0120}
\DeclareUnicodeComposite{\.}             {g}{"0121}
\DeclareUnicodeComposite{\c}             {G}{"0122}
\DeclareUnicodeComposite{\c}             {g}{"0123}
\DeclareUnicodeComposite{\^}             {H}{"0124}
\DeclareUnicodeComposite{\^}             {h}{"0125}
\DeclareUnicodeComposite{\~}             {I}{"0128}
\DeclareUnicodeComposite{\~}             \i {"0129}
\DeclareUnicodeComposite{\~}             {i}{"0129}
\DeclareUnicodeComposite{\=}             {I}{"012A}
\DeclareUnicodeComposite{\=}             \i {"012B}
\DeclareUnicodeComposite{\=}             {i}{"012B}
\DeclareUnicodeComposite{\u}             {I}{"012C}
\DeclareUnicodeComposite{\u}             \i {"012D}
\DeclareUnicodeComposite{\u}             {i}{"012D}
\DeclareUnicodeComposite{\k}             {I}{"012E}
\DeclareUnicodeComposite{\k}             \i {"012F}
\DeclareUnicodeComposite{\k}             {i}{"012F}
\DeclareUnicodeComposite{\.}             {I}{"0130}
\DeclareUnicodeComposite{\^}             {J}{"0134}
\DeclareUnicodeComposite{\^}             \j {"0135}
\DeclareUnicodeComposite{\^}             {j}{"0135}
\DeclareUnicodeComposite{\c}             {K}{"0136}
\DeclareUnicodeComposite{\c}             {k}{"0137}
\DeclareUnicodeComposite{\'}             {L}{"0139}
\DeclareUnicodeComposite{\'}             {l}{"013A}
\DeclareUnicodeComposite{\c}             {L}{"013B}
\DeclareUnicodeComposite{\c}             {l}{"013C}
\DeclareUnicodeComposite{\v}             {L}{"013D}
\DeclareUnicodeComposite{\v}             {l}{"013E}
\DeclareUnicodeComposite{\'}             {N}{"0143}
\DeclareUnicodeComposite{\'}             {n}{"0144}
\DeclareUnicodeComposite{\c}             {N}{"0145}
\DeclareUnicodeComposite{\c}             {n}{"0146}
\DeclareUnicodeComposite{\v}             {N}{"0147}
\DeclareUnicodeComposite{\v}             {n}{"0148}
\DeclareUnicodeComposite{\=}             {O}{"014C}
\DeclareUnicodeComposite{\=}             {o}{"014D}
\DeclareUnicodeComposite{\u}             {O}{"014E}
\DeclareUnicodeComposite{\u}             {o}{"014F}
\DeclareUnicodeComposite{\H}             {O}{"0150}
\DeclareUnicodeComposite{\H}             {o}{"0151}
\DeclareUnicodeComposite{\'}             {R}{"0154}
\DeclareUnicodeComposite{\'}             {r}{"0155}
\DeclareUnicodeComposite{\c}             {R}{"0156}
\DeclareUnicodeComposite{\c}             {r}{"0157}
\DeclareUnicodeComposite{\v}             {R}{"0158}
\DeclareUnicodeComposite{\v}             {r}{"0159}
\DeclareUnicodeComposite{\'}             {S}{"015A}
\DeclareUnicodeComposite{\'}             {s}{"015B}
\DeclareUnicodeComposite{\^}             {S}{"015C}
\DeclareUnicodeComposite{\^}             {s}{"015D}
\DeclareUnicodeComposite{\c}             {S}{"015E}
```

```
\DeclareUnicodeComposite{\c}            {s}{"015F}
\DeclareUnicodeComposite{\v}            {S}{"0160}
\DeclareUnicodeComposite{\v}            {s}{"0161}
\DeclareUnicodeComposite{\c}            {T}{"0162}
\DeclareUnicodeComposite{\c}            {t}{"0163}
\DeclareUnicodeComposite{\v}            {T}{"0164}
\DeclareUnicodeComposite{\v}            {t}{"0165}
\DeclareUnicodeComposite{\~}            {U}{"0168}
\DeclareUnicodeComposite{\~}            {u}{"0169}
\DeclareUnicodeComposite{\=}            {U}{"016A}
\DeclareUnicodeComposite{\=}            {u}{"016B}
\DeclareUnicodeComposite{\u}            {U}{"016C}
\DeclareUnicodeComposite{\u}            {u}{"016D}
\DeclareUnicodeComposite{\r}            {U}{"016E}
\DeclareUnicodeComposite{\r}            {u}{"016F}
\DeclareUnicodeComposite{\H}            {U}{"0170}
\DeclareUnicodeComposite{\H}            {u}{"0171}
\DeclareUnicodeComposite{\k}            {U}{"0172}
\DeclareUnicodeComposite{\k}            {u}{"0173}
\DeclareUnicodeComposite{\^}            {W}{"0174}
\DeclareUnicodeComposite{\^}            {w}{"0175}
\DeclareUnicodeComposite{\^}            {Y}{"0176}
\DeclareUnicodeComposite{\^}            {y}{"0177}
\DeclareUnicodeComposite{\"}            {Y}{"0178}
\DeclareUnicodeComposite{\'}            {Z}{"0179}
\DeclareUnicodeComposite{\'}            {z}{"017A}
\DeclareUnicodeComposite{\.}            {Z}{"017B}
\DeclareUnicodeComposite{\.}            {z}{"017C}
\DeclareUnicodeComposite{\v}            {Z}{"017D}
\DeclareUnicodeComposite{\v}            {z}{"017E}
\DeclareUnicodeComposite{\v}            {A}{"01CD}
\DeclareUnicodeComposite{\v}            {a}{"01CE}
\DeclareUnicodeComposite{\v}            {I}{"01CF}
\DeclareUnicodeComposite{\v}            \i {"01D0}
\DeclareUnicodeComposite{\v}            {i}{"01D0}
\DeclareUnicodeComposite{\v}            {O}{"01D1}
\DeclareUnicodeComposite{\v}            {o}{"01D2}
\DeclareUnicodeComposite{\v}            {U}{"01D3}
\DeclareUnicodeComposite{\v}            {u}{"01D4}
\DeclareUnicodeComposite{\=}            \AE{"01E2}
\DeclareUnicodeComposite{\=}            \ae{"01E3}
\DeclareUnicodeComposite{\v}            {G}{"01E6}
\DeclareUnicodeComposite{\v}            {g}{"01E7}
\DeclareUnicodeComposite{\v}            {K}{"01E8}
\DeclareUnicodeComposite{\v}            {k}{"01E9}
\DeclareUnicodeComposite{\k}            {O}{"01EA}
\DeclareUnicodeComposite{\k}            {o}{"01EB}
\DeclareUnicodeComposite{\v}            \j {"01F0}
\DeclareUnicodeComposite{\v}            {j}{"01F0}
\DeclareUnicodeComposite{\'}            {G}{"01F4}
\DeclareUnicodeComposite{\'}            {g}{"01F5}
\DeclareUnicodeComposite{\textcommabelow}{S}{"0218}
\DeclareUnicodeComposite{\textcommabelow}{s}{"0219}
\DeclareUnicodeComposite{\textcommabelow}{T}{"021A}
```

```
\DeclareUnicodeComposite{\textcommabelow}{t}{"021B}
\DeclareUnicodeComposite{\=}              {Y}{"0232}
\DeclareUnicodeComposite{\=}              {y}{"0232}
\DeclareUnicodeComposite{\.}              {B}{"1E02}
\DeclareUnicodeComposite{\.}              {b}{"1E03}
\DeclareUnicodeComposite{\d}              {B}{"1E04}
\DeclareUnicodeComposite{\d}              {b}{"1E05}
\DeclareUnicodeComposite{\d}              {D}{"1E0C}
\DeclareUnicodeComposite{\d}              {d}{"1E0D}
\DeclareUnicodeComposite{\=}              {G}{"1E20}
\DeclareUnicodeComposite{\=}              {g}{"1E21}
\DeclareUnicodeComposite{\d}              {H}{"1E24}
\DeclareUnicodeComposite{\d}              {h}{"1E25}
\DeclareUnicodeComposite{\d}              {K}{"1E32}
\DeclareUnicodeComposite{\d}              {k}{"1E33}
\DeclareUnicodeComposite{\d}              {L}{"1E36}
\DeclareUnicodeComposite{\d}              {l}{"1E37}
\DeclareUnicodeComposite{\d}              {M}{"1E42}
\DeclareUnicodeComposite{\d}              {m}{"1E43}
\DeclareUnicodeComposite{\d}              {N}{"1E46}
\DeclareUnicodeComposite{\d}              {n}{"1E47}
\DeclareUnicodeComposite{\d}              {R}{"1E5A}
\DeclareUnicodeComposite{\d}              {r}{"1E5B}
\DeclareUnicodeComposite{\d}              {S}{"1E62}
\DeclareUnicodeComposite{\d}              {s}{"1E63}
\DeclareUnicodeComposite{\d}              {T}{"1E6C}
\DeclareUnicodeComposite{\d}              {t}{"1E6D}
\DeclareUnicodeComposite{\d}              {V}{"1E7E}
\DeclareUnicodeComposite{\d}              {v}{"1E7F}
\DeclareUnicodeComposite{\d}              {W}{"1E88}
\DeclareUnicodeComposite{\d}              {w}{"1E89}
\DeclareUnicodeComposite{\d}              {Z}{"1E92}
\DeclareUnicodeComposite{\d}              {z}{"1E93}
\DeclareUnicodeComposite{\d}              {A}{"1EA0}
\DeclareUnicodeComposite{\d}              {a}{"1EA1}
\DeclareUnicodeComposite{\d}              {E}{"1EB8}
\DeclareUnicodeComposite{\d}              {e}{"1EB9}
\DeclareUnicodeComposite{\d}              {I}{"1ECA}
\DeclareUnicodeComposite{\d}              {i}{"1ECB}
\DeclareUnicodeComposite{\d}              {O}{"1ECC}
\DeclareUnicodeComposite{\d}              {o}{"1ECD}
\DeclareUnicodeComposite{\d}              {U}{"1EE4}
\DeclareUnicodeComposite{\d}              {u}{"1EE5}
\DeclareUnicodeComposite{\d}              {Y}{"1EF4}
\DeclareUnicodeComposite{\d}              {y}{"1EF5}
```

⟨/TU⟩

21 Package files

This file now also contains some packages that provide access to the more specialised encodings.

21.1 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.

```
⟨*package⟩
```

Here we define a macro that extends the `\@uclclist` if needed and afterwards turns itself in a noop.

```
\def\update@uclc@with@cyrillic{%
 \expandafter\def\expandafter\@uclclist\expandafter
  {\@uclclist
  \cyra\CYRA\cyrabhch\CYRABHCH\cyrabhchdsc\CYRABHCHDSC\cyrabhdze
  \CYRABHDZE\cyrabhha\CYRABHHA\cyrae\CYRAE\cyrb\CYRB\cyrbyus
  \CYRBYUS\cyrc\CYRC\cyrch\CYRCH\cyrchldsc\CYRCHLDSC\cyrchrdsc
  \CYRCHRDSC\cyrchvcrs\CYRCHVCRS\cyrd\CYRD\cyrdelta\CYRDELTA
  \cyrdje\CYRDJE\cyrdze\CYRDZE\cyrdzhe\CYRDZHE\cyre\CYRE\cyreps
  \CYREPS\cyrerev\CYREREV\cyrery\CYRERY\cyrf\CYRF\cyrfita
  \CYRFITA\cyrg\CYRG\cyrgdsc\CYRGDSC\cyrgdschcrs\CYRGDSCHCRS
  \cyrghcrs\CYRGHCRS\cyrghk\CYRGHK\cyrgup\CYRGUP\cyrh\CYRH
  \cyrhdsc\CYRHDSC\cyrhhcrs\CYRHHCRS\cyrhhk\CYRHHK\cyrhrdsn
  \CYRHRDSN\cyri\CYRI\cyrie\CYRIE\cyrii\CYRII\cyrishrt\CYRISHRT
  \cyrishrtdsc\CYRISHRTDSC\cyrizh\CYRIZH\cyrje\CYRJE\cyrk\CYRK
  \cyrkbeak\CYRKBEAK\cyrkdsc\CYRKDSC\cyrkhcrs\CYRKHCRS\cyrkhk
  \CYRKHK\cyrkvcrs\CYRKVCRS\cyrl\CYRL\cyrldsc\CYRLDSC\cyrlhk
  \CYRLHK\cyrlje\CYRLJE\cyrm\CYRM\cyrmdsc\CYRMDSC\cyrmhk\CYRMHK
  \cyrn\CYRN\cyrndsc\CYRNDSC\cyrng\CYRNG\cyrnhk\CYRNHK\cyrnje
  \CYRNJE\cyrnlhk\CYRNLHK\cyro\CYRO\cyrotld\CYROTLD\cyrp\CYRP
  \cyrphk\CYRPHK\cyrq\CYRQ\cyrr\CYRR\cyrrdsc\CYRRDSC\cyrrhk
  \CYRRHK\cyrrtick\CYRRTICK\cyrs\CYRS\cyrsacrs\CYRSACRS
  \cyrschwa\CYRSCHWA\cyrsdsc\CYRSDSC\cyrsemisftsn\CYRSEMISFTSN
  \cyrsftsn\CYRSFTSN\cyrsh\CYRSH\cyrshch\CYRSHCH\cyrshha\CYRSHHA
  \cyrt\CYRT\cyrtdsc\CYRTDSC\cyrtetse\CYRTETSE\cyrtshe\CYRTSHE
  \cyru\CYRU\cyrushrt\CYRUSHRT\cyrv\CYRV\cyrw\CYRW\cyry\CYRY
  \cyrya\CYRYA\cyryat\CYRYAT\cyryhcrs\CYRYHCRS\cyryi\CYRYI\cyryo
  \CYRYO\cyryu\CYRYU\cyrz\CYRZ\cyrzdsc\CYRZDSC\cyrzh\CYRZH
  \cyrzhdsc\CYRZHDSC}%
 \let\update@uclc@with@cyrillic\relax
}
```

Here we process each option:

```
\DeclareOption*{%
   \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 `@tempswa` to true and we always load if we run in an older format (or rather in a rollback situation).

```
   \@tempswafalse
   \@ifl@t@r\fmtversion{2020/02/02}%
       {\expandafter\ifx\csname T@\CurrentOption\endcsname\relax
         \@tempswatrue\fi}%
       {\@tempswatrue}%
```

Load if necessary:

```
   \if@tempswa
     \edef\reserved@f{%
       \lowercase{\def\noexpand\reserved@f{\CurrentOption enc.def}}}%
     \reserved@f
     \InputIfFileExists\reserved@f
          {}{\PackageError{fontenc}%
           {Encoding file `\reserved@f' not found.%
            \MessageBreak
             You might have misspelt the name of the encoding}%
           {Necessary code for this encoding was not
            loaded.\MessageBreak
            Thus calling the encoding later on will
            produce further error messages.}}%
    \let\reserved@f\relax
```

In case the current encoding is one of a list of known cyrillic ones we extend the `\@uclclist`:

```
    \expandafter\in@\expandafter{\CurrentOption}%
                                {T2A,T2B,T2C,X2,LCY,OT2}%
    \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.

```
       \expandafter\in@\expandafter\cyra\expandafter
                               {\@uclclist}%
       \ifin@
       \else
         \update@uclc@with@cyrillic
       \fi
    \fi
   \fi
}
\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:

```
%\fontencoding\encodingdefault\selectfont
```

So instead we do this here:

```
\usefont\encodingdefault\familydefault\seriesdefault\shapedefault
```

To save some space we get rid of the macro extending the `\@uclclist` (might have happened already).

```
\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.

```
\let\@elt\relax
\xdef\@fontenc@load@list{\@fontenc@load@list
  \@elt{\csname opt@fontenc.sty\endcsname}}
```

```
\global\expandafter\let\csname ver@fontenc.sty\endcsname\relax
\global\expandafter\let\csname opt@fontenc.sty\endcsname\relax
\global\let\@ifl@ter@@\@ifl@ter
\def\@ifl@ter#1#2#3#4#5{\global\let\@ifl@ter\@ifl@ter@@}
⟨/package⟩
```

File m

ltcounts.dtx

22 Counters and Lengths

Commands for defining and using counters. This file defines:

`\newcounter` To define a new counter.

`\setcounter` To set the value of counters.

`\addtocounter` Increase the counter `#1` by the number `#2`.

`\stepcounter` Increase a counter by one.

`\refstepcounter` Increase a counter by one, also setting the value used by `\label`.

`\value` For accessing the value of the counter as a TeX number (as opposed to `\the`⟨*counter*⟩ which expands to the *printed* representation of ⟨*counter*⟩)

`\arabic` `\arabic{`⟨*counter*⟩`}`: 1, 2, 3, ...

`\roman` `\roman{`⟨*counter*⟩`}`: i, ii, iii, ...

`\Roman` `\Roman{`⟨*counter*⟩`}`: I, II, III, ...

`\alph` `\alph{`⟨*counter*⟩`}`: a, b, c, ...

`\Alph` `\Alph{`⟨*counter*⟩`}`: A, B, C, ...

`\fnsymbol` `\fnsymbol{`⟨*counter*⟩`}`: $*$, †, ‡, ...

`\counterwithin` `\counterwithin{`⟨*counter*⟩`}{`⟨*within-counter*⟩`}`: Resets ⟨*counter*⟩ whenever ⟨*within-counter*⟩ is stepped. Also redefines `\the`⟨*counter*⟩ command to produce `\the`⟨*within-counter*⟩`.\arabic{`⟨*counter*⟩`}`. Star form omits redefining the print representation.

`\counterwithout` `\counterwithout{`⟨*counter*⟩`}{`⟨*within-counter*⟩`}`: Removes ⟨*counter*⟩ from the reset list of ⟨*within-counter*⟩. Also redefines `\the`⟨*counter*⟩ command to produce `\arabic{`⟨*counter*⟩`}`. Star form omits redefining the print representation.

```
⟨*2ekernel⟩
```

22.1 Environment Counter Macros

An environment foo has an associated counter defined by the following control sequences:

`\c@foo` Contains the counter's numerical value. It is defined by `\newcount\foocounter`.

`\thefoo` Macro that expands to the printed value of `\foocounter`. For example, if sections are numbered within chapters, and section headings look like
Section II-3. The Nature of Counters
then `\thesection` might be defined by:

```
\def\thesection
    {\@Roman{\c@chapter}-\@arabic{\c@section}}
```

`\p@foo` Macro that expands to a printed 'reference prefix' of counter foo. Any `\ref` to a value created by counter foo will produce the expansion of `\p@foo\thefoo` when the `\label` command is executed. See file `ltxref.dtx` for an extension of this mechanism.

`\cl@foo` List of counters to be reset when foo stepped. Has format `\@elt{countera}\@elt{counterb}\@elt{counterc}`.

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*⟩.

```
\def\setcounter#1#2{%
  \@ifundefined{c@#1}%
    {\@nocounterr{#1}}%
    {\global\csname c@#1\endcsname#2\relax}}
```

`\addtocounter` `\addtocounter{`⟨*foo*⟩`}{`⟨*val*⟩`}` Globally increments `\foocounter` by ⟨*val*⟩.

```
\def\addtocounter#1#2{%
  \@ifundefined{c@#1}%
    {\@nocounterr{#1}}%
    {\global\advance\csname c@#1\endcsname #2\relax}}
```

`\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.

```
\def\newcounter#1{%
  \expandafter\@ifdefinable \csname c@#1\endcsname
    {\@definecounter{#1}}%
  \@ifnextchar[{\@newctr{#1}}{}}
```

`\value` `\value{`⟨*ctr*⟩`}` produces the value of counter ⟨*ctr*⟩, for use with a `\setcounter` or `\addtocounter` command.

```
\def\value#1{\csname c@#1\endcsname}
```

`\@newctr`

```
\def\@newctr#1[#2]{%
  \@ifundefined{c@#2}{\@nocounterr{#2}}{\@addtoreset{#1}{#2}}}
```

`\stepcounter` `\stepcounter`foo Globally increments counter `\c@FOO` and resets all subsidiary counters.

```
\def\stepcounter#1{%
```

```
  \addtocounter{#1}\@ne
  \begingroup
    \let\@elt\@stpelt
    \csname cl@#1\endcsname
  \endgroup}
```

\@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.

```
⟨/2ekernel⟩
⟨latexrelease⟩\IncludeInRelease{2015/01/01}{\@stpelt}
⟨latexrelease⟩                              {Reset nested counters}%
⟨*2ekernel | latexrelease⟩
\def\@stpelt#1{\global\csname c@#1\endcsname \m@ne\stepcounter{#1}}%
⟨latexrelease⟩\EndIncludeInRelease
⟨/2ekernel | latexrelease⟩
⟨latexrelease⟩\IncludeInRelease{0000/00/00}{\@stpelt}
⟨latexrelease⟩                              {Reset nested counters}%%
⟨latexrelease⟩\def\@stpelt#1{\global\csname c@#1\endcsname \z@}%
⟨latexrelease⟩\EndIncludeInRelease
⟨*2ekernel⟩
```

\cl@@ckpt

```
\def\cl@@ckpt{\@elt{page}}
```

\@definecounter

```
\def\@definecounter#1{\expandafter\newcount\csname c@#1\endcsname
     \setcounter{#1}\z@
     \global\expandafter\let\csname cl@#1\endcsname\@empty
     \@addtoreset{#1}{@ckpt}%
     \global\expandafter\let\csname p@#1\endcsname\@empty
     \expandafter
     \gdef\csname the#1\expandafter\endcsname\expandafter
          {\expandafter\@arabic\csname c@#1\endcsname}}
```

\@addtoreset

```
\def\@addtoreset#1#2{\expandafter\@cons\csname cl@#2\endcsname {{#1}}}
```

```
⟨/2ekernel⟩
```

\@removefromreset

```
⟨latexrelease⟩\IncludeInRelease{2018-04-01}
⟨latexrelease⟩                  {\@removefromreset}{Add interfaces}%
⟨*2ekernel | latexrelease⟩
```

```
\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.

```
  \@ifundefined{c@#2}\relax
  {\begingroup
     \expandafter\let\csname c@#1\endcsname\@removefromreset
     \def\@elt##1{%
       \expandafter\ifx\csname c@##1\endcsname\@removefromreset
       \else
         \noexpand\@elt{##1}%
       \fi}%
     \expandafter\xdef\csname cl@#2\endcsname
      {\csname cl@#2\endcsname}%
    \endgroup}}
```

\@ifbothcounters Test if arg #1 and #2 are counters and if so execute #3.

```
\def\@ifbothcounters#1#2#3{%
  \@ifundefined{c@#1}{\@nocounterr{#1}}%
    {% else counter is defined
     \@ifundefined{c@#2}{\@nocounterr{#2}}%
       {% else both counter and within  are defined
        #3}}}
```

\counterwithout

```
\def\counterwithout {\@ifstar\counterwithout@s\counterwithout@x}
```

```
\def\counterwithout@s#1#2{%
  \@ifbothcounters{#1}{#2}{\@removefromreset{#1}{#2}}}
```

```
\def\counterwithout@x#1#2{%
  \@ifbothcounters{#1}{#2}%
      {\@removefromreset{#1}{#2}%
       \expandafter
       \gdef\csname the#1\expandafter\endcsname\expandafter
            {\expandafter
             \@arabic\csname c@#1\endcsname}}}
```

\counterwithin

```
\def\counterwithin{\@ifstar\counterwithin@s\counterwithin@x}
```

```
\def\counterwithin@s#1#2{%
  \@ifbothcounters{#1}{#2}{\@addtoreset{#1}{#2}}}
```

```
\def\counterwithin@x#1#2{%
  \@ifbothcounters{#1}{#2}%
      {\@addtoreset{#1}{#2}%
       \expandafter
       \gdef\csname the#1\expandafter\endcsname\expandafter
            {\csname the#2\expandafter\endcsname\expandafter
             .\expandafter
             \@arabic\csname c@#1\endcsname}}}
```

```
⟨/2ekernel | latexrelease⟩
⟨latexrelease⟩\EndIncludeInRelease
⟨latexrelease⟩\IncludeInRelease{0000-00-00}
⟨latexrelease⟩                 {\@removefromreset}{Add interfaces}%
⟨latexrelease⟩\let \@removefromreset \undefined
⟨latexrelease⟩\let \@ifbothcounters  \undefined
⟨latexrelease⟩\let \counterwithout   \undefined
⟨latexrelease⟩\let \counterwithout@s \undefined
⟨latexrelease⟩\let \counterwithout@x \undefined
⟨latexrelease⟩\let \counterwithin    \undefined
⟨latexrelease⟩\let \counterwithin@s  \undefined
⟨latexrelease⟩\let \counterwithin@x  \undefined
```

```
⟨latexrelease⟩\EndIncludeInRelease
⟨*2ekernel⟩
```

Numbering commands for definitions of `\theCOUNTER` and `\list` arguments. All commands can now be used in text and math mode.

`\arabic` Representation of⟨*counter*⟩ as arabic numerals. Changed 29 Apr 86 to make it print the obvious thing it COUNTER not positive.

```
\def\arabic#1{\expandafter\@arabic\csname c@#1\endcsname}
```

`\roman` Representation of ⟨*counter*⟩ as lower-case Roman numerals.

```
\def\roman#1{\expandafter\@roman\csname c@#1\endcsname}
```

`\Roman` Representation of ⟨*counter*⟩ as upper-case Roman numerals.

```
\def\Roman#1{\expandafter\@Roman\csname c@#1\endcsname}
```

`\alph` Representation of ⟨*counter*⟩ as a lower-case letter: 1 = a, 2 = b, etc.

```
\def\alph#1{\expandafter\@alph\csname c@#1\endcsname}
```

`\Alph` Representation of ⟨*counter*⟩ as an upper-case letter: 1 = A, 2 = B, etc.

```
\def\Alph#1{\expandafter\@Alph\csname c@#1\endcsname}
```

`\fnsymbol` Representation of ⟨*COUNTER*⟩ as a footnote symbol: 1 = ∗, 2 = †, etc.

```
\def\fnsymbol#1{\expandafter\@fnsymbol\csname c@#1\endcsname}
```

`\@arabic` `\@arabic\FOOcounter` Representation of `\FOOcounter` as arabic numerals.

```
\def\@arabic#1{\number #1}  %% changed 29 Apr 86
```

`\@roman` `\@roman\FOOcounter` Representation of `\FOOcounter` as lower-case Roman numerals.

```
\def\@roman#1{\romannumeral #1}
```

`\@Roman` `\@Roman\FOOcounter` Representation of `\FOOcounter` as upper-case Roman numerals.

```
\def\@Roman#1{\expandafter\@slowromancap\romannumeral #1@}
```

`\@slowromancap` Fully expandable macro to change a roman number to uppercase.

```
\def\@slowromancap#1{\ifx @#1% then terminate
     \else
       \if i#1I\else\if v#1V\else\if x#1X\else\if l#1L\else\if
       c#1C\else\if d#1D\else \if m#1M\else#1\fi\fi\fi\fi\fi\fi\fi
       \expandafter\@slowromancap
     \fi
}
```

`\@alph` `\@alph\FOOcounter` Representation of `\FOOcounter` as a lower-case letter: 1 = a, 2 = b, etc.

```
\def\@alph#1{%
  \ifcase#1\or a\or b\or c\or d\or e\or f\or g\or h\or i\or j\or
   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
    y\or z\else\@ctrerr\fi}
```

`\@Alph` `\@Alph\FOOcounter` Representation of `\FOOcounter` as an upper-case letter: 1 = A, 2 = B, etc.

```
\def\@Alph#1{%
  \ifcase#1\or A\or B\or C\or D\or E\or F\or G\or H\or I\or J\or
   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
    Y\or Z\else\@ctrerr\fi}
```

`\@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.

```
⟨/2ekernel⟩
⟨latexrelease⟩\IncludeInRelease{2015/01/01}{\@fnsymbol}{Use \TexOrMath}%
⟨*2ekernel | latexrelease⟩
\def\@fnsymbol#1{%
   \ifcase#1\or \TextOrMath\textasteriskcentered *\or
   \TextOrMath \textdagger \dagger\or
   \TextOrMath \textdaggerdbl \ddagger \or
   \TextOrMath \textsection  \mathsection\or
   \TextOrMath \textparagraph \mathparagraph\or
   \TextOrMath \textbardbl \|\or
   \TextOrMath {\textasteriskcentered\textasteriskcentered}{**}\or
   \TextOrMath {\textdagger\textdagger}{\dagger\dagger}\or
   \TextOrMath {\textdaggerdbl\textdaggerdbl}{\ddagger\ddagger}\else
   \@ctrerr \fi
}%
⟨/2ekernel | latexrelease⟩
⟨latexrelease⟩\EndIncludeInRelease
⟨latexrelease⟩\IncludeInRelease{0000/00/00}{\@fnsymbol}{Use \TexOrMath}%
⟨latexrelease⟩\def\@fnsymbol#1{\ensuremath{%
⟨latexrelease⟩   \ifcase#1\or *\or \dagger\or \ddagger\or \mathsection\or
⟨latexrelease⟩      \mathparagraph\or \|\or **\or \dagger\dagger
⟨latexrelease⟩      \or \ddagger\ddagger \else\@ctrerr\fi}}%
⟨latexrelease⟩\EndIncludeInRelease
⟨*2ekernel⟩
```

`\TextOrMath` When using regular TEX, 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 wil have the disadvantage of choosing the wrong branch if appearing at the beginning of an alignment cell.

However, users of eTeX 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 eTeX but making sure not to permanently turn `\protected` into `\relax`.

```
⟨/2ekernel⟩
⟨latexrelease⟩\IncludeInRelease{2015/01/01}{\TextOrMath}{\TextOrMath}%
⟨*2ekernel | latexrelease⟩
\begingroup\expandafter\expandafter\expandafter\endgroup
\expandafter\ifx\csname protected\endcsname\relax
```

In case of ordinary TeX 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.

```
\DeclareRobustCommand\TextOrMath{%
  \ifmmode  \expandafter\@secondoftwo
  \else     \expandafter\@firstoftwo  \fi}
\protected@edef\TextOrMath#1#2{\TextOrMath{#1}{#2}}
\else
```

For eTeX the situation is similar. The robust macro is a hidden one so that we again avoid problems of gobbling spaces in the input.

```
\protected\expandafter\def\csname TextOrMath\space\endcsname{%
  \ifmmode  \expandafter\@secondoftwo
  \else     \expandafter\@firstoftwo  \fi}
\edef\TextOrMath#1#2{%
  \expandafter\noexpand\csname TextOrMath\space\endcsname
  {#1}{#2}}
\fi
⟨/2ekernel | latexrelease⟩
⟨latexrelease⟩\EndIncludeInRelease
⟨latexrelease⟩\IncludeInRelease{0000/00/00}{\TextOrMath}{\TextOrMath}%
⟨latexrelease⟩\let\TextOrMath\@undefined
⟨latexrelease⟩\EndIncludeInRelease
⟨*2ekernel⟩
```

```
⟨/2ekernel⟩
```

File n
ltlength.dtx

23 Lengths

\newlength
\setlength
\addtolength
\settowidth
\settoheight
\settodepth

Declare #1 to be a new length command.

Set the length command, #1, to the value #2.

Increase the value of the length command, #1, by the value #2.

Set the length, #1 to the width of a box containing #2.

Set the length, #1 to the height of a box containing #2.

Set the length, #1 to the depth of a box containing #2.

```
⟨*2ekernel⟩
\message{lengths,}
```

\newlength

```
\def\newlength#1{\@ifdefinable#1{\newskip#1}}
```

\setlength

```
⟨/2ekernel⟩
⟨latexrelease⟩\IncludeInRelease{2015/01/01}%
⟨latexrelease⟩                 {\setlength}{Using \setlength with \dimen0}%
⟨*2ekernel | latexrelease⟩
```

```
\def\setlength#1#2{#1 #2\relax}
⟨/2ekernel | latexrelease⟩
⟨latexrelease⟩\EndIncludeInRelease
⟨latexrelease⟩\IncludeInRelease{0000/00/00}%
⟨latexrelease⟩                 {\setlength}{Using \setlength with \dimen0}%
⟨latexrelease⟩\def\setlength#1#2{#1#2\relax}
⟨latexrelease⟩\EndIncludeInRelease
⟨*2ekernel⟩
```

\addtolength

\relax added 24 Mar 86

```
\def\addtolength#1#2{\advance#1 #2\relax}
```

\settoheight
\settodepth
\settowidth
\@settodim

The obvious analogs of \settowidth.

```
\def\@settodim#1#2#3{\setbox\@tempboxa\hbox{{#3}}#2#1\@tempboxa
```

Clear the memory afterwards (which might be a lot).

```
        \setbox\@tempboxa\box\voidb@x}
\DeclareRobustCommand\settoheight{\@settodim\ht}
\DeclareRobustCommand\settodepth {\@settodim\dp}
\DeclareRobustCommand\settowidth {\@settodim\wd}
```

\@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 calulating a dimension.

```
\def\@settopoint#1{\divide#1\p@\multiply#1\p@}
⟨/2ekernel⟩
```

File o
ltfssbas.dtx

This file contains the main implementation of the 'low level' font selection commands. See other parts of the LaTeX distribution, or *The LaTeX Companion* for higher level documentation of the LaTeX '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.

24 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.

```
⟨*2ekernel⟩
\def\@nomath#1{\relax\ifmmode
   \@font@warning{Command \noexpand#1invalid in math mode}\fi}
```

\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 TeX from scanning too far in certain situations.

```
\gdef\no@alphabet@error#1{\relax \ifmmode
    \@latex@error{Math\space alphabet\space identifier\space
          \noexpand#1is\space undefined\space in\space math\space
           version\space `\math@version'}%
        {Your\space requested\space math\space alphabet\space
         is\space undefined\space in\space the\space current\space
          math\space version.^^JCheck\space the\space spelling\space
          or\space use\space the\space \noexpand\SetMathAlphabet\space
          command.}
     \fi}
```

\new@mathgroup \mathgroup We also give a new name to `\newfam` and `\fam` to avoid verbal confusion (see the introduction).[2]

```
%\def\new@mathgroup{\alloc@8\mathgroup\chardef\sixt@@n}
\let\mathgroup\fam
%\let\newfam\new@mathgroup
\@onlypreamble\new@mathgroup
```

[2] For 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.

25 Macros for setting up the tables

\DeclareFontShape

The macro `\DeclareFontShape` takes 6 arguments:

```
\def\DeclareFontShape{\begingroup
```

First we restore the catcodes of all characters used in the syntax.

```
    \nfss@catcodes
```

We use `\expandafter` `\endgroup` to restore catcode in case something goes wrong with the argument parsing (suggested by Tim Van Zandt)

```
    \expandafter\endgroup
    \DeclareFontShape@}
```

\DeclareFontShape@

```
⟨/2ekernel⟩
⟨*2ekernel | latexrelease⟩
⟨latexrelease⟩\IncludeInRelease{2020/02/02}%
⟨latexrelease⟩                {\DeclareFontShape@}{Maybe drop one m}%
\def\DeclareFontShape@#1#2#3#4#5#6{%
    \expandafter\ifx\csname #1+#2\endcsname\relax
     \@latex@error{Font family `#1+#2' unknown}\@eha
    \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.

```
     \edef\reserved@a{#3}%
     \expandafter\series@maybe@drop@one@m\expandafter{#3}\reserved@b
     \ifx\reserved@a\reserved@b\else
       \@latex@warning{Font shape declaration has incorrect series
         value `#3'.\MessageBreak It should not contain an `m'!
         Please correct it.\MessageBreak Found}%
     \fi
     \expandafter
       \xdef\csname#1/#2/\reserved@b/#4\endcsname
                {\expandafter\noexpand\csname #5\endcsname}%
%
```

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.

```
     \def\reserved@a{#6}%
     \global
     \expandafter\let\csname#5\expandafter\endcsname
        \ifx\reserved@a\@empty
          \@empty
        \else
          \reserved@a
        \fi
   \fi
  }
```

```
⟨/2ekernel | latexrelease⟩
⟨latexrelease⟩\EndIncludeInRelease
⟨latexrelease⟩\IncludeInRelease{0000/00/00}%
⟨latexrelease⟩                 {\DeclareFontShape@}{Maybe drop one m}%
⟨latexrelease⟩
⟨latexrelease⟩\def\DeclareFontShape@#1#2#3#4#5#6{%
⟨latexrelease⟩   \expandafter\ifx\csname #1+#2\endcsname\relax
⟨latexrelease⟩     \@latex@error{Font family ‘#1+#2’ unknown}\@eha
⟨latexrelease⟩   \else
⟨latexrelease⟩     \expandafter
⟨latexrelease⟩       \xdef\csname#1/#2/#3/#4\endcsname{\expandafter\noexpand
⟨latexrelease⟩                                   \csname #5\endcsname}%
⟨latexrelease⟩     \def\reserved@a{#6}%
⟨latexrelease⟩     \global
⟨latexrelease⟩     \expandafter\let\csname#5\expandafter\endcsname
⟨latexrelease⟩        \ifx\reserved@a\@empty
⟨latexrelease⟩          \@empty
⟨latexrelease⟩        \else
⟨latexrelease⟩          \reserved@a
⟨latexrelease⟩        \fi
⟨latexrelease⟩   \fi
⟨latexrelease⟩  }
⟨latexrelease⟩\EndIncludeInRelease
⟨*2ekernel⟩
```

\DeclareFixedFont

Define a direct font switch that avoids all overhead.

```
\def\DeclareFixedFont#1#2#3#4#5#6{%
   \begingroup
      \math@fontsfalse
      \every@math@size{}%
      \fontsize{#6}\z@
      \usefont{#2}{#3}{#4}{#5}%
      \global\expandafter\let\expandafter#1\the\font
   \endgroup
  }
```

\do@subst@correction

```
\def\do@subst@correction{%
       \xdef\subst@correction{%
          \font@name
          \global\expandafter\font
            \csname \curr@fontshape/\f@size\endcsname
            \noexpand\fontname\font
           \relax}%
```

Calling \subst@correction after the current group means calling it after we have loaded the substitution font which is done inside a group.

```
       \aftergroup\subst@correction
}
```

\DeclareFontFamily

```
\def\DeclareFontFamily#1#2#3{%
```

If we want fast checking for the encoding scheme we can just check for \T@.. being defined.

```
% \@tempswafalse
% \def\reserved@b{#1}%
% \def\cdp@elt##1##2##3##4{\def\reserved@c{##1}%
%      \ifx\reserved@b\reserved@c \@tempswatrue\fi}%
% \cdp@list
% \if@tempswa
 \@ifundefined{T@#1}%
    {%
     \@latex@error{Encoding scheme  `#1' unknown}\@eha
    }%
    {%
```

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`.

```
     \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 a doing a `\let` to `\reserved@a` — the latter would blow one extra memory location rather then reusing the one from `\@empty`.

```
     \global
     \expandafter\let\csname #1+#2\expandafter\endcsname
            \ifx \reserved@a\@empty
              \@empty
            \else \reserved@a
            \fi
    }%
}
```

`\cdp@list` We initialize the code page list to be empty.

```
\let\cdp@list\@empty
\@onlypreamble\cdp@list
```

`\cdp@elt`

```
\let\cdp@elt\relax
\@onlypreamble\cdp@elt
```

`\DeclareFontEncoding`

```
\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.

```
   \begingroup
   \nfss@catcodes
   \expandafter\endgroup
   \DeclareFontEncoding@}
\@onlypreamble\DeclareFontEncoding
```

```
\def\DeclareFontEncoding@#1#2#3{%
  \expandafter
  \ifx\csname T@#1\endcsname\relax
     \def\cdp@elt{\noexpand\cdp@elt}%
     \xdef\cdp@list{\cdp@list\cdp@elt{#1}%
```

```
                    {\default@family}{\default@series}%
                    {\default@shape}}%
```

To support encoding dependent commands (like accents) we initialise the command \⟨*encoding*⟩-cmd to be \@changed@cmd. (See ltoutenc.dtx for details.)

```
     \expandafter\let\csname#1-cmd\endcsname\@changed@cmd
  \else
     \@font@info{Redeclaring font encoding #1}%
  \fi
  \global\@namedef{T@#1}{#2}%
  \global\@namedef{M@#1}{\default@M#3}%
```

Keep a record of the last encoding being declared:

```
  \xdef\LastDeclaredEncoding{#1}%
  }
\@onlypreamble\DeclareFontEncoding@
```

\LastDeclaredEncoding The last encoding being declared by \DeclareFontEncoding.

```
\def\LastDeclaredEncoding{}
```

\DeclareFontSubstitution

```
\def\DeclareFontSubstitution#1#2#3#4{%
  \expandafter
  \ifx\csname T@#1\endcsname\relax
    \@latex@error{Encoding scheme  `#1' unknown}\@eha
  \else
    \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.

```
       \edef\reserved@a{#1}%
       \toks@{}%
       \def\cdp@elt##1##2##3##4{%
          \def\reserved@b{##1}%
          \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.

```
             \addto@hook\toks@{\cdp@elt{##1}{#2}{#3}{#4}}%
          \else
```

If \reserved@a and \reserved@b differ then we simply copy from the old list to the new.

```
             \addto@hook\toks@{\cdp@elt{##1}{##2}{##3}{##4}}%
          \fi}%
        \cdp@list
        \xdef\cdp@list{\the\toks@}%
    \endgroup
    \global
    \@namedef{D@#1}{%
           \def\default@family{#2}%
           \def\default@series{#3}%
           \def\default@shape{#4}%
           }%
  \fi
 }
\@onlypreamble\DeclareFontSubstitution
```

\DeclareFontEncodingDefaults

```
\def\DeclareFontEncodingDefaults#1#2{%
  \ifx\relax#1\else
    \ifx\default@T\@empty\else
      \@font@info{Overwriting encoding scheme text defaults}%
    \fi
    \gdef\default@T{#1}%
  \fi
  \ifx\relax#2\else
    \ifx\default@M\@empty\else
      \@font@info{Overwriting encoding scheme math defaults}%
    \fi
    \gdef\default@M{#2}%
  \fi
}
\@onlypreamble\DeclareFontEncodingDefaults
```

\default@T
\default@M

```
\let\default@T\@empty
\let\default@M\@empty
```

\DeclarePreloadSizes

```
\def\DeclarePreloadSizes#1#2#3#4#5{%
 \@ifundefined{T@#1}%
   {\@latex@error{Encoding scheme  ‘#1’ unknown}\@eha}%
   {%
```

Don’t know at the moment what this group here does!

```
   \begingroup
```

We define a macro `\reserved@f`[3] that grabs the next *size* and loads the corresponding font. This is done by delimiting `\reserved@f`’s only argument by the token , (comma).

```
    \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 TEXbook: if the argument is empty the `\if` will select the first clause and `\let \reserved@f` equal to `\relax`. (We use the `>` character here since it cannot appear in font file names.)

```
         \if>##1>%
          \let\reserved@f\relax
         \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

[3] We cannot use `\@tempa` since it is needed in `\pickup@font`.

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.

```
          \xdef\font@name{\csname#1/#2/#3/#4/##1\endcsname}%
          \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.

```
          \global\expandafter\let\font@name\relax
        \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.

```
        \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.

```
     \reserved@f#5,,%
   \endgroup
   }%
}
\@onlypreamble\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.

```
\newif\ifmath@fonts \math@fontstrue
```

\DeclareMathSizes
\DeclareMathSizes* \DeclareMathSizes takes the text size, math text size, math script size, and math scriptscript size as arguments and defines the right \S@... macro.

```
\def\DeclareMathSizes{%
  \@ifstar{\@DeclareMathSizes\math@fontsfalse}%
         {\@DeclareMathSizes{}}}
\@onlypreamble\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}.

```
⟨/2ekernel⟩
⟨latexrelease⟩\IncludeInRelease{2015/01/01}{\@DeclareMathSizes}%
⟨latexrelease⟩                 {Arbitrary units in \DeclareMathSizes}%
⟨*2ekernel | latexrelease⟩
\def\@DeclareMathSizes #1#2#3#4#5{%
  \@defaultunits\dimen@ #2pt\relax\@nnil
  \if $#3$%
    \expandafter\let\csname S@\strip@pt\dimen@\endcsname\math@fontsfalse
  \else
    \@defaultunits\dimen@ii #3pt\relax\@nnil
    \@defaultunits\@tempdima #4pt\relax\@nnil
    \@defaultunits\@tempdimb #5pt\relax\@nnil
    \toks@{#1}%
    \expandafter\xdef\csname S@\strip@pt\dimen@\endcsname{%
      \gdef\noexpand\tf@size{\strip@pt\dimen@ii}%
      \gdef\noexpand\sf@size{\strip@pt\@tempdima}%
      \gdef\noexpand\ssf@size{\strip@pt\@tempdimb}%
      \the\toks@
    }%
  \fi
}%
⟨/2ekernel | latexrelease⟩
⟨latexrelease⟩\EndIncludeInRelease
⟨latexrelease⟩\IncludeInRelease{0000/00/00}{\@DeclareMathSizes}%
⟨latexrelease⟩                {Arbitrary units in \DeclareMathSizes}%
⟨latexrelease⟩\def\@DeclareMathSizes#1#2#3#4#5{%
⟨latexrelease⟩    \@defaultunits\dimen@#2pt\relax\@nnil
⟨latexrelease⟩    \if$#3$%
⟨latexrelease⟩      \expandafter \let
⟨latexrelease⟩        \csname S@\strip@pt\dimen@\endcsname
⟨latexrelease⟩        \math@fontsfalse
⟨latexrelease⟩    \else
⟨latexrelease⟩      \expandafter \gdef
⟨latexrelease⟩      \csname S@\strip@pt\dimen@\endcsname
⟨latexrelease⟩            {\gdef\tf@size{#3}\gdef\sf@size{#4}%
⟨latexrelease⟩                             \gdef\ssf@size{#5}%
⟨latexrelease⟩             #1%
⟨latexrelease⟩                             }%
⟨latexrelease⟩    \fi}%
⟨latexrelease⟩\EndIncludeInRelease
⟨*2ekernel⟩
```

```
\@onlypreamble\@DeclareMathSizes
```

26 Selecting a new font

26.1 Macros for the user

\fontencoding
\f@encoding

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.

```
\DeclareRobustCommand\fontencoding[1]{%
    \expandafter\ifx\csname T@#1\endcsname\relax
      \@latex@error{Encoding scheme `#1' unknown}\@eha
    \else
      \edef\f@encoding{#1}%
      \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`.

```
        \let\enc@update\relax
      \else
```

If current and new encoding differ we define the macro `\enc@update` to contain all updates necessary at `\selectfont` time.

```
    \let\enc@update\@@enc@update
   \fi
  \fi
}
```

\@@enc@update

```
\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`.

```
     \expandafter
     \let
      \csname\cf@encoding -cmd\endcsname
      \@changed@cmd
```

Then we turn the one for the new encoding to `\@current@cmd` (see `ltoutenc.dtx` for further explanations).

```
     \expandafter
     \let
      \csname\f@encoding-cmd\endcsname
      \@current@cmd
```

We execute the default settings `\default@T`, followed by the one for the new encoding.

```
    \default@T
    \csname T@\f@encoding\endcsname
```

Finally we change the default substitution values, disable `\enc@update` and make `\f@encoding` officially the current encoding.

```
    \csname D@\f@encoding\endcsname
    \let\enc@update\relax
    \let\cf@encoding\f@encoding
}
```

\enc@update The default action in `\selectfont` is to do nothing.

```
\let\enc@update\relax
```

\fontfamily
\f@family
\fontseries
\f@series
\fontshape
\f@shape

```
\DeclareRobustCommand\fontfamily[1]{\edef\f@family{#1}}
```

There are now defined later (and differently).

```
%\DeclareRobustCommand\fontseries[1]{\edef\f@series{#1}}
%\DeclareRobustCommand\fontshape [1]{\edef\f@shape{#1}}
```

\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!

```
\DeclareRobustCommand\usefont[4]{\fontencoding{#1}%
```

```
  \edef\f@family{#2}%
  \edef\f@series{#3}%
  \edef\f@shape{#4}\selectfont
  \ignorespaces}
```

\linespread The command \linespread changes the current \baselinestretch by calling \set@fontsize. The values for \f@size and \f@baselineskip will be left unchanged.

```
\DeclareRobustCommand\linespread[1]
  {\set@fontsize{#1}\f@size\f@baselineskip}
```

\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.

```
\DeclareRobustCommand\fontsize[2]
  {\set@fontsize\baselinestretch{#1}{#2}}
```

\f@linespread This macro holds the current internal value for \baselinestretch.

```
\let\f@family\@empty
\let\f@series\@empty
\let\f@shape\@empty
\let\f@size\@empty
\let\f@baselineskip\@empty
\let\f@linespread\@empty
```

\cf@encoding

```
\let\f@encoding\@empty
\let\cf@encoding\@empty
```

\@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.

```
\def\@defaultunits{\afterassignment\remove@to@nnil}
```

\strip@pt
\rem@pt This macro strips the characters pt produced by using \the on a dimen register.

```
\begingroup
  \catcode`P=12
  \catcode`T=12
  \lowercase{
    \def\x{\def\rem@pt##1.##2PT{##1\ifnum##2>\z@.##2\fi}}}
  \expandafter\endgroup\x
\def\strip@pt{\expandafter\rem@pt\the}
```

\mathversion
\math@version

`\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.

```
\DeclareRobustCommand\mathversion[1]
         {\@nomath\mathversion
          \expandafter\ifx\csname mv@#1\endcsname\relax
          \@latex@error{Math version ‘#1’ is not defined}\@eha\else
          \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@currsize` to an invalid value since this will trigger the setup when the formula starts.

```
          \gdef\glb@currsize{}%
```

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 todo the setting only if the version really has changed after all. This might be interesting in case of `amstext` and `boldsymbol`.

```
          \aftergroup\glb@settings
          \fi}
```

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 LaTeX 2_ε 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
\frozen@everydisplay

New internal names for `\everymath` and `\everydisplay`.

```
\let\frozen@everymath\everymath
\let\frozen@everydisplay\everydisplay
```

\everymath
\everydisplay

Now we provide now user hooks that will be called in the frozen internals.

```
\newtoks\everymath
\newtoks\everydisplay
```

\frozen@everymath

Now we define the behaviour of the frozen hooks: first check the math setup then call the user hook.

```
\frozen@everymath = {\check@mathfonts
                     \the\everymath}
```

\frozen@everydisplay

Ditto for the display hook.

```
\frozen@everydisplay = {\check@mathfonts
                        \the\everydisplay}
```

\curr@math@size

This holds locally the current math size.

```
\let\curr@math@size\@empty
```

26.2 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.

```
\def\pickup@font{%
    \expandafter \ifx \font@name \relax
       \define@newfont
    \fi}
```

\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`

```
{\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.

```
\gdef\split@name#1/#2/#3/#4/#5\@nil{\def\f@encoding{#1}%
                                    \def\f@family{#2}%
                                    \def\f@series{#3}%
                                    \def\f@shape{#4}%
                                    \def\f@size{#5}}}
```

\curr@fontshape

Abbreviation which may get removed again for speed.

```
\def\curr@fontshape{\f@encoding/\f@family/\f@series/\f@shape}
```

\define@newfont

Now we can tackle the problem of defining a new font.

```
\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.

```
  \begingroup
    \let\typeout\@font@info
    \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`.

```
    \expandafter\expandafter\expandafter
       \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.

```
%    \expandafter\ifx
%        \csname\curr@fontshape\endcsname \relax
      \try@load@fontshape % try always
%    \fi
    \expandafter\ifx
       \csname\curr@fontshape\endcsname \relax
      \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).

```
%      \csname\curr@fontshape\endcsname
      \extract@font\fi
```

We are nearly finished and must only restore the \escapechar by closing the group.

```
  \endgroup}
```

```
\def\try@load@fontshape{%
   \expandafter
   \ifx\csname \f@encoding+\f@family\endcsname\relax
     \@font@info{Trying to load font information for
                   \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.

```
    \global\expandafter\let
       \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).

```
     \nfss@catcodes
     \let\nfss@catcodes\relax
```

For increased portability make the external filename monocase, but look for the (old style) mixed case filename if the first attempt fails.

On any monocase 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.

```
     \edef\reserved@a{%
       \lowercase{%
         \noexpand\InputIfFileExists{\f@encoding\f@family.fd}}}%
   \reserved@a\relax
          {\@input@{\f@encoding\f@family.fd}}%
   \fi}
```

\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.

```
\def\nfss@catcodes{%
```

We start by making `@` a letter and ignoring all blanks and newlines.

```
     \makeatletter
     \catcode`\ 9%
     \catcode`\^^I9%
     \catcode`\^^M9%
```

Then we set up `\`, `{`, `}`, `#` and `%` in case an `.fd` file is loaded during a verbatim environment.

```
     \catcode`\\\z@
     \catcode`\{\@ne
     \catcode`\}\tw@
     \catcode`\#6%
     \catcode`\^7%
     \catcode`\%14%
```

The we make sure that the important syntax parts have the right `\catcode`.

```
   \@makeother\<%
   \@makeother\>%
   \@makeother\*%
   \@makeother\.%
   \@makeother\-%
   \@makeother\/%
   \@makeother\[%
   \@makeother\]%
   \@makeother\`%
   \@makeother\'%
   \@makeother\"%
}
```

`\LoadFontDefinitionFile` Load and `.fd` files for some encoding and family (if it exists).

```
⟨/2ekernel⟩
⟨*2ekernel | latexrelease⟩
⟨latexrelease⟩\IncludeInRelease{2020/02/02}%
⟨latexrelease⟩                 {\LoadFontDefinitionFile}{Loading .fd files}%
\def\LoadFontDefinitionFile#1#2{%
  \begingroup
    \edef\f@encoding{#1}%
    \edef\f@family{#2}%
    \try@load@fontshape
  \endgroup
}
⟨/2ekernel | latexrelease⟩
⟨latexrelease⟩\EndIncludeInRelease
⟨latexrelease⟩\IncludeInRelease{0000/00/00}%
⟨latexrelease⟩                 {\LoadFontDefinitionFile}{Loading .fd files}%
⟨latexrelease⟩
⟨latexrelease⟩\let\LoadFontDefinitionFile\@undefined
⟨latexrelease⟩\EndIncludeInRelease
⟨*2ekernel⟩
```

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 tht 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 TEX 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).

```
⟨/2ekernel⟩
⟨*2ekernel | latexrelease⟩
⟨latexrelease⟩\IncludeInRelease{2020/02/02}%
⟨latexrelease⟩        {\DeclareFontFamilySubstitution}{Provide family substituation}%
\begingroup
\nfss@catcodes
\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 arange for this. Otherwise we might run into trouble because the necessary `\DeclareFontFamily` has not been seen.

```
  \LoadFontDefinitionFile{#1}{#2}%
  \LoadFontDefinitionFile{#1}{#3}%

  \DeclareFontShape{#1}{#2}{m}{it}{<->ssub * #3/m/it}{}%
  \DeclareFontShape{#1}{#2}{m}{n}{<->ssub * #3/m/n}{}%
  \DeclareFontShape{#1}{#2}{m}{sc}{<->ssub * #3/m/sc}{}%
  \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.

```
  \DeclareFontShape{#1}{#2}{m}{sw}{<->ssub * #3/m/sw}{}%
  \DeclareFontShape{#1}{#2}{m}{scit}{<->ssub * #3/m/scit}{}%
  \DeclareFontShape{#1}{#2}{m}{scsl}{<->ssub * #3/m/scsl}{}%
```

Same game with `b` and `bx`, for other weights you are on your own:

```
  \DeclareFontShape{#1}{#2}{b}{it}{<->ssub * #3/b/it}{}%
  \DeclareFontShape{#1}{#2}{b}{n}{<->ssub * #3/b/n}{}%
  \DeclareFontShape{#1}{#2}{b}{scit}{<->ssub * #3/b/scit}{}%
  \DeclareFontShape{#1}{#2}{b}{scsl}{<->ssub * #3/b/scsl}{}%
  \DeclareFontShape{#1}{#2}{b}{sc}{<->ssub * #3/b/sc}{}%
  \DeclareFontShape{#1}{#2}{b}{sl}{<->ssub * #3/b/sl}{}%
  \DeclareFontShape{#1}{#2}{b}{sw}{<->ssub * #3/b/sw}{}%
  \DeclareFontShape{#1}{#2}{bx}{it}{<->ssub * #3/bx/it}{}%
  \DeclareFontShape{#1}{#2}{bx}{n}{<->ssub * #3/bx/n}{}%
  \DeclareFontShape{#1}{#2}{bx}{scit}{<->ssub * #3/bx/scit}{}%
  \DeclareFontShape{#1}{#2}{bx}{scsl}{<->ssub * #3/bx/scsl}{}%
  \DeclareFontShape{#1}{#2}{bx}{sc}{<->ssub * #3/bx/sc}{}%
  \DeclareFontShape{#1}{#2}{bx}{sl}{<->ssub * #3/bx/sl}{}%
  \DeclareFontShape{#1}{#2}{bx}{sw}{<->ssub * #3/bx/sw}{}%
```

```
}
\endgroup
⟨/2ekernel | latexrelease⟩
⟨latexrelease⟩\EndIncludeInRelease
⟨latexrelease⟩\IncludeInRelease{0000/00/00}%
⟨latexrelease⟩       {\DeclareFontFamilySubstitution}{Provide family substituation}%
⟨latexrelease⟩
⟨latexrelease⟩\let\DeclareFontFamilySubstitution\@undefined
⟨latexrelease⟩\EndIncludeInRelease
⟨*2ekernel⟩
```

\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!

```
⟨/2ekernel⟩
⟨*2ekernel | latexrelease⟩
⟨latexrelease⟩\IncludeInRelease{2019/10/01}%
⟨latexrelease⟩                   {\DeclareErrorFont}{No side effects please}%
\def\DeclareErrorFont#1#2#3#4#5{%
      \xdef\error@fontshape{%
          \noexpand\expandafter\noexpand\split@name\noexpand\string
          \expandafter\noexpand\csname#1/#2/#3/#4/#5\endcsname
          \noexpand\@nil}%
```

Initialize all those internal variables which may or may not have values in the first seconds of NFSS' bootstraping 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 grap "error default values" as well. However, this probably should go also—and now it did.

```
%       \gdef\f@encoding{#1}%
      \gdef\default@family{#2}%
      \gdef\default@series{#3}%
      \gdef\default@shape{#4}%
}
⟨/2ekernel | latexrelease⟩
⟨latexrelease⟩\EndIncludeInRelease
⟨latexrelease⟩\IncludeInRelease{0000/00/00}%
⟨latexrelease⟩                   {\DeclareErrorFont}{No side effects please}%
⟨latexrelease⟩
⟨latexrelease⟩\def\DeclareErrorFont#1#2#3#4#5{%
⟨latexrelease⟩       \xdef\error@fontshape{%
⟨latexrelease⟩           \noexpand\expandafter\noexpand\split@name\noexpand\string
⟨latexrelease⟩           \expandafter\noexpand\csname#1/#2/#3/#4/#5\endcsname
⟨latexrelease⟩           \noexpand\@nil}%
⟨latexrelease⟩       \gdef\default@family{#2}%
⟨latexrelease⟩       \gdef\default@series{#3}%
⟨latexrelease⟩       \gdef\default@shape{#4}%
⟨latexrelease⟩       \global\let\f@family\default@family
⟨latexrelease⟩       \global\let\f@series\default@series
⟨latexrelease⟩       \global\let\f@shape\default@shape
⟨latexrelease⟩       \gdef\f@size{#5}%
⟨latexrelease⟩       \gdef\f@baselineskip{#5pt}%
⟨latexrelease⟩}
```

```
⟨latexrelease⟩\EndIncludeInRelease
⟨*2ekernel⟩
```

```
\@onlypreamble\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 TEX will go into an infinite loop. But if the defaults are set incorrectly one deserves nothing else!

```
⟨/2ekernel⟩
⟨latexrelease⟩\IncludeInRelease{2015/01/01}{\wrong@fontshape}%
⟨latexrelease⟩                  {Font substituation in preamble}%
⟨*2ekernel | latexrelease⟩
\def\wrong@fontshape{%
    \csname D@\f@encoding\endcsname   % install defaults if in math
```

We remember the wanted `\curr@fontshape` combination which we will need in a moment.

```
    \edef\reserved@a{\csname\curr@fontshape\endcsname}%
  \ifx\last@fontshape\reserved@a
     \errmessage{Corrupted NFSS tables}%
     \error@fontshape
  \else
```

Then we warn the user about the mess and set the shape to its default.

```
    \let\f@shape\default@shape
```

If the combination is not known, try the default *series*.

```
    \expandafter\ifx\csname\curr@fontshape\endcsname\relax
       \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!

```
         \expandafter
           \ifx\csname\curr@fontshape\endcsname\relax
            \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 corruped". After begin document that will not happen as all `.fd` files involved in substituation are loaded at `\begin{document}`.

```
            \begingroup
               \try@load@fontshape
            \endgroup
         \fi \fi
  \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 TEX 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.

```
     \@font@warning{Font shape ‘\expandafter\string\reserved@a’
                     \expandafter\@gobble\string\@undefined\MessageBreak
                   using ‘\curr@fontshape’ instead\@wrong@font@char}%
    \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`.

```
    \gdef\@defaultsubs{%
      \@font@warning{Some font shapes were not available, defaults
                      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.

```
    \global\expandafter\expandafter\expandafter\let
       \expandafter\reserved@a
           \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.

```
    \xdef\font@name{%
      \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.

```
    \pickup@font}
⟨/2ekernel | latexrelease⟩
⟨latexrelease⟩\EndIncludeInRelease
⟨latexrelease⟩\IncludeInRelease{0000/00/00}{\wrong@fontshape}%
⟨latexrelease⟩                  {Font substituation in preamble}%
⟨latexrelease⟩\def\wrong@fontshape{%
⟨latexrelease⟩    \csname D@\f@encoding\endcsname
⟨latexrelease⟩    \edef\reserved@a{\csname\curr@fontshape\endcsname}%
⟨latexrelease⟩  \ifx\last@fontshape\reserved@a
⟨latexrelease⟩     \errmessage{Corrupted NFSS tables}%
⟨latexrelease⟩     \error@fontshape
⟨latexrelease⟩  \else
⟨latexrelease⟩    \let\f@shape\default@shape
⟨latexrelease⟩    \expandafter\ifx\csname\curr@fontshape\endcsname\relax
⟨latexrelease⟩       \let\f@series\default@series
⟨latexrelease⟩        \expandafter
⟨latexrelease⟩          \ifx\csname\curr@fontshape\endcsname\relax
⟨latexrelease⟩           \let\f@family\default@family
⟨latexrelease⟩        \fi \fi
⟨latexrelease⟩  \fi
⟨latexrelease⟩     \@font@warning{Font shape
⟨latexrelease⟩            ‘\expandafter\string\reserved@a’
⟨latexrelease⟩            \expandafter\@gobble\string\@undefined
⟨latexrelease⟩            \MessageBreak
⟨latexrelease⟩            using `\curr@fontshape' instead\@wrong@font@char}%
⟨latexrelease⟩    \global\let\last@fontshape\reserved@a
⟨latexrelease⟩    \gdef\@defaultsubs{%
⟨latexrelease⟩      \@font@warning{Some font shapes were not available,
⟨latexrelease⟩                        defaults substituted.\@gobbletwo}}%
⟨latexrelease⟩    \global\expandafter\expandafter\expandafter\let
⟨latexrelease⟩       \expandafter\reserved@a
⟨latexrelease⟩           \csname\curr@fontshape\endcsname
⟨latexrelease⟩    \xdef\font@name{%
⟨latexrelease⟩      \csname\curr@fontshape/\f@size\endcsname}%
⟨latexrelease⟩    \pickup@font}
⟨latexrelease⟩\EndIncludeInRelease
⟨*2ekernel⟩
```

\@wrong@font@char Normally empty but redefined in \UseTextSymbol so that the Font shape undefined message can refer to the symbol causing the problem.

```
\let\@wrong@font@char\@empty
```

\@@defaultsubs
\@defaultsubs See above.

```
\let\@defaultsubs\relax
```

\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 TEXbook, p. 382).

```
\def\strip@prefix#1>{}
```

27 Assigning math fonts to *versions*

\install@mathalphabet This is just another name for \gdef but we can redefine it if necessary later on.

```
\let\install@mathalphabet\gdef
```

\math@fonts

```
\let\math@fonts\@empty
```

\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.

```
%\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.

```
%  \begingroup
```

We set the math fonts for the *family* in question by calling \getanddefine@fonts in the correct environment.

```
%      \escapechar\m@ne
%     \getanddefine@fonts{\csname c@mv@\math@version\endcsname}#3%
```

We globally select the math fonts. . .

```
%     \globaldefs\@ne  \math@fonts
```

. . . and close the group to restore \globaldefs and \escapechar.

```
%  \endgroup
```

As long as no *size* or *version* change occurs the ⟨*math alphabet identifier*⟩ should simply switch to the installed *math group* instead of calling \select@group unnecessarily. So we globally redefine the first argument (the new ⟨*math alphabet identifier*⟩) 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 ⟨*math alphabet identifier*⟩ and then calling the new definition to switch to the wanted ⟨*math group*⟩. Now we define the ⟨*math alphabet identifier*⟩ as a call to the \use@mathgroup command.

```
%  \xdef#1{\noexpand\use@mathgroup\noexpand#2%
%           {\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 identifer #1 and the math version macro as arguments.

```
%      \expandafter\extract@alph@from@version
%         \csname mv@\math@version\expandafter\endcsname
%        \expandafter{\number\csname c@mv@\math@version\endcsname}%
%        #1%
%      \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 od \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.

```
%\expandafter #1\fi}
```

\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.

```
\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`.

```
     \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`.

```
         \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`.

```
         \def\reserved@c####1{\gdef#1{##1####1##3}}}%
```

Then we execute our auxiliary macro.

```
     \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.

```
     \def\reserved@a\select@group#3##1##2\@nil{%
```

This macro can now directly rebuild the math version definition by calling `\reserved@c`:

```
        \reserved@c{%
           \getanddefine@fonts{#2}##2%
           \install@mathalphabet#3{%
              \relax\ifmmode \else \non@alpherr#3\fi
              \use@mathgroup##1{#2}}}%
```

In addtion it defines the alphabet the way it should be used from now on.

```
       \gdef#3{\relax\ifmmode \else \non@alpherr#3\fi
               \use@mathgroup##1{#2}}}%
```

Finally, we only have to call this macro `\reserved@a` on the old definitions recorded in `\reserved@b`:

```
     \expandafter\reserved@a\reserved@b\@nil
     }
```

`\math@bgroup` `\math@egroup` Here are the default definitions for `\math@bgroup` and `\math@egroup`. We use `\bgroup` instead of `\begingroup` to avoid 'leaking out' of style changes. This has the side effect of always producing mathord atoms.

```
\let\math@bgroup\bgroup
\def\math@egroup#1{#1\egroup}
```

`\calculate@math@sizes` Here is the default definition for `\calculate@math@sizes` a more elaborate interface is under testing in mthscale.sty.

```
\gdef\calculate@math@sizes{%
  \@font@info{Calculating\space math\space sizes\space for\space
              size\space <\f@size>}%
  \dimen@\f@size \p@
  \@tempdimb \defaultscriptratio \dimen@
  \dimen@ \defaultscriptscriptratio \dimen@
  \expandafter\xdef\csname S@\f@size\endcsname{%
    \gdef\noexpand\tf@size{\f@size}%
    \gdef\noexpand\sf@size{\strip@pt\@tempdimb}%
    \gdef\noexpand\ssf@size{\strip@pt\dimen@}%
    \noexpand\math@fontstrue}}
```

`\defaultscriptratio`
`\defaultscriptscriptratio`

The default ratio for math sizes is:
1 to `\defaultscriptratio` to `\defaultscriptscriptratio`.
By default this is 1 to .7 to .5.

```
\def\defaultscriptratio{.7}
\def\defaultscriptscriptratio{.5}
```

`\noaccents@`

If we don't have a definition for `\noaccents@` we provide a dummy.

```
\ifx\noaccents@\@undefined
  \let\noaccents@\@empty
\fi
```

`\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.

```
⟨/2ekernel⟩
⟨latexrelease⟩\IncludeInRelease{2017/01/01}{\showhyphens}%
⟨latexrelease⟩                 {XeTeX support for \showhyphens}%
⟨*2ekernel | latexrelease⟩
\ifx\XeTeXcharclass\@undefined
```

Version for engines other than XeTeX.

```
\DeclareRobustCommand\showhyphens[1]{%
  \setbox0\vbox{%
    \color@begingroup
    \everypar{}%
    \parfillskip\z@skip\hsize\maxdimen
    \normalfont
    \pretolerance\m@ne\tolerance\m@ne\hbadness\z@\showboxdepth\z@\ #1%
    \color@endgroup}}
```

```
\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.

```
\DeclareRobustCommand\showhyphens[1]{%
  \setbox0\vbox{%
```

```
    \usefont{TU}{lmr}{m}{n}%
    \hsize 1sp %
    \hbadness\@M
    \hfuzz\maxdimen
    \tracingonline\z@
    \everypar={}%
    \leftskip\z@skip
    \rightskip\z@skip
    \parfillskip\z@skip
    \hyphenpenalty=-\@M
    \pretolerance\m@ne
    \interlinepenalty\z@
    \clubpenalty\z@
    \widowpenalty\z@
    \brokenpenalty1127 %
    \setbox\z@\hbox{}%
    \noindent
    \hskip\z@skip
    #1%
    \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`).

```
     \loop
     \@tempswafalse
     \ifnum\lastnodetype=11\unskip\@tempswatrue\fi
     \ifnum\lastnodetype=12\unkern\@tempswatrue\fi
     \ifnum\lastnodetype=13 %
      \count@\lastpenalty
      \unpenalty\@tempswatrue
    \fi
    \ifnum\lastnodetype=\@ne
     \setbox\tw@\lastbox\@tempswatrue
     \setbox0\hbox{\unhbox\tw@\unskip\unskip\unpenalty
                   \ifnum\count@=1127 \else\ \fi
                   \unhbox0}%
     \count@\z@
    \fi
    \if@tempswa
    \repeat
   \hbadness\z@
   \hsize\maxdimen
   \showboxdepth\z@
   \tolerance\m@ne
   \hyphenpenalty\z@
   \noindent\unhbox\z@
}}
\fi
⟨/2ekernel | latexrelease⟩
⟨latexrelease⟩\EndIncludeInRelease
⟨latexrelease⟩\IncludeInRelease{0000/00/00}{\showhyphens}%
⟨latexrelease⟩                {XeTeX support for \showhyphens}%
⟨latexrelease⟩\gdef\showhyphens#1{%
```

```
⟨latexrelease⟩  \setbox0\vbox{%
⟨latexrelease⟩    \color@begingroup
⟨latexrelease⟩    \everypar{}%
⟨latexrelease⟩    \parfillskip\z@skip\hsize\maxdimen
⟨latexrelease⟩    \normalfont
⟨latexrelease⟩    \pretolerance\m@ne\tolerance\m@ne
⟨latexrelease⟩    \hbadness\z@\showboxdepth\z@\ #1%
⟨latexrelease⟩    \color@endgroup}}
⟨latexrelease⟩\EndIncludeInRelease
⟨*2ekernel⟩
```

\addto@hook We need a macro to add tokens to a hook.

```
\long\def\addto@hook#1#2{#1\expandafter{\the#1#2}}
```

\@vpt

```
\def\@vpt{5}
```

\@vipt

```
\def\@vipt{6}
```

\@viipt

```
\def\@viipt{7}
```

\@viiipt

```
\def\@viiipt{8}
```

\@ixpt

```
\def\@ixpt{9}
```

\@xpt

```
\def\@xpt{10}
```

\@xipt

```
\def\@xipt{10.95}
```

\@xiipt

```
\def\@xiipt{12}
```

\@xivpt

```
\def\@xivpt{14.4}
```

\@xviipt

```
\def\@xviipt{17.28}
```

\@xxpt

```
\def\@xxpt{20.74}
```

\@xxvpt

```
\def\@xxvpt{24.88}
```

```
⟨/2ekernel⟩
```

File p
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 LaTeX distribution, or *The LaTeX Companion* for higher level documentation of the LaTeX Font Selection Scheme.

Everything in the this file got introduced 2020/02/02, so we do a single rollback (for now).

```
⟨*2ekernel⟩
⟨/2ekernel⟩
⟨*2ekernel | latexrelease⟩
⟨latexrelease⟩\IncludeInRelease{2020/02/02}%
⟨latexrelease⟩    {\DeclareFontSeriesChangeRule}{Series change rules}%
```

28 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`.

28.1 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.

```
\def\DeclareFontSeriesChangeRule#1#2#3#4{%
  \@namedef{series@#1@#2}{{#3}{#4}}}
```

28.2 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 "`m`*width*" and "*weight*`m`", 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 automatially gets reduced result in `b` or `mc` (medium weight condensed) which becomes `c` as a result.

- Only a few entries have "alterative" 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 wth `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.

```
\DeclareFontSeriesChangeRule {bc}{b}{bc}{}
\DeclareFontSeriesChangeRule {bc}{c}{bc}{}
\DeclareFontSeriesChangeRule {bc}{eb}{ebc}{}
\DeclareFontSeriesChangeRule {bc}{ec}{bec} {bc}
\DeclareFontSeriesChangeRule {bc}{el}{elc}{}
\DeclareFontSeriesChangeRule {bc}{l}{lc}{}
\DeclareFontSeriesChangeRule {bc}{sb}{sbc}{}
\DeclareFontSeriesChangeRule {bc}{sc}{bsc} {bc}
\DeclareFontSeriesChangeRule {bc}{sl}{slc}{}
\DeclareFontSeriesChangeRule {bc}{ub}{ubc}{}
\DeclareFontSeriesChangeRule {bc}{ul}{ulc}{}
\DeclareFontSeriesChangeRule {bc}{x}{bx}{}
```

```
\DeclareFontSeriesChangeRule {bx}{b}{bx}{}
\DeclareFontSeriesChangeRule {bx}{c}  {bc}  {bx} %<-----
\DeclareFontSeriesChangeRule {bx}{eb}{ebx}{}
\DeclareFontSeriesChangeRule {bx}{ec}  {bec}  {bx} %<-----
\DeclareFontSeriesChangeRule {bx}{el}{elx}{}
\DeclareFontSeriesChangeRule {bx}{l}{lx}{}
\DeclareFontSeriesChangeRule {bx}{sb} {sbx} {}
\DeclareFontSeriesChangeRule {bx}{sc}  {bsc}  {bx} %<-----
\DeclareFontSeriesChangeRule {bx}{sl}{slx}  {}
\DeclareFontSeriesChangeRule {bx}{ub}{ubx}{}
\DeclareFontSeriesChangeRule {bx}{ul}{ulx}{}
\DeclareFontSeriesChangeRule {bx}{x}{bx}{}

\DeclareFontSeriesChangeRule {b}{bx} {bx}  {b} %<-----
\DeclareFontSeriesChangeRule {b}{c}  {bc}  {b} %<-----
\DeclareFontSeriesChangeRule {b}{ec}  {bec}  {b} %<-----
\DeclareFontSeriesChangeRule {b}{sb} {sb}  {b} %<-----
\DeclareFontSeriesChangeRule {b}{sc}  {bsc}  {b} %<-----
\DeclareFontSeriesChangeRule {b}{x}  {bx}  {b} %<-----

\DeclareFontSeriesChangeRule {c}{bx} {bx}  {b}  %<-----
\DeclareFontSeriesChangeRule {c}{b}{bc}{}
\DeclareFontSeriesChangeRule {c}{eb}{ebc}{}
\DeclareFontSeriesChangeRule {c}{el}{elc}{}
\DeclareFontSeriesChangeRule {c}{l}{lc}{}
\DeclareFontSeriesChangeRule {c}{sb}{sbc}{}
\DeclareFontSeriesChangeRule {c}{sl}{slc}{}
\DeclareFontSeriesChangeRule {c}{ub}{ubc}{}
\DeclareFontSeriesChangeRule {c}{ul}{ulc}{}
\DeclareFontSeriesChangeRule {c}{x}{x}{m}              %<-----

\DeclareFontSeriesChangeRule {ebc}{b}{bc}{}
\DeclareFontSeriesChangeRule {ebc}{c}{ebc}{}
\DeclareFontSeriesChangeRule {ebc}{eb}{ebc}{}
\DeclareFontSeriesChangeRule {ebc}{ec}{ebec}{ebc}
\DeclareFontSeriesChangeRule {ebc}{el}{elc}{}
\DeclareFontSeriesChangeRule {ebc}{l}{lc}{}
\DeclareFontSeriesChangeRule {ebc}{sb}{sbc}{}
\DeclareFontSeriesChangeRule {ebc}{sc}{ebsc}{ebc}
\DeclareFontSeriesChangeRule {ebc}{sl}{slc}{}
\DeclareFontSeriesChangeRule {ebc}{ub}{ubc}{}
\DeclareFontSeriesChangeRule {ebc}{ul}{ulc}{}
\DeclareFontSeriesChangeRule {ebc}{x}{ebx}{}

\DeclareFontSeriesChangeRule {ec}{bx} {bx}  {b}  %<-----
\DeclareFontSeriesChangeRule {ec}{b}{bec}{}
\DeclareFontSeriesChangeRule {ec}{eb}{ebec}{}
\DeclareFontSeriesChangeRule {ec}{el}{elec}{}
\DeclareFontSeriesChangeRule {ec}{l}{lec}{}
\DeclareFontSeriesChangeRule {ec}{sb}{sbec}{}
\DeclareFontSeriesChangeRule {ec}{sl}{slec}{}
\DeclareFontSeriesChangeRule {ec}{ub}{ubec}{}
\DeclareFontSeriesChangeRule {ec}{ul}{ulec}{}
\DeclareFontSeriesChangeRule {ec}{x}{x}{m}             %<-----

\DeclareFontSeriesChangeRule {sc}{bx} {bx}  {b}  %<-----
\DeclareFontSeriesChangeRule {sc}{b}{bsc}{}
```

```
\DeclareFontSeriesChangeRule {sc}{eb}{ebsc}{}
\DeclareFontSeriesChangeRule {sc}{el}{elsc}{}
\DeclareFontSeriesChangeRule {sc}{l}{lsc}{}
\DeclareFontSeriesChangeRule {sc}{sb}{sbsc}{}
\DeclareFontSeriesChangeRule {sc}{sl}{slsc}{}
\DeclareFontSeriesChangeRule {sc}{ub}{ubsc}{}
\DeclareFontSeriesChangeRule {sc}{ul}{ulsc}{}
\DeclareFontSeriesChangeRule {sc}{x}{x}{m}              %<-----

\DeclareFontSeriesChangeRule {ebx}{b}{bx}{}
\DeclareFontSeriesChangeRule {ebx}{c}{ebc}{}
\DeclareFontSeriesChangeRule {ebx}{eb}{ebx}{}
\DeclareFontSeriesChangeRule {ebx}{ec}{ebec}{}
\DeclareFontSeriesChangeRule {ebx}{el}{elx}{}
\DeclareFontSeriesChangeRule {ebx}{l}{lx}{}
\DeclareFontSeriesChangeRule {ebx}{sb}{sbx}{}
\DeclareFontSeriesChangeRule {ebx}{sc}{ebsc}{}
\DeclareFontSeriesChangeRule {ebx}{sl}{slx}{}
\DeclareFontSeriesChangeRule {ebx}{ub}{ubx}{}
\DeclareFontSeriesChangeRule {ebx}{ul}{ulx}{}
\DeclareFontSeriesChangeRule {ebx}{x}{ebx}{}

\DeclareFontSeriesChangeRule {eb}{c}{ebc}{}
\DeclareFontSeriesChangeRule {eb}{ec}{ebec}{}
\DeclareFontSeriesChangeRule {eb}{sc}{ebsc}{}
\DeclareFontSeriesChangeRule {eb}{x}{ebx}{}

\DeclareFontSeriesChangeRule {elc}{b}{bc}{}
\DeclareFontSeriesChangeRule {elc}{c}{elc}{}
\DeclareFontSeriesChangeRule {elc}{eb}{ebc}{}
\DeclareFontSeriesChangeRule {elc}{ec}{elec}{}
\DeclareFontSeriesChangeRule {elc}{el}{elc}{}
\DeclareFontSeriesChangeRule {elc}{l}{lc}{}
\DeclareFontSeriesChangeRule {elc}{sb}{sbc}{}
\DeclareFontSeriesChangeRule {elc}{sc}{elsc}{}
\DeclareFontSeriesChangeRule {elc}{sl}{slc}{}
\DeclareFontSeriesChangeRule {elc}{ub}{ubc}{}
\DeclareFontSeriesChangeRule {elc}{ul}{ulc}{}
\DeclareFontSeriesChangeRule {elc}{x}{elx}{}

\DeclareFontSeriesChangeRule {elx}{b}{bx}{}
\DeclareFontSeriesChangeRule {elx}{c}{elc}{}
\DeclareFontSeriesChangeRule {elx}{eb}{ebx}{}
\DeclareFontSeriesChangeRule {elx}{ec}{elec}{}
\DeclareFontSeriesChangeRule {elx}{el}{elx}{}
\DeclareFontSeriesChangeRule {elx}{l}{lx}{}
\DeclareFontSeriesChangeRule {elx}{sb}{sbx}{}
\DeclareFontSeriesChangeRule {elx}{sc}{elsc}{}
\DeclareFontSeriesChangeRule {elx}{sl}{slx}{}
\DeclareFontSeriesChangeRule {elx}{ub}{ubx}{}
\DeclareFontSeriesChangeRule {elx}{ul}{ulx}{}
\DeclareFontSeriesChangeRule {elx}{x}{elx}{}

\DeclareFontSeriesChangeRule {el}{c}{elc}{}
\DeclareFontSeriesChangeRule {el}{ec}{elec}{}
\DeclareFontSeriesChangeRule {el}{sc}{elsc}{}
\DeclareFontSeriesChangeRule {el}{x}{elx}{}
```

```
\DeclareFontSeriesChangeRule {lc}{b}{bc}{}
\DeclareFontSeriesChangeRule {lc}{c}{lc}{}
\DeclareFontSeriesChangeRule {lc}{eb}{ebc}{}
\DeclareFontSeriesChangeRule {lc}{ec}{lec}{}
\DeclareFontSeriesChangeRule {lc}{el}{elc}{}
\DeclareFontSeriesChangeRule {lc}{l}{lc}{}
\DeclareFontSeriesChangeRule {lc}{sb}{sbc}{}
\DeclareFontSeriesChangeRule {lc}{sc}{lsc}{}
\DeclareFontSeriesChangeRule {lc}{sl}{slc}{}
\DeclareFontSeriesChangeRule {lc}{ub}{ubc}{}
\DeclareFontSeriesChangeRule {lc}{ul}{ulc}{}
\DeclareFontSeriesChangeRule {lc}{x}{lx}{}

\DeclareFontSeriesChangeRule {lx}{b}{bx}{}
\DeclareFontSeriesChangeRule {lx}{c}{lc}{}
\DeclareFontSeriesChangeRule {lx}{eb}{ebx}{}
\DeclareFontSeriesChangeRule {lx}{ec}{lec}{}
\DeclareFontSeriesChangeRule {lx}{el}{elx}{}
\DeclareFontSeriesChangeRule {lx}{l}{lx}{}
\DeclareFontSeriesChangeRule {lx}{sb}{sbx}{}
\DeclareFontSeriesChangeRule {lx}{sc}{lsc}{}
\DeclareFontSeriesChangeRule {lx}{sl}{slx}{}
\DeclareFontSeriesChangeRule {lx}{ub}{ubx}{}
\DeclareFontSeriesChangeRule {lx}{ul}{ulx}{}
\DeclareFontSeriesChangeRule {lx}{x}{lx}{}

\DeclareFontSeriesChangeRule {l}{bx} {bx}  {b} %<-----
\DeclareFontSeriesChangeRule {l}{b}  {b}   {bx} %<-----
\DeclareFontSeriesChangeRule {l}{c}  {lc}  {l}  % ? %<-----
\DeclareFontSeriesChangeRule {l}{ec}  {lec}  {l}  % ? %<-----
\DeclareFontSeriesChangeRule {l}{sb} {sb}  {b}  % ? %<-----
\DeclareFontSeriesChangeRule {l}{sc}  {lsc}  {l}  % ? %<-----
\DeclareFontSeriesChangeRule {l}{x}  {lx}  {l}  % ? %<-----

\DeclareFontSeriesChangeRule {m}{bx} {bx} {b}    %<-----
\DeclareFontSeriesChangeRule {m}{b}  {b}  {bx} %<-----
\DeclareFontSeriesChangeRule {m}{c}  {c}  {m} %<-----
\DeclareFontSeriesChangeRule {m}{ec}  {ec}  {m} %<-----
\DeclareFontSeriesChangeRule {m}{l}  {l}  {m} %<-----
\DeclareFontSeriesChangeRule {m}{sb} {sb} {b} %<-----
\DeclareFontSeriesChangeRule {m}{sc}  {sc}  {m} %<-----
\DeclareFontSeriesChangeRule {m}{x}  {x}  {m} %<-----

\DeclareFontSeriesChangeRule {sbc}{b}{bc}{}
\DeclareFontSeriesChangeRule {sbc}{c}{sbc}{}
\DeclareFontSeriesChangeRule {sbc}{eb}{ebc}{}
\DeclareFontSeriesChangeRule {sbc}{ec}{sbec}{sbc}
\DeclareFontSeriesChangeRule {sbc}{el}{elc}{}
\DeclareFontSeriesChangeRule {sbc}{l}{lc}{}
\DeclareFontSeriesChangeRule {sbc}{sb}{sbc}{}
\DeclareFontSeriesChangeRule {sbc}{sc}{sbsc}{sbc}
\DeclareFontSeriesChangeRule {sbc}{sl}{slc}{}
\DeclareFontSeriesChangeRule {sbc}{ub}{ubc}{}
\DeclareFontSeriesChangeRule {sbc}{ul}{ulc}{}
\DeclareFontSeriesChangeRule {sbc}{x}{sbx}{}

\DeclareFontSeriesChangeRule {sbx}{b}{bx}{}
```

```
\DeclareFontSeriesChangeRule {sbx}{c}{sbc}{}
\DeclareFontSeriesChangeRule {sbx}{eb}{ebx}{}
\DeclareFontSeriesChangeRule {sbx}{ec}{sbec}{}
\DeclareFontSeriesChangeRule {sbx}{el}{elx}{}
\DeclareFontSeriesChangeRule {sbx}{l}{lx}{}
\DeclareFontSeriesChangeRule {sbx}{sb}{sbx}{}
\DeclareFontSeriesChangeRule {sbx}{sc}{sbsc}{}
\DeclareFontSeriesChangeRule {sbx}{sl}{slx}{}
\DeclareFontSeriesChangeRule {sbx}{ub}{ubx}{}
\DeclareFontSeriesChangeRule {sbx}{ul}{ulx}{}
\DeclareFontSeriesChangeRule {sbx}{x}{sbx}{}

\DeclareFontSeriesChangeRule {sb}{c}  {sbc} {bc} %? %<-----
\DeclareFontSeriesChangeRule {sb}{ec}  {sbec} {sbc} %? %<-----
\DeclareFontSeriesChangeRule {sb}{sc}  {sbsc} {sbc} %? %<-----
\DeclareFontSeriesChangeRule {sb}{x}  {sbx} {bx} %? %<-----

\DeclareFontSeriesChangeRule {slc}{b}{bc}{}
\DeclareFontSeriesChangeRule {slc}{c}{slc}{}
\DeclareFontSeriesChangeRule {slc}{eb}{ebc}{}
\DeclareFontSeriesChangeRule {slc}{ec}{slec}{}
\DeclareFontSeriesChangeRule {slc}{el}{elc}{}
\DeclareFontSeriesChangeRule {slc}{l}{lc}{}
\DeclareFontSeriesChangeRule {slc}{sb}{sbc}{}
\DeclareFontSeriesChangeRule {slc}{sc}{slsc}{}
\DeclareFontSeriesChangeRule {slc}{sl}{slc}{}
\DeclareFontSeriesChangeRule {slc}{ub}{ubc}{}
\DeclareFontSeriesChangeRule {slc}{ul}{ulc}{}
\DeclareFontSeriesChangeRule {slc}{x}{slx}{}

\DeclareFontSeriesChangeRule {slx}{b}{bx}{}
\DeclareFontSeriesChangeRule {slx}{c}{slc}{}
\DeclareFontSeriesChangeRule {slx}{eb}{ebx}{}
\DeclareFontSeriesChangeRule {slx}{ec}{slec}{}
\DeclareFontSeriesChangeRule {slx}{el}{elx}{}
\DeclareFontSeriesChangeRule {slx}{l}{lx}{}
\DeclareFontSeriesChangeRule {slx}{sb}{sbx}{}
\DeclareFontSeriesChangeRule {slx}{sc}{slsc}{}
\DeclareFontSeriesChangeRule {slx}{sl}{slx}{}
\DeclareFontSeriesChangeRule {slx}{ub}{ubx}{}
\DeclareFontSeriesChangeRule {slx}{ul}{ulx}{}
\DeclareFontSeriesChangeRule {slx}{x}{slx}{}

\DeclareFontSeriesChangeRule {sl}{c}{slc}{}
\DeclareFontSeriesChangeRule {sl}{ec}{slec}{}
\DeclareFontSeriesChangeRule {sl}{sc}{slsc}{}
\DeclareFontSeriesChangeRule {sl}{x}{slx}{}

\DeclareFontSeriesChangeRule {ubc}{b}{bc}{}
\DeclareFontSeriesChangeRule {ubc}{c}{ubc}{}
\DeclareFontSeriesChangeRule {ubc}{eb}{ebc}{}
\DeclareFontSeriesChangeRule {ubc}{ec}{ubec}{}
\DeclareFontSeriesChangeRule {ubc}{el}{elc}{}
\DeclareFontSeriesChangeRule {ubc}{l}{lc}{}
\DeclareFontSeriesChangeRule {ubc}{sb}{sbc}{}
\DeclareFontSeriesChangeRule {ubc}{sc}{ubsc}{}
\DeclareFontSeriesChangeRule {ubc}{sl}{slc}{}
```

```
\DeclareFontSeriesChangeRule {ubc}{ub}{ubc}{}
\DeclareFontSeriesChangeRule {ubc}{ul}{ulc}{}
\DeclareFontSeriesChangeRule {ubc}{x}{ubx}{}

\DeclareFontSeriesChangeRule {ubx}{b}{bx}{}
\DeclareFontSeriesChangeRule {ubx}{c}{ubc}{}
\DeclareFontSeriesChangeRule {ubx}{eb}{ebx}{}
\DeclareFontSeriesChangeRule {ubx}{ec}{ubec}{}
\DeclareFontSeriesChangeRule {ubx}{el}{elx}{}
\DeclareFontSeriesChangeRule {ubx}{l}{lx}{}
\DeclareFontSeriesChangeRule {ubx}{sb}{sbx}{}
\DeclareFontSeriesChangeRule {ubx}{sc}{ubsc}{}
\DeclareFontSeriesChangeRule {ubx}{sl}{slx}{}
\DeclareFontSeriesChangeRule {ubx}{ub}{ubx}{}
\DeclareFontSeriesChangeRule {ubx}{ul}{ulx}{}
\DeclareFontSeriesChangeRule {ubx}{x}{ubx}{}

\DeclareFontSeriesChangeRule {ub}{c}{ubc}{}
\DeclareFontSeriesChangeRule {ub}{ec}{ubec}{}
\DeclareFontSeriesChangeRule {ub}{sc}{ubsc}{}
\DeclareFontSeriesChangeRule {ub}{x}{ubx}{}

\DeclareFontSeriesChangeRule {ulc}{b}{bc}{}
\DeclareFontSeriesChangeRule {ulc}{c}{ulc}{}
\DeclareFontSeriesChangeRule {ulc}{eb}{ebc}{}
\DeclareFontSeriesChangeRule {ulc}{ec}{ulec}{ulc}
\DeclareFontSeriesChangeRule {ulc}{el}{elc}{}
\DeclareFontSeriesChangeRule {ulc}{l}{lc}{}
\DeclareFontSeriesChangeRule {ulc}{sb}{sbc}{}
\DeclareFontSeriesChangeRule {ulc}{sc}{ulsc}{ulc}
\DeclareFontSeriesChangeRule {ulc}{sl}{slc}{}
\DeclareFontSeriesChangeRule {ulc}{ub}{ubc}{}
\DeclareFontSeriesChangeRule {ulc}{ul}{ulc}{}
\DeclareFontSeriesChangeRule {ulc}{x}{ulx}{}

\DeclareFontSeriesChangeRule {ulx}{b}{bx}{}
\DeclareFontSeriesChangeRule {ulx}{c}{ulc}{}
\DeclareFontSeriesChangeRule {ulx}{eb}{ebx}{}
\DeclareFontSeriesChangeRule {ulx}{ec}{ulec}{}
\DeclareFontSeriesChangeRule {ulx}{el}{elx}{}
\DeclareFontSeriesChangeRule {ulx}{l}{lx}{}
\DeclareFontSeriesChangeRule {ulx}{sb}{sbx}{}
\DeclareFontSeriesChangeRule {ulx}{sc}{ulsc}{}
\DeclareFontSeriesChangeRule {ulx}{sl}{slx}{}
\DeclareFontSeriesChangeRule {ulx}{ub}{ubx}{}
\DeclareFontSeriesChangeRule {ulx}{ul}{ulx}{}
\DeclareFontSeriesChangeRule {ulx}{x}{ulx}{}

\DeclareFontSeriesChangeRule {ul}{c}{ulc}{}
\DeclareFontSeriesChangeRule {ul}{ec}{ulec}{}
\DeclareFontSeriesChangeRule {ul}{sc}{ulsc}{}
\DeclareFontSeriesChangeRule {ul}{x}{ulx}{}

\DeclareFontSeriesChangeRule {x}{b}{bx}{}
\DeclareFontSeriesChangeRule {x}{c}{c}{}
\DeclareFontSeriesChangeRule {x}{eb}{ebx}{}
\DeclareFontSeriesChangeRule {x}{ec}{ec}{}
\DeclareFontSeriesChangeRule {x}{el}{elx}{}
```

```
\DeclareFontSeriesChangeRule {x}{l}{lx}{}
\DeclareFontSeriesChangeRule {x}{sb}{sbx}{}
\DeclareFontSeriesChangeRule {x}{sc}{sc}{}
\DeclareFontSeriesChangeRule {x}{sl}{slx}{}
\DeclareFontSeriesChangeRule {x}{ub}{ubx}{}
\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?`:

```
\DeclareFontSeriesChangeRule {bc}{m?}{c}{}
\DeclareFontSeriesChangeRule {bec}{m?}{ec}{}
\DeclareFontSeriesChangeRule {bsc}{m?}{sc}{}
\DeclareFontSeriesChangeRule {bx}{m?}{x}{}
\DeclareFontSeriesChangeRule {b}{m?}{m}{}
\DeclareFontSeriesChangeRule {c}{m?}{c}{}
\DeclareFontSeriesChangeRule {ebc}{m?}{c}{}
\DeclareFontSeriesChangeRule {ebec}{m?}{ec}{}
\DeclareFontSeriesChangeRule {ebsc}{m?}{sc}{}
\DeclareFontSeriesChangeRule {ebx}{m?}{x}{}
\DeclareFontSeriesChangeRule {eb}{m?}{m}{}
\DeclareFontSeriesChangeRule {ec}{m?}{ec}{}
\DeclareFontSeriesChangeRule {elc}{m?}{c}{}
\DeclareFontSeriesChangeRule {elec}{m?}{ec}{}
\DeclareFontSeriesChangeRule {elsc}{m?}{sc}{}
\DeclareFontSeriesChangeRule {elx}{m?}{x}{}
\DeclareFontSeriesChangeRule {el}{m?}{m}{}
\DeclareFontSeriesChangeRule {lc}{m?}{c}{}
\DeclareFontSeriesChangeRule {lec}{m?}{ec}{}
\DeclareFontSeriesChangeRule {lsc}{m?}{sc}{}
\DeclareFontSeriesChangeRule {lx}{m?}{x}{}
\DeclareFontSeriesChangeRule {l}{m?}{m}{}
\DeclareFontSeriesChangeRule {m}{m?}{m}{}
\DeclareFontSeriesChangeRule {sbc}{m?}{c}{}
\DeclareFontSeriesChangeRule {sbec}{m?}{ec}{}
\DeclareFontSeriesChangeRule {sbsc}{m?}{sc}{}
\DeclareFontSeriesChangeRule {sbx}{m?}{x}{}
\DeclareFontSeriesChangeRule {sb}{m?}{m}{}
\DeclareFontSeriesChangeRule {sc}{m?}{sc}{}
\DeclareFontSeriesChangeRule {slc}{m?}{c}{}
\DeclareFontSeriesChangeRule {slec}{m?}{ec}{}
\DeclareFontSeriesChangeRule {slsc}{m?}{sc}{}
\DeclareFontSeriesChangeRule {slx}{m?}{x}{}
\DeclareFontSeriesChangeRule {sl}{m?}{m}{}
\DeclareFontSeriesChangeRule {ubc}{m?}{c}{}
\DeclareFontSeriesChangeRule {ubec}{m?}{ec}{}
\DeclareFontSeriesChangeRule {ubsc}{m?}{sc}{}
\DeclareFontSeriesChangeRule {ubx}{m?}{x}{}
\DeclareFontSeriesChangeRule {ub}{m?}{ub}{}
\DeclareFontSeriesChangeRule {ulc}{m?}{c}{}
\DeclareFontSeriesChangeRule {ulec}{m?}{ec}{}
\DeclareFontSeriesChangeRule {ulsc}{m?}{sc}{}
\DeclareFontSeriesChangeRule {ulx}{m?}{x}{}
```

```
\DeclareFontSeriesChangeRule {ul}{m?}{m}{}
\DeclareFontSeriesChangeRule {x}{m?}{x}{}
```

And there the special rules for `?m`:

```
\DeclareFontSeriesChangeRule {bc}{?m}{b}{}
\DeclareFontSeriesChangeRule {bec}{?m}{b}{}
\DeclareFontSeriesChangeRule {bsc}{?m}{b}{}
\DeclareFontSeriesChangeRule {bsc}{?m}{b}{}
\DeclareFontSeriesChangeRule {bx}{?m}{b}{}
\DeclareFontSeriesChangeRule {b}{?m}{b}{}
\DeclareFontSeriesChangeRule {c}{?m}{m}{}
\DeclareFontSeriesChangeRule {ebc}{?m}{eb}{}
\DeclareFontSeriesChangeRule {ebec}{?m}{eb}{}
\DeclareFontSeriesChangeRule {ebsc}{?m}{eb}{}
\DeclareFontSeriesChangeRule {ebsc}{?m}{eb}{}
\DeclareFontSeriesChangeRule {ebx}{?m}{eb}{}
\DeclareFontSeriesChangeRule {eb}{?m}{eb}{}
\DeclareFontSeriesChangeRule {ec}{?m}{m}{}
\DeclareFontSeriesChangeRule {elc}{?m}{el}{}
\DeclareFontSeriesChangeRule {elec}{?m}{el}{}
\DeclareFontSeriesChangeRule {elsc}{?m}{el}{}
\DeclareFontSeriesChangeRule {elsc}{?m}{el}{}
\DeclareFontSeriesChangeRule {elx}{?m}{el}{}
\DeclareFontSeriesChangeRule {el}{?m}{el}{}
\DeclareFontSeriesChangeRule {lc}{?m}{l}{}
\DeclareFontSeriesChangeRule {lec}{?m}{l}{}
\DeclareFontSeriesChangeRule {lsc}{?m}{l}{}
\DeclareFontSeriesChangeRule {lsc}{?m}{l}{}
\DeclareFontSeriesChangeRule {lx}{?m}{l}{}
\DeclareFontSeriesChangeRule {l}{?m}{l}{}
\DeclareFontSeriesChangeRule {m}{?m}{m}{}
\DeclareFontSeriesChangeRule {sbc}{?m}{sb}{}
\DeclareFontSeriesChangeRule {sbec}{?m}{sb}{}
\DeclareFontSeriesChangeRule {sbsc}{?m}{sb}{}
\DeclareFontSeriesChangeRule {sbsc}{?m}{sb}{}
\DeclareFontSeriesChangeRule {sbx}{?m}{sb}{}
\DeclareFontSeriesChangeRule {sb}{?m}{sb}{}
\DeclareFontSeriesChangeRule {sc}{?m}{m}{}
\DeclareFontSeriesChangeRule {sc}{?m}{m}{}
\DeclareFontSeriesChangeRule {slc}{?m}{sl}{}
\DeclareFontSeriesChangeRule {slec}{?m}{sl}{}
\DeclareFontSeriesChangeRule {slsc}{?m}{sl}{}
\DeclareFontSeriesChangeRule {slsc}{?m}{sl}{}
\DeclareFontSeriesChangeRule {slx}{?m}{sl}{}
\DeclareFontSeriesChangeRule {sl}{?m}{sl}{}
\DeclareFontSeriesChangeRule {ubc}{?m}{ub}{}
\DeclareFontSeriesChangeRule {ubec}{?m}{ub}{}
\DeclareFontSeriesChangeRule {ubsc}{?m}{ub}{}
\DeclareFontSeriesChangeRule {ubsc}{?m}{ub}{}
\DeclareFontSeriesChangeRule {ubx}{?m}{ub}{}
\DeclareFontSeriesChangeRule {ub}{?m}{m}{}
\DeclareFontSeriesChangeRule {ulc}{?m}{ul}{}
\DeclareFontSeriesChangeRule {ulec}{?m}{ul}{}
\DeclareFontSeriesChangeRule {ulsc}{?m}{ul}{}
\DeclareFontSeriesChangeRule {ulsc}{?m}{ul}{}
```

```
\DeclareFontSeriesChangeRule {ulx}{?m}{ul}{}
\DeclareFontSeriesChangeRule {ul}{?m}{ul}{}
\DeclareFontSeriesChangeRule {x}{?m}{m}{}
```

28.3 Changing to a new series

\if@forced@series

If the series gets forced we need to know that fact later on.

```
\newif\if@forced@series
```

\fontseriesforce

To change unconditionally to a new series you can use \fontseriesforce. If course, if the series doesn't exist for the current family substitution still happens, but there is not dependency on the current series.

```
\DeclareRobustCommand\fontseriesforce[1]{\@forced@seriestrue\edef\f@series{#1}}
```

\fontseries

The \fontseries command takes one argument which is the requested new font series. In the orginal 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.

```
\DeclareRobustCommand\fontseries[1]{\@forced@seriesfalse\merge@font@series{#1}}
```

\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.

```
\def\merge@font@series#1{%
  \expandafter\expandafter\expandafter
  \merge@font@series@
    \csname series@\f@series @#1\endcsname
    {#1}%
    \@nil
}
```

\merge@font@series@

This now defines the new \f@series:

```
\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.

```
%\ifcat\expandafter X\detokenize{#1}X%
  \def\reserved@a{#3}%
  \ifx\reserved@a\@empty
    \set@target@series{#2}%
  \else
```

Otherwise we check if the desired result for the series (#1) exists for the font family and the current shape. As the .fd is perhaps not loaded yet, we first have to do that, otherwise the test would fail even if the face is actually available.

```
    \maybe@load@fontshape
    \edef\reserved@a{\f@encoding /\f@family /#1/\f@shape}%
     \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).

```
       \set@target@series{#1}%
```

If not, then we try the alternate result (#2).

```
    \else
       \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.

```
         \set@target@series{#2}%
         \@font@shape@subst@warning
```

If that doesn't exist either, then we use the requested series unmodified (again with a warning).

```
       \else
         \set@target@series{#3}%
         \@font@shape@subst@warning
       \fi
    \fi
  \fi
}
```

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 `\@reserveda` above.

```
\def\@font@shape@subst@warning{%
   \edef\reserved@b{\curr@fontshape}%
   \ifx\reserved@a\reserved@b \else
     \@font@warning{Font shape `\reserved@a' undefined\MessageBreak
                    using `\reserved@b' instead}%
   \fi
}
```

`\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).

```
\def\maybe@load@fontshape{\begingroup\try@load@fontshape\endgroup}
```

`\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.

```
\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 andassign the result to `\f@series`.

```
    \edef\f@series{#1}%
    \expandafter\series@maybe@drop@one@m\expandafter{\f@series}\f@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 it 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.

```
\def\series@maybe@drop@one@m#1#2{%
```

The code below is an inline version of the `\in@` macro without the group, so that it works in `\accent`.

```
  \def\in@@##1,#1,{}%
  \series@check@toks\expandafter{\in@@
    ,ulm,elm,lm,slm,mm,sbm,bm,ebm,ubm,muc,mec,mc,msc,msx,mx,mex,mux,{}{},#1,}%
  \edef\in@@{\the\series@check@toks}%
  \ifx\in@@\@empty
    \def#2{#1}%
  \else
    \edef#2{\expandafter\series@drop@one@m #1m\series@drop@one@m}%
  \fi
}
```

As a precaution we use a private toks register not `\toks@` as that is no longer hidden inside the group.

```
\newtoks\series@check@toks
```

\series@drop@one@m Drop up to two `m`s 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 istelf and we added one after it, so all we have to do is now returning `#1#2` and dropping the rest.

```
\def\series@drop@one@m#1m#2m#3\series@drop@one@m{%
%  \ifx\relax#1#2\relax m\else#1#2\fi
   #1#2%
}
```

29 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).

\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).

```
\def\DeclareFontShapeChangeRule #1#2#3#4{%
  \@namedef{shape@#1@#2}{{#3}{#4}}}
```

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`
`\textulc`
`\ulcdefault`

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).

```
\DeclareRobustCommand\ulcshape
        {\not@math@alphabet\ulcshape\relax
         \fontshape\ulcdefault\selectfont}
\newcommand\ulcdefault{ulc}
```

`\swshape`
`\textsw`
`\swdefault`

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.

```
\DeclareRobustCommand\swshape
        {\not@math@alphabet\swshape\relax
         \fontshape\swdefault\selectfont}
\newcommand\swdefault{sw}
```

`\sscshape`
`\textssc`
`\sscdefault`

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`.

```
\DeclareRobustCommand\sscshape
        {\not@math@alphabet\sscshape\relax
         \fontshape\sscdefault\selectfont}
\newcommand\sscdefault{ssc}
```

29.1 Mapping rules for shape combinations

Many of the entries are commented out as we will get that result without any entry.

```
%\DeclareFontShapeChangeRule {n}{n}   {n}   {}
\DeclareFontShapeChangeRule {n}{it}  {it}  {sl}
\DeclareFontShapeChangeRule {n}{sl}  {sl}  {it}
%\DeclareFontShapeChangeRule {n}{sw}  {sw}  {}
%\DeclareFontShapeChangeRule {n}{sc}  {sc}  {}
\DeclareFontShapeChangeRule {n}{ulc} {n}   {}
\DeclareFontShapeChangeRule {n}{up} {n}    {}

%\DeclareFontShapeChangeRule {it}{n}   {n}     {}
%\DeclareFontShapeChangeRule {it}{it}  {it}    {}
\DeclareFontShapeChangeRule {it}{sl}  {sl}    {it}
%\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.

```
\DeclareFontShapeChangeRule {it}{sc}  {scit}  {scsl}
\DeclareFontShapeChangeRule {it}{ulc} {it}    {}
\DeclareFontShapeChangeRule {it}{up} {n}      {}

%\DeclareFontShapeChangeRule {sl}{n}   {n}     {}
\DeclareFontShapeChangeRule {sl}{it}  {it}    {sl}
%\DeclareFontShapeChangeRule {sl}{sl}  {sl}    {}
%\DeclareFontShapeChangeRule {sl}{sw}  {sw}    {}
\DeclareFontShapeChangeRule {sl}{sc}  {scsl}  {scit}
\DeclareFontShapeChangeRule {sl}{ulc} {sl}    {}
\DeclareFontShapeChangeRule {sl}{up} {n}      {}

%\DeclareFontShapeChangeRule {sc}{n}   {n}     {}
\DeclareFontShapeChangeRule {sc}{it}  {scit}  {scsl}
\DeclareFontShapeChangeRule {sc}{sl}  {scsl}  {scit}
\DeclareFontShapeChangeRule {sc}{sw}  {scsw}  {sw}
%\DeclareFontShapeChangeRule {sc}{sc}  {sc}    {}
\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 hat 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 odditywhic I would rather liked to have avoided.

```
%\DeclareFontShapeChangeRule {sc}{up} {sc}     {}
\DeclareFontShapeChangeRule {sc}{up} {n}     {}

%\DeclareFontShapeChangeRule {scit}{n}   {n}    {}
\DeclareFontShapeChangeRule {scit}{it}  {scit}  {}
\DeclareFontShapeChangeRule {scit}{sl}  {scsl}  {scit}
\DeclareFontShapeChangeRule {scit}{sw}  {scsw}  {sc}    % or scit?
\DeclareFontShapeChangeRule {scit}{sc}  {scit}  {}
\DeclareFontShapeChangeRule {scit}{ulc} {it}    {}
\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 aternative, 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.

```
%\DeclareFontShapeChangeRule {scsl}{n}   {n}     {}
\DeclareFontShapeChangeRule {scsl}{it}  {scit}  {scsl}
\DeclareFontShapeChangeRule {scsl}{sl}  {scsl}  {}
\DeclareFontShapeChangeRule {scsl}{sw}  {scsw}  {sc}    % or scsl?
```

```
\DeclareFontShapeChangeRule {scsl}{sc}  {scsl}  {}
\DeclareFontShapeChangeRule {scsl}{ulc} {sl}    {}
\DeclareFontShapeChangeRule {scsl}{up}   {sc}   {}
```

```
%\DeclareFontShapeChangeRule {scsw}{n}   {n}     {}
\DeclareFontShapeChangeRule {scsw}{it}  {scit}  {scsw}
\DeclareFontShapeChangeRule {scsw}{sl}  {scsl}  {}
\DeclareFontShapeChangeRule {scsw}{sw}  {scsw}  {}
\DeclareFontShapeChangeRule {scsw}{sc}  {scsw}  {}
\DeclareFontShapeChangeRule {scsw}{ulc} {sw}    {}
\DeclareFontShapeChangeRule {scsw}{up} {sc}     {}
```

```
%\DeclareFontShapeChangeRule {sw}{n}   {n}     {}
%\DeclareFontShapeChangeRule {sw}{it}  {it}    {}
%\DeclareFontShapeChangeRule {sw}{sl}  {sl}    {}
%\DeclareFontShapeChangeRule {sw}{sw}  {sw}    {}
\DeclareFontShapeChangeRule {sw}{sc}  {scsw}  {}
\DeclareFontShapeChangeRule {sw}{ulc} {sw}    {}
\DeclareFontShapeChangeRule {sw}{up} {n}      {}
```

29.2 Changing to a new shape

\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.

```
\DeclareRobustCommand\fontshape[1]{\merge@font@shape{#1}}
```

\fontshapeforce

The unconditional version:

```
\DeclareRobustCommand\fontshapeforce[1]{\edef\f@shape{#1}}
```

\merge@font@shape

Look up the database entry (if existing) and act accordingly.

```
\def\merge@font@shape#1{%
  \expandafter\expandafter\expandafter
  \merge@font@shape@
    \csname shape@\f@shape @#1\endcsname
    {#1}%
    \@nil
}
```

\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.

```
\def\merge@font@shape@#1#2#3\@nil{%
  \def\reserved@a{#3}%
  \ifx\reserved@a\@empty
    \edef\f@shape{#2}%
  \else
    \maybe@load@fontshape
    \edef\reserved@a{\f@encoding /\f@family /\f@series/#1}%
     \ifcsname \reserved@a\endcsname
      \edef\f@shape{#1}%
    \else
      \ifcsname \f@encoding /\f@family /\f@series/#2\endcsname
        \edef\f@shape{#2}%
        \@font@shape@subst@warning
      \else
        \edef\f@shape{#3}%
        \@font@shape@subst@warning
      \fi
   \fi
 \fi
}
```

\normalshape

\normalshape resets both sub-axes if the default rules are used.

```
\protected\def\normalshape
    {\not@math@alphabet\normalshape\relax
      \fontshape\shapedefault\selectfont}%
```

30 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.

```
\def\reinstall@nfss@defs{%
  \protected\def\upshape
          {\not@math@alphabet\upshape\relax
           \fontshape\updefault\selectfont}%
  \protected\def\slshape
          {\not@math@alphabet\slshape\relax
           \fontshape\sldefault\selectfont}%
  \protected\def\scshape
          {\not@math@alphabet\scshape\relax
           \fontshape\scdefault\selectfont}%
  \protected\def\itshape
          {\not@math@alphabet\itshape\mathit
           \fontshape\itdefault\selectfont}%
  \protected\def\ulcshape
          {\not@math@alphabet\ulcshape\relax
           \fontshape{ulc}\selectfont}%
  \protected\def\swshape
          {\not@math@alphabet\swshape\relax
           \fontshape\swdefault\selectfont}%
  \protected\def\sscshape
          {\not@math@alphabet\sscshape\relax
           \fontshape\sscdefault\selectfont}%
}
```

Supporting rollback . . .

```
⟨/2ekernel | latexrelease⟩
⟨latexrelease⟩\EndIncludeInRelease
⟨latexrelease⟩\IncludeInRelease{0000/00/00}%
⟨latexrelease⟩    {\DeclareFontSeriesChangeRule}{Series change rules}%
```

```
⟨latexrelease⟩
⟨latexrelease⟩\DeclareRobustCommand\fontseries[1]{\edef\f@series{#1}}
⟨latexrelease⟩\DeclareRobustCommand\fontshape [1]{\edef\f@shape{#1}}
⟨latexrelease⟩\let\fontseriesforce\@undefined
⟨latexrelease⟩\let\fontshapeforce\@undefined
⟨latexrelease⟩
⟨latexrelease⟩\let\DeclareFontSeriesChangeRule\@undefined
⟨latexrelease⟩\let\merge@font@series\@undefined
⟨latexrelease⟩\let\merge@font@series@\@undefined
⟨latexrelease⟩\let\@font@shape@subst@warning\@undefined
⟨latexrelease⟩\let\maybe@load@fontshape\@undefined
⟨latexrelease⟩\let\set@target@series\@undefined
⟨latexrelease⟩\let\series@maybe@drop@one@m\@undefined
⟨latexrelease⟩\let\series@drop@one@m\@undefined
⟨latexrelease⟩\let\DeclareFontShapeChangeRule\@undefined
⟨latexrelease⟩\let\merge@font@shape\@undefined
⟨latexrelease⟩\let\merge@font@shape@\@undefined
⟨latexrelease⟩\let\normalshape\@undefined
⟨latexrelease⟩\let\ulcshape\@undefined
⟨latexrelease⟩\let\ulcdefault\@undefined
⟨latexrelease⟩\let\swshape\@undefined
⟨latexrelease⟩\let\swdefault\@undefined
⟨latexrelease⟩\let\sscshape\@undefined
⟨latexrelease⟩\let\sscdefault\@undefined
⟨latexrelease⟩\let\normalshape\@undefined
```

This is always called in `\document` so don't make it undefined.

```
⟨latexrelease⟩
⟨latexrelease⟩\let\reinstall@nfss@defs\relax
⟨latexrelease⟩\EndIncludeInRelease
⟨*2ekernel⟩
⟨/2ekernel⟩
```

File q

ltfsstrc.dtx

31 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 LaTeX will stop.

32 A driver for this document

The next bit of code contains the documentation driver file for TeX, 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 LaTeX this will produce the documentation as well.

```
⟨*driver⟩
\documentclass{ltxdoc}


%\OnlyDescription  % comment out for implementation details

\begin{document}
   \DocInput{ltfsstrc.dtx}
\end{document}
⟨/driver⟩
```

33 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.

```
⟨*package⟩
%\NeedsTeXFormat{LaTeX2e}
%\ProvidesPackage{tracefnt}[??/??/?? v?.??
%                           Standard LaTeX package (font tracing)]
⟨/package⟩
```

The `debug` module makes use of commands contained in a special package file named `trace.sty`.[4]

```
⟨+debug⟩ \input trace.sty
```

34 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.

```
⟨*2ekernel⟩
\def\tracingfonts{%
  \@font@warning{Command \noexpand\tracingfonts
           not provided.\MessageBreak
           Use the ‘tracefnt’ package.\MessageBreak Command found:}%
       \count@}
⟨/2ekernel⟩
```

The `\count@` in the line above will remove the number after `\tracingfonts`. Note that this definition will be overwritten be the next line if one of these modules are included.

```
⟨*package, trace, debug⟩
\newcount\tracingfonts
\tracingfonts=0
⟨/package, trace, debug⟩
```

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.

```
⟨*package⟩
\DeclareOption{errorshow}{%
   \def\@font@info#1{%
         \GenericInfo{(Font)\@spaces\@spaces\@spaces\space\space}%
```

[4] This 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 artefacts.

```
                     {LaTeX Font Info: \space\space\space#1}}%
    \def\@font@warning#1{%
         \GenericInfo{(Font)\@spaces\@spaces\@spaces\space\space}%
                        {LaTeX Font Warning: #1}}%
     }
\DeclareOption{warningshow}{%
   \def\@font@info#1{%
         \GenericInfo{(Font)\@spaces\@spaces\@spaces\space\space}%
                     {LaTeX Font Info: \space\space\space#1}}%
    \def\@font@warning#1{%
         \GenericWarning{(Font)\@spaces\@spaces\@spaces\space\space}%
                        {LaTeX Font Warning: #1}}%
     }
\DeclareOption{infoshow}{%
   \def\@font@info#1{%
         \GenericWarning{(Font)\@spaces\@spaces\@spaces\space\space}%
                     {LaTeX Font Info: \space\space\space#1}}%
    \def\@font@warning#1{%
         \GenericWarning{(Font)\@spaces\@spaces\@spaces\space\space}%
                        {LaTeX Font Warning: #1}}%
     }
\DeclareOption{loading}{%
    \tracingfonts\tw@
   }
\DeclareOption{debugshow}{%
    \ExecuteOptions{infoshow}%
    \tracingfonts\thr@@
   }
\DeclareOption{pausing}{%
    \def\@font@warning#1{%
      \GenericError
             {(Font)\@spaces\@spaces\@spaces\space\space}%
             {LaTeX Font Warning: #1}%
             {See the LaTeX Companion for details.}%
             {I'll stop for every LaTeX Font Warning because
              you requested\MessageBreak the ‘pausing’ option
              to the tracefnt package.}}%
   }
```

We make `infoshow` the default, which in turn defines `\font@warning` and `\font@info`.

```
\ExecuteOptions{infoshow}
\ProcessOptions
⟨/package⟩
```

We also need a default definition inside the kernel:

```
⟨*2ekernel⟩
\def\@font@info#1{%
         \GenericInfo{(Font)\@spaces\@spaces\@spaces\space\space}%
                     {LaTeX Font Info: \space\space\space#1}}%
\def\@font@warning#1{%
         \GenericWarning{(Font)\@spaces\@spaces\@spaces\space\space}%
```

```
                              {LaTeX Font Warning: #1}}%
⟨/2ekernel⟩
```

35 Macros common to fam.tex and tracefnt.sty

In the first versions of `tracefnt.dtx` some macros of `fam.dtx`[5] were redefined to included 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 `ltfss.dtx`.

35.1 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.

```
⟨*2ekernel | package⟩
\def\extract@font{%
   \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`).

```
   \global\expandafter\font\font@name\external@font\relax
```

When tracing we typeout the internal and external font name.

```
⟨*trace⟩
   \ifnum \tracingfonts >\@ne
   \@font@info{External font `\external@font'
               loaded as\MessageBreak \font@name}\fi
⟨/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.

```
   \font@name \relax
```

The next two lines execute the "loading actions" for the family and then for the individual font shape.

```
   \csname \f@encoding+\f@family\endcsname
   \csname\curr@fontshape\endcsname
   \relax
      }
⟨/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.

`\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`.

```
⟨*2ekernel⟩
\def\get@external@font{%
```

[5] This file is currently not distributed in documented form. Its code is part of `ltfss.dtx`.

We don't know the external font name at the beginning.

```
   \let\external@font\@empty
   \edef\font@info{\expandafter\expandafter\expandafter\string
        \csname \curr@fontshape \endcsname}%
   \try@size@range
```

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.

```
   \ifx\external@font\@empty
      \try@size@substitution
      \ifx\external@font\@empty
         \@latex@error{Font \expandafter \string\font@name\space
                     not found}\@eha
         \error@fontshape
         \get@external@font
   \fi\fi
}
⟨/2ekernel⟩
```

`\selectfont` The macro `\selectfont` is called whenever a font change must take place.

```
⟨*2ekernel | package⟩
\DeclareRobustCommand\selectfont
        {%
```

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.

```
⟨+debug⟩  \pushtracing
⟨+debug⟩  \ifnum\tracingfonts<4 \tracingoff
⟨+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`.

```
    \ifx\f@linespread\baselinestretch \else
      \set@fontsize\baselinestretch\f@size\f@baselineskip \fi
```

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 LaTeX's `\twfbf`, etc. We define `\font@name` globally, as always. The reason for this is explained later on.

```
    \xdef\font@name{%
      \csname\curr@fontshape/\f@size\endcsname}%
```

We call the macro `\pickup@font` which will load the font if necessary.

```
    \pickup@font
```

Then we select the font.

```
    \font@name
```

If `\tracingfonts` is greater than 2 we also show the font switch. We do this before `\glb@settings` is called since this macro might redefine `\font@name`.

```
⟨*trace⟩
```

```
    \ifnum \tracingfonts>\tw@
      \@font@info{Switching to \font@name}\fi
⟨/trace⟩
```

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.

```
    \size@update
```

A similar function is called to handle anything related to encoding updates. This one is changed from `\relax` by `\fontencoding`.

```
    \enc@update
```

Just before ending this macro we have to pop the tracing stack if it was pushed before.

```
⟨+debug⟩ \poptracing
    }
```

`\set@fontsize`

The macro `\set@fontsize` does the actual work. First it assigns new values to `\f@size`, `\f@baselineskip` and `\f@linespread`.

```
\def\set@fontsize#1#2#3{%
    \@defaultunits\@tempdimb#2pt\relax\@nnil
    \edef\f@size{\strip@pt\@tempdimb}%
    \@defaultunits\@tempskipa#3pt\relax\@nnil
    \edef\f@baselineskip{\the\@tempskipa}%
    \edef\f@linespread{#1}%
```

For backward compatibility and for later testing within `\selectfont` the internal value of `\f@linespread` is passed back to `\baselinestretch`.

```
    \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:

```
      \def\size@update{%
```

First calculate the new `\baselineskip` and also store it in `normalbaselineskip`

```
        \baselineskip\f@baselineskip\relax
        \baselineskip\f@linespread\baselineskip
        \normalbaselineskip\baselineskip
```

then to set up a new `\strutbox`

```
        \setbox\strutbox\hbox{%
          \vrule\@height.7\baselineskip
                \@depth.3\baselineskip
                \@width\z@}%
```

We end with a bit of tracing information.

```
⟨*trace⟩
   \ifnum \tracingfonts>\tw@
      \ifx\f@linespread\@empty
        \let\reserved@a\@empty
      \else
        \def\reserved@a{\f@linespread x}%
      \fi
      \@font@info{Changing size to \f@size/\reserved@a
                \f@baselineskip}%
      \aftergroup\type@restoreinfo \fi
⟨/trace⟩
```

When all this is processed \size@update redefines itself to \relax so that in later calls of \selectfont no extra code will be executed.

```
        \let\size@update\relax}%
  }
```

Instead of defining this macro internally we might speed things up by placing the code into a separate macro and use \let!

\size@update Normally this macro does nothing; it will be redefined by \set@fontsize to initiate an update.

```
\let\size@update\relax
```

\type@restoreinfo This macro produces some info when a font size and/or baseline change will get restored.

```
⟨*trace⟩
   \def\type@restoreinfo{%
      \ifx\f@linespread\@empty
        \let\reserved@a\@empty
      \else
        \def\reserved@a{\f@linespread x}%
      \fi
      \@font@info{Restoring size to
               \f@size/\reserved@a\f@baselineskip}}
⟨/trace⟩
```

\glb@settings \glb@currsize The macro \glb@settings globally selects all math fonts for the current size if necessary.

```
\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.

```
     \expandafter\ifx\csname S@\f@size\endcsname\relax
        \calculate@math@sizes
     \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.

```
     \csname S@\f@size\endcsname
     \ifmath@fonts
⟨*trace⟩
       \ifnum \tracingfonts>\tw@
         \@font@info{Setting up math fonts for
                     \f@size/\f@baselineskip}\fi
⟨/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.

```
      \begingroup
        \escapechar\m@ne
        \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@currsize` is set equal to `\f@size`. This signals that the fonts for math in this size are set up.

```
        \globaldefs\@ne
        \math@fonts
        \let \glb@currsize \f@size
      \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.

```
       \the\every@math@size
```

Otherwise we announce that the math fonts are not set up for this size.

```
⟨*trace⟩
      \else
        \ifnum \tracingfonts>\tw@
          \@font@info{No math setup for
                    \f@size/\f@baselineskip}\fi
⟨/trace⟩
    \fi
}
⟨/2ekernel | package⟩
```

`\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).

```
⟨*2ekernel⟩
\def\baselinestretch{1}
```

`\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.

```
\newtoks\every@math@size
\every@math@size={}
⟨/2ekernel⟩
```

35.2 Math fonts setup

35.2.1 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 \mbox{ \small for all $b$ and $c\in Z$}$
```

Here the inner formulae `b` and `c\in 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 `\gbl@currsize` 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 `\gbl@settings` which changes the math font setup and updates `\gbl@currsize`.

    (a) If we are recursively inside another formula (`\if@inmath`) we ensure that `\gbl@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 typesetable. 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 × ⟨non-math levels⟩ per inner formula).

35.2.2 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.

```
⟨*2ekernel | package⟩
\def\check@mathfonts{%
  \ifx \glb@currsize \f@size
⟨*trace⟩
    \ifnum \tracingfonts>\thr@@
        \@font@info{*** MATH: no change \f@size\space
         curr/global (\curr@math@size/\glb@currsize)}\fi
⟨/trace⟩
  \else
⟨*trace⟩
    \ifnum \tracingfonts>\thr@@
        \@font@info{*** MATH: setting up \f@size\space
         curr/global (\curr@math@size/\glb@currsize)}\fi
⟨/trace⟩
     \glb@settings
     \init@restore@glb@settings
  \fi
  \let\curr@math@size\f@size
  \def\init@restore@glb@settings{\aftergroup\restglb@settings}%
}
```

\init@restore@glb@settings

This macros does by default nothing but get redefined inside \check@mathfonts to initiate fontsize restoring in nested formulas.

```
⟨-trace⟩\let\init@restore@glb@settings\relax
⟨*trace⟩
\def\init@restore@glb@settings{%
       \ifnum \tracingfonts>\thr@@
        \@font@info{*** MATH: no resetting (not in
                    nested math)}\fi
}
⟨/trace⟩
```

\restglb@settings

This macro will be executed the first time after the current formula.

```
\def\restglb@settings{%
⟨*trace⟩
    \ifnum \tracingfonts>\thr@@
        \@font@info{*** MATH: restoring}\fi
⟨/trace⟩
      \begingroup
        \let\f@size\curr@math@size
        \ifx\glb@currsize \f@size
⟨*trace⟩
    \ifnum \tracingfonts>\thr@@
        \@font@info{*** MATH: ... already okay (\f@size)}\fi
⟨/trace⟩
        \else
⟨*trace⟩
    \ifnum \tracingfonts>\thr@@
        \@font@info{*** MATH: ... to \f@size}\fi
⟨/trace⟩
```

```
          \glb@settings
        \fi
      \endgroup
}
```

35.2.3 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.

```
\def\use@mathgroup#1#2{\relax\ifmmode
```

```
⟨*trace⟩
  \ifnum \tracingfonts>\tw@
    \count@#2\relax
    \@font@info{Using \noexpand\mathgroup
               (\the\count@) #2}\fi
⟨/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.

```
     \math@bgroup
         \expandafter\ifx\csname M@\f@encoding\endcsname#1\else
         #1\fi
         \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.

```
     \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 $\mathcal{AMS}$-TeX macros for placing accents.

\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.

```
⟨*trace⟩
  \ifx\math@bgroup\bgroup
    \def\math@egroup#1{#1\egroup
```

```
     \ifnum \tracingfonts>\tw@
    \@font@info{Restoring \noexpand\mathgroup
       (\ifnum\mathgroup=\m@ne default\else \the\mathgroup \fi)%
       }\fi}
  \fi
⟨/trace⟩
```

\getanddefine@fonts

`\getanddefine@fonts` has two arguments: the ⟨*math group number*⟩ and the *family/series/shape* name as a control sequence.

```
\def\getanddefine@fonts#1#2{%
```

First we turn of tracing when `\tracingfonts` is less than 4.

```
⟨+debug⟩  \pushtracing
⟨+debug⟩  \ifnum\tracingfonts<4 \tracingoff
⟨+debug⟩  \else \tracingon\getanddefine@fonts \fi

⟨*trace⟩
  \ifnum \tracingfonts>\tw@
  \count@#1\relax
    \@font@info{\noexpand\mathgroup (\the\count@) #1 :=\MessageBreak
                \string#2 \tf@size/\sf@size/\ssf@size}\fi
⟨/trace⟩
```

We append the current `\tf@size` to `#2` to obtain the font name.[6] Again, `font@name` is defined globally, for the reasons explained in the description of `\wrong@fontshape`.

```
  \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`.

```
  \pickup@font \let\textfont@name\font@name
```

Same game for `\scriptfont` and `\scriptscriptfont`:

```
  \xdef\font@name{\csname \string#2/\sf@size\endcsname}%
  \pickup@font \let\scriptfont@name\font@name
  \xdef\font@name{\csname \string#2/\ssf@size\endcsname}%
  \pickup@font
```

Then we append the new `\textfont`... assignments to the `\math@fonts`.

```
  \edef\math@fonts{\math@fonts
               \textfont#1\textfont@name
               \scriptfont#1\scriptfont@name
               \scriptscriptfont#1\font@name}%
```

Just before ending this macro we have to pop the tracing stack if it was pushed before.

```
⟨+debug⟩ \poptracing
    }
⟨/2ekernel | package⟩
```

[6] One 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.

36 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 it argument.

```
⟨*2ekernel⟩
\def\ifnot@nil#1{\def\reserved@a{#1}%
  \ifx\reserved@a\@nnil \expandafter\@gobble
  \else \expandafter\@firstofone\fi}
```

\remove@to@nnil
\remove@angles
\remove@star

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.

```
\def\remove@to@nnil#1\@nnil{}
\def\remove@angles#1>{\set@simple@size@args}
\def\remove@star#1*{#1}
```

\extract@sizefn

This macro takes a size specification and parses it into size function and the optional and mandatory arguments.

```
\def\extract@sizefn#1*#2\@nil{%
  \if>#2>\set@size@funct@args#1\@nil
         \let\sizefn@info\@empty
  \else\expandafter\set@size@funct@args\remove@star#2\@nil
       \def\sizefn@info{#1}\fi
  }
```

\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 `#3` 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`.

```
% % this could be replaced by \try@size@range making the subst slower!
\def\try@simple@size{%
```

`\reserved@a` is made an abbreviation for the head of the definition of the macro `\extract@fontinfo`.

```
    \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.

```
    \expandafter\reserved@a\expandafter<\f@size>##2<##3\@nnil{%
         \ifnot@nil{##2}%
```

```
            {\set@simple@size@args##2<##3\@nnil
             \execute@size@function\sizefn@info
            }}%
```

Now we call `\extract@fontinfo`. Note the `<\@nil tokens` at the end.

```
  \expandafter\expandafter
  \expandafter\extract@fontinfo\expandafter\font@info
  \expandafter<\f@size>\@nil<\@nnil
}
```

`\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,

```
\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.

```
          \if<#1<%
            \expandafter\remove@angles
          \else
            \extract@sizefn#1*\@nil
            \expandafter\remove@to@nnil
          \fi}
```

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 macors 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.

```
\def\extract@rangefontinfo#1<#2>{%
       \is@range#2->\@nil#2>}
```

`\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.

```
\def\is@range#1-#2\@nil{%
   \if>#2\expandafter\check@single\else
      \expandafter\check@range\fi}
```

\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.

```
\def\check@range#1-#2>#3<#4\@nnil{%
  \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.

```
     \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.

```
     \upper@bound0#2\p@
     \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.

```
     \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.

```
       \lower@bound0#1\p@
       \ifdim \f@size \p@<\lower@bound
        \else
```

If both tests are passed we can try executing the size function.

```
          \set@simple@size@args#3<#4\@nnil
          \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.

```
         \ifx\external@font\@empty
         \else
           \let\reserved@f\@empty
         \fi
       \fi
     \fi
     \reserved@f}}
```

\lower@bound
\upper@bound

We use two dimen registers \lower@bound and \upper@bound to store the lower and upper endpoints of the range we found.

```
\newdimen\lower@bound
\newdimen\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).

```
\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.

```
     \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.

```
     \ifdim \f@size \p@=#1\p@
```

Otherwise if this test is passed we can try executing the size function.

```
        \set@simple@size@args#2<#3\@nnil
        \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.

```
        \ifx\external@font\@empty
        \else
          \let\reserved@f\@empty
        \fi
     \fi
     \reserved@f}
```

`\set@size@funct@args`
`\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`.

```
\def\set@size@funct@args{\@ifnextchar[%
  \set@size@funct@args@{\set@size@funct@args@[]}}
```

```
\def\set@size@funct@args@[#1]#2\@nil{%
  \def\mandatory@arg{#2}%
  \def\optional@arg{#1}}
⟨/2ekernel⟩
```

`\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 `<...>`.

```
⟨*2ekernel⟩
\def\DeclareSizeFunction#1#2{\@namedef{s@fct@#1}{#2}}
\@onlypreamble\DeclareSizeFunction
⟨/2ekernel⟩
```

`\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`).

```
⟨*2ekernel | package⟩
\def\execute@size@function#1{%
⟨*trace⟩
         \@ifundefined{s@fct@#1}%
           {\errmessage{Undefined font size function #1}%
```

```
            \s@fct@}%
           {\csname s@fct@#1\endcsname}%
⟨/trace⟩
⟨-trace⟩        \csname s@fct@#1\endcsname
}
⟨/2ekernel | package⟩
```

\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.

```
⟨*2ekernel⟩
\def\try@size@range{%
    \expandafter\extract@rangefontinfo\font@info <-*>\@nil<\@nnil
}
```

\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.

```
\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.

```
  \@tempdimb \maxdimen
  \let \best@size \@empty
```

Now we loop over the specification

```
  \expandafter \try@simples \font@info <\number\@M>\@nil<\@nnil
}
```

\font@submax
\fontsubfuzz 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).

```
\def\font@submax{0pt}
\def\fontsubfuzz{.4pt}
⟨/2ekernel⟩
⟨+package⟩\def\fontsubfuzz{0pt}
```

\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.

```
⟨*2ekernel⟩
\gdef\try@simples#1<#2>{%
  \tryif@simple#2->\tryif@simple}
```

\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.

```
\gdef\tryif@simple#1-#2\tryif@simple{%
```

Most common case for `\reserved@f` first:

```
  \let \reserved@f \try@simples
  \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.

```
    \dimen@ #1\p@
    \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.

```
      \ifdim \f@size\p@<\dimen@
        \@tempdimc \dimen@
        \advance\@tempdimc -\f@size\p@
      \else
        \@tempdimc \f@size\p@
        \advance\@tempdimc -\dimen@
      \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`.

```
      \ifdim \@tempdimc<\@tempdimb
        \@tempdimb \@tempdimc
        \def \best@size{#1}%
      \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`.

```
    \else
```

`\subst@size` This macro substitutes the size recorded in `\best@size` for the unavailable size `\f@size`. `\font@submax` records the maximum difference between desired size and selected size in the whole run.

```
% %\subst@size            %% coded inline
% %\def\subst@size{%
  \ifx \external@font\@empty
    \ifx \best@size\@empty
    \else
      \ifdim \@tempdimb>\font@submax \relax
        \xdef \font@submax {\the\@tempdimb}%
      \fi
      \let \f@user@size \f@size
      \let \f@size \best@size
      \ifdim \@tempdimb>\fontsubfuzz\relax
        \@font@warning{Font\space shape\space
            `\curr@fontshape'\space in\space size\space
             <\f@user@size>\space not\space available\MessageBreak
             size\space <\f@size>\space substituted}%
      \fi
      \try@simple@size
      \do@subst@correction
    \fi
 \fi
% %}
```

This brings us back into the main part of `\tryif@simple`. Finally we get rid of any rubbish left over on the input stack.

```
      \let \reserved@f \remove@to@nnil
    \fi
  \fi
```

If it's a range iterate also.

```
  \reserved@f}
```

36.1 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.

```
\DeclareSizeFunction{}{\empty@sfcnt\@font@warning}
\DeclareSizeFunction{s}{\empty@sfcnt\@font@info}
```

```
\def\empty@sfcnt#1{%
      \@tempdimb \f@size\p@
      \ifx\optional@arg\@empty
      \else
        \@tempdimb \optional@arg\@tempdimb
         #1{Font\space shape\space `\curr@fontshape'\space
            will\space be\MessageBreak
            scaled\space to\space size\space \the\@tempdimb}%
      \fi
      \edef\external@font{\mandatory@arg\space at\the\@tempdimb}}
```

`\s@fct@gen`
`\s@fct@sgen` 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.

```
\DeclareSizeFunction{gen}{\gen@sfcnt\@font@warning}
\DeclareSizeFunction{sgen}{\gen@sfcnt\@font@info}
```

```
\def\gen@sfcnt{%
      \edef\mandatory@arg{\mandatory@arg\f@size}%
      \empty@sfcnt}
```

`\s@fct@genb`
`\s@fct@sgenb` 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 no optional argument is present, and to `\f@size` multiplied by the optional argument otherwise.

```
\DeclareSizeFunction{genb}{\genb@sfcnt\@font@warning}
\DeclareSizeFunction{sgenb}{\genb@sfcnt\@font@info}
```

```
\def\genb@sfcnt{%
    \edef\mandatory@arg{\mandatory@arg\expandafter\genb@x\f@size..\@@}%
    \empty@sfcnt}
```

\genb@x \genb@y The auxiliary macros `\genb@x` and `\genb@y` are used to convert the `\f@size` into centipoints.

```
\def\genb@x#1.#2.#3\@@{\two@digits{#1}\genb@y#200\@@}
\def\genb@y#1#2#3\@@{#1#2}
```

\s@fct@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. documentation fixes

```
\DeclareSizeFunction{sub}{\sub@sfcnt\@font@warning}
\DeclareSizeFunction{ssub}{\sub@sfcnt\@font@info}
```

```
\def\sub@sfcnt#1{%
    \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.

```
    \begingroup
     \expandafter\split@name\mandatory@arg/\@nil
     \try@load@fontshape
    \endgroup
```

Then we record the current `\f@size` since it may get clobbered.

```
    \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`.

```
    \expandafter
    \ifx\csname\mandatory@arg\endcsname\relax
      \errmessage{No\space declaration\space for\space
                  shape\space \mandatory@arg}%
      \error@fontshape
    \else
```

Otherwise we warn the user about the substitution taking place.

```
      #1{Font\space shape\space `\curr@fontshape'\space in\space
         size\space <\f@size>\space not\space available\MessageBreak
         Font\space shape\space `\mandatory@arg'\space tried\space
         instead}%
      \expandafter\split@name\mandatory@arg/\@nil
    \fi
```

Then we restart the font specification scan by calling `\get@external@font`.

```
    \edef\f@size{\f@user@size}%
    \get@external@font
```

Finally `\do@subst@correction` is called to get the font name right.

```
    \do@subst@correction
}
```

\@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.

```
⟨/2ekernel⟩
⟨*2ekernel | latexrelease⟩
⟨latexrelease⟩\IncludeInRelease{2020/02/01}%
⟨latexrelease⟩                 {\@font@aliasinfo}{alias size function}%
\DeclareSizeFunction{alias}{\sub@sfcnt\@font@aliasinfo}
\def\@font@aliasinfo#1{%
  \@font@info{Font\space shape\space `\curr@fontshape'\space
              aliased\space to\MessageBreak `\mandatory@arg'}%
}
⟨/2ekernel | latexrelease⟩
⟨latexrelease⟩\EndIncludeInRelease
⟨latexrelease⟩\IncludeInRelease{0000/00/00}%
⟨latexrelease⟩                 {\@font@aliasinfo}{alias size function}%
⟨latexrelease⟩\let\s@fct@alias\@undefined
⟨latexrelease⟩\let\@font@aliasinfo\@undefined
⟨latexrelease⟩
⟨latexrelease⟩\EndIncludeInRelease
⟨*2ekernel⟩
```

\s@fct@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.

```
\DeclareSizeFunction{subf}{\subf@sfcnt\@font@warning}
\DeclareSizeFunction{ssubf}{\subf@sfcnt\@font@info}
```

```
\def\subf@sfcnt#1{%
      #1{Font\space shape\space `\curr@fontshape'\space in\space
         size\space \f@size\space not\space available\MessageBreak
        external\space font\space `\mandatory@arg'\space used}%
      \empty@sfcnt#1%
      }
```

\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.

```
\DeclareSizeFunction{fixed}{\fixed@sfcnt\@font@warning}
\DeclareSizeFunction{sfixed}{\fixed@sfcnt\@font@info}
```

```
\def\fixed@sfcnt#1{%
  \ifx\optional@arg\@empty
    \let\external@font\mandatory@arg
  \else
```

```
    \edef\external@font{\mandatory@arg\space at\optional@arg pt}%
  \fi
  #1{External\space font\space `\external@font'\space loaded\space
     for\space size\MessageBreak
     <\f@size>}%
}
⟨/2ekernel⟩
```

File r

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.

```
⟨*latexrelease⟩
\IncludeInRelease{2015/01/01}{\new@fontshape}%
                             {NFSS version1 commands}%
\let\new@fontshape\@undefined
\let\warn@rel@i\@undefined
\let\scan@fontshape\@undefined
\let\scan@@fontshape\@undefined
\let\subst@fontshape\@undefined
\let\extra@def\@undefined
\let\default@mextra\@undefined
\let\preload@sizes\@undefined
\let\err@rel@i\@undefined
\let\newmathalphabet\@undefined
\let\newmathalphabet@\@undefined
\let\newmathalphabet@@@\@undefined
\let\if@no@font@opt\@undefined
\let\@no@font@optfalse\@undefined
\let\define@mathalphabet\@undefined
\let\define@mathgroup\@undefined
\let\addtoversion\@undefined
\EndIncludeInRelease
```

In older releases we provide the original definitions.

```
\IncludeInRelease{0000/00/00}{\new@fontshape}%
                             {NFSS version1 commands}%
```

\new@fontshape The interface is now \DeclareFontShape.

```
\gdef\new@fontshape#1#2#3#4{%
     \warn@rel@i\new@fontshape\DeclareFontShape
     \expandafter\scan@fontshape\@gobble#4<\@nil><<%
     \DeclareFontShape U{#1}{#2}{#3}\reserved@f}%
\@onlypreamble\new@fontshape
```

\warn@rel@i The warning message used above.

```
\gdef\warn@rel@i#1#2{%
\@font@warning{***  NFSS release 1 command
               \noexpand#1found\MessageBreak
   ***  Update by using release 2 command
```

```
        \string#2.\MessageBreak
   ***  Recovery is probably possible}%
}%
\@onlypreamble\warn@rel@i
```

\scan@fontshape This will scan the old font shape definition syntax.

```
\gdef\scan@fontshape{%
  \let\reserved@f\@empty
  \let\reserved@e\@empty %        holds last info
  \scan@@fontshape
}%
\@onlypreamble\scan@fontshape
```

\scan@@fontshape

```
\gdef\scan@@fontshape#1>#2#3<{%
  \ifx\@nil#1%
    \edef\reserved@f{\reserved@f\reserved@e}%
  \else
    \def\reserved@b{#1}%       nick names
    \def\reserved@c{#3}%
    \in@{ at}{#3}%
    \ifin@
      \in@{pt}{#3}%  not a proof but a good chance
      \ifin@
```

We grap also everything after pt and discard it if people have forgotten to place a percent sign there.

```
        \def\reserved@a##1 at##2pt##3\@nil{%
           \def\reserved@b{##2}%
           \def\reserved@c{##1}%
           }%
        \reserved@a#3\@nil
      \fi
    \fi
    \ifnum 0<0#2
      \edef\reserved@d{subf*\reserved@c}%
      \ifcase #2\or
      \or
      \else
        \errmessage{*** What's this? NFSS release 0? ***}%
      \fi
    \else
      \edef\reserved@d{#2\reserved@c}%
    \fi
    \ifx\reserved@d\reserved@e
      \edef\reserved@f{\reserved@f<\reserved@b>}%
    \else
      \edef\reserved@f{\reserved@f\reserved@e<\reserved@b>}%add old info
      \let\reserved@e\reserved@d
    \fi
    \expandafter\scan@@fontshape
  \fi
}%
\@onlypreamble\scan@@fontshape
```

`\subst@fontshape` This is now also handled by the extend syntax of `\DeclareFontShape`.

```
\gdef\subst@fontshape#1#2#3#4#5#6{%
     \warn@rel@i\subst@fontshape\DeclareFontShape
     \DeclareFontShape{U}{#1}{#2}{#3}{<->sub*#4/#5/#6}{}}%
\@onlypreamble\subst@fontshape
```

`\extra@def` This was replaced by `\DeclareFontFamily`.

```
\gdef\extra@def#1#2#3{%
     \warn@rel@i\extra@def\DeclareFontFamily
     \DeclareFontFamily{U}{#1}{}%
}%
\@onlypreamble\extra@def
```

`\default@mextra` The new name is `\DeclareFontEncodingDefaults` but in this case we don't feel comfortable with this either.

```
\gdef\default@mextra{%
  \warn@rel@i\default@mextra\DeclareFontEncodingDefaults
```

We pick up the argument to `\default@mextra` implicitly as the second argument of `\DeclareFontEncodingDefaults`.

```
  \DeclareFontEncodingDefaults\relax
}%
\@onlypreamble\default@mextra
```

`\preload@sizes` The new interface is `\DeclarePreloadSizes`.

```
\gdef\preload@sizes{%
     \warn@rel@i\preload@sizes\DeclarePreloadSizes
     \DeclarePreloadSizes U%
}%
\@onlypreamble\preload@sizes
```

`\err@rel@i` This macro is used in cases where emulation with NFSS2 features is not really possible.

```
\gdef\err@rel@i#1#2{%
  \@latex@error{***  NFSS release 1 command \noexpand#1found%
          ^^J***  Recovery not possible. Use \string#2}%
       {The new release of NFSS doesn't support the
       \noexpand#1command^^Jany longer.
        Please upgrade your file to the syntax of NFSS
        release 2^^Jusing the \noexpand#2command.}%
```

Let's die.

```
  \batchmode\input.\relax
}%
\@onlypreamble\err@rel@i
```

`\newmathalphabet`
`\newmathalphabet@@`
`\newmathalphabet@@@`

`\newmathalphabet` is the old form.

```
\gdef\newmathalphabet{%
  \if@no@font@opt
    \@latex@error{*** NFSS release 1 command
                    \noexpand\newmathalphabet found%
     ^^J \space*** Automatic recovery not possible.%
     ^^J \space*** TYPE H for Help%
              }%
```

```
       {Please look at the file usrguide.tex for hints on
        how to resolve this problem.}%
  \else
     \warn@rel@i\newmathalphabet\DeclareMathAlphabet
  \fi
  \@ifstar\newmathalphabet@@@
          \newmathalphabet@@}%
\gdef\newmathalphabet@@#1{\DeclareMathAlphabet#1{U}{}{}{}}%
\gdef\newmathalphabet@@@#1#2#3#4{%
       \DeclareMathAlphabet{#1}{U}{#2}{#3}{#4}}%
\@onlypreamble\newmathalphabet
\@onlypreamble\newmathalphabet@@
\@onlypreamble\newmathalphabet@@@
```

\if@no@font@opt
\@no@font@optfalse

```
\global\let\if@no@font@opt\iftrue
\gdef\@no@font@optfalse{\let\if@no@font@opt\iffalse}%
```

\define@mathalphabet This is a case where dying is best.

```
\gdef\define@mathalphabet{%
      \err@rel@i\define@mathalphabet\DeclareMathAlphabet
}%
\@onlypreamble\define@mathalphabet
```

\define@mathgroup And here is another one

```
\gdef\define@mathgroup{%
      \err@rel@i\define@mathgroup\DeclareSymbolFont
}%
\@onlypreamble\define@mathgroup
```

\addtoversion `\addtoversion` is the old form.

```
\def\addtoversion#1#2{%
  \warn@rel@i\addtoversion\SetMathAlphabet
  \SetMathAlphabet#2{#1}{U}}%
\@onlypreamble\addtoversion
```

Finishing off this huge `\IncludeInRelease` argument:

```
\EndIncludeInRelease
⟨/latexrelease⟩
```

File s
ltfssdcl.dtx

This file contains the main implementation of the font selection scheme commands. See other parts of the LaTeX distribution, or *The LaTeX 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.

37 Interface Commands

`\in@` `\ifin@` `\@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).

```
⟨*2ekernel⟩
\def\in@#1#2%
 {%
   \begingroup
     \def\in@@##1#1{}%
     \toks@\expandafter{\in@@#2{}{}#1}%
     \edef\in@@{\the\toks@}%
   \expandafter\endgroup
   \ifx\in@@\@empty
     \in@false
   \else
     \in@true
   \fi
 }
\newif\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`⟨*version*$_1$⟩`\version@elt`⟨*version*$_2$⟩...
  `\version@elt`⟨*version*$_n$⟩

- 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 ⟨*font info*⟩ 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.

```
\let\version@list=\@empty
\@onlypreamble\version@list
```

`\version@elt`

```
\let\version@elt\relax
\@onlypreamble\version@elt
```

`\new@mathversion` The macro `\new@mathversion` is called with the version control sequence as its argument.

```
%\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.

```
%  \expandafter\in@\expandafter#1\expandafter{\version@list}%
%  \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.

```
%    \@latex@error{Math version
%                 `\expandafter\@gobblefour\string#1'
%                 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`.

```
%  \else
%     \global\expandafter\newcount\csname c@\expandafter
%                                 \@gobble\string#1\endcsname
%     \global\csname c@\expandafter
%                                 \@gobble\string#1\endcsname\@ne
%     \def\version@elt{\noexpand\version@elt\noexpand}%
%     \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.

```
%     \def\reserved@c{\noexpand\reserved@c\noexpand}%
%     \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).

```
%     \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.

```
%         \edef##1{\expandafter\remove@nil##1%
%                 \reserved@c
%                 #1%
%                 \reserved@e
%                 \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.

```
%     \alpha@list
%  \fi}
```

`\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`.

```
\let\alpha@list\@empty
\@onlypreamble\alpha@list
```

`\alpha@elt`

```
\let\alpha@elt\relax
\@onlypreamble\alpha@elt
```

`\newgroup` Start the group (fam) allocation at 0. (Doesn't belong here.)

```
\count18=-1
```

`\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.

```
⟨/2ekernel⟩
⟨latexrelease⟩\IncludeInRelease{2015/01/01}
⟨latexrelease⟩                 {\select@group}{\select@group}%
⟨*2ekernel | latexrelease⟩
\def\select@group#1#2#3#4{%
 \ifx\math@bgroup\bgroup\else\relax\expandafter\@firstofone\fi
 {%
 \ifmmode
  \ifnum\csname c@mv@\math@version\endcsname<\e@mathgroup@top
     \begingroup
       \escapechar\m@ne
       \getanddefine@fonts{\csname c@mv@\math@version\endcsname}#3%
       \globaldefs\@ne  \math@fonts
     \endgroup
     \init@restore@version
     \xdef#1{\noexpand\use@mathgroup\noexpand#2%
             {\number\csname c@mv@\math@version\endcsname}}%
     \global\advance\csname c@mv@\math@version\endcsname\@ne
   \else
     \let#1\relax
     \@latex@error{Too many math alphabets used in
                   version \math@version}%
        \@eha
   \fi
 \else \expandafter\non@alpherr\fi
 #1{#4}%
 }%
}
⟨/2ekernel | latexrelease⟩
⟨latexrelease⟩\EndIncludeInRelease
⟨latexrelease⟩\IncludeInRelease{0000/00/00}
⟨latexrelease⟩                 {\select@group}{\select@group}%
⟨latexrelease⟩\def\select@group#1#2#3#4{%
⟨latexrelease⟩ \ifx\math@bgroup\bgroup\else\relax\expandafter\@firstofone\fi
⟨latexrelease⟩ {%
⟨latexrelease⟩ \ifmmode
⟨latexrelease⟩  \ifnum\csname c@mv@\math@version\endcsname<\sixt@@n
⟨latexrelease⟩     \begingroup
⟨latexrelease⟩       \escapechar\m@ne
⟨latexrelease⟩       \getanddefine@fonts
⟨latexrelease⟩         {\csname c@mv@\math@version\endcsname}#3%
⟨latexrelease⟩       \globaldefs\@ne  \math@fonts
⟨latexrelease⟩     \endgroup
⟨latexrelease⟩     \init@restore@version
⟨latexrelease⟩     \xdef#1{\noexpand\use@mathgroup\noexpand#2%
⟨latexrelease⟩             {\number\csname c@mv@\math@version\endcsname}}%
⟨latexrelease⟩     \global\advance\csname c@mv@\math@version\endcsname\@ne
⟨latexrelease⟩   \else
⟨latexrelease⟩     \let#1\relax
⟨latexrelease⟩     \@latex@error{Too many math alphabets used in
⟨latexrelease⟩                   version \math@version}%
⟨latexrelease⟩        \@eha
⟨latexrelease⟩   \fi
⟨latexrelease⟩ \else \expandafter\non@alpherr\fi
⟨latexrelease⟩  #1{#4}%
⟨latexrelease⟩  }%
⟨latexrelease⟩}
⟨latexrelease⟩\EndIncludeInRelease
⟨*2ekernel⟩
```

```
\@onlypreamble\restore@mathversion
```

\init@restore@version

```
\def\init@restore@version{%
        \global\let\init@restore@version\relax
        \xdef\restore@mathversion
             {\expandafter\noexpand\csname mv@\math@version\endcsname
              \global\csname c@mv@\math@version\endcsname
              \number\csname c@mv@\math@version\endcsname\relax}%
        \aftergroup\dorestore@version
}
\@onlypreamble\init@restore@version
```

\non@alpherr

```
\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.

```
    \string#1allowed only in math mode}\@ehd}
```

\dorestore@version

```
\def\dorestore@version
 {\ifmmode
    \aftergroup\dorestore@version
  \else
    \gdef\init@restore@version{%
        \global\let\init@restore@version\relax
        \xdef\restore@mathversion
             {\expandafter\noexpand\csname mv@\math@version\endcsname
              \global\csname c@mv@\math@version\endcsname
              \number\csname c@mv@\math@version\endcsname\relax}%
        \aftergroup\dorestore@version
    }%
    \begingroup
      \let\getanddefine@fonts\@gobbletwo
      \restore@mathversion
    \endgroup
  \fi}%
\@onlypreamble\dorestore@version
```

\document@select@group We surround `\select@group` with braces so that functions using it can be used directly after _ or ^.

```
⟨/2ekernel⟩
⟨latexrelease⟩\IncludeInRelease{2015/01/01}
⟨latexrelease⟩    {\document@select@group}{\document@select@group}%
⟨*2ekernel | latexrelease⟩
\def\document@select@group#1#2#3#4{%
 \ifx\math@bgroup\bgroup\else\relax\expandafter\@firstofone\fi
 {%
 \ifmmode
   \ifnum\csname c@mv@\math@version\endcsname<\e@mathgroup@top
     \begingroup
       \escapechar\m@ne
       \getanddefine@fonts{\csname c@mv@\math@version\endcsname}#3%
       \globaldefs\@ne  \math@fonts
     \endgroup
     \expandafter\extract@alph@from@version
         \csname mv@\math@version\expandafter\endcsname
         \expandafter{\number\csname
                       c@mv@\math@version\endcsname}%
          #1%
     \global\advance\csname c@mv@\math@version\endcsname\@ne
   \else
     \let#1\relax
     \@latex@error{Too many math alphabets used
                   in version \math@version}%
       \@eha
  \fi
 \else \expandafter\non@alpherr\fi
 #1{#4}%
 }%
}
⟨/2ekernel | latexrelease⟩
⟨latexrelease⟩\EndIncludeInRelease
⟨latexrelease⟩\IncludeInRelease{0000/00/00}
⟨latexrelease⟩  {\document@select@group}{\document@select@group}%
⟨latexrelease⟩\def\document@select@group#1#2#3#4{%
⟨latexrelease⟩ \ifx\math@bgroup\bgroup\else\relax\expandafter\@firstofone\fi
⟨latexrelease⟩ {%
⟨latexrelease⟩ \ifmmode
⟨latexrelease⟩   \ifnum\csname c@mv@\math@version\endcsname<\sixt@@n
⟨latexrelease⟩     \begingroup
⟨latexrelease⟩       \escapechar\m@ne
⟨latexrelease⟩       \getanddefine@fonts
⟨latexrelease⟩         {\csname c@mv@\math@version\endcsname}#3%
⟨latexrelease⟩       \globaldefs\@ne  \math@fonts
⟨latexrelease⟩     \endgroup
⟨latexrelease⟩     \expandafter\extract@alph@from@version
⟨latexrelease⟩         \csname mv@\math@version\expandafter\endcsname
⟨latexrelease⟩         \expandafter{\number\csname
⟨latexrelease⟩                       c@mv@\math@version\endcsname}%
⟨latexrelease⟩          #1%
⟨latexrelease⟩     \global\advance\csname c@mv@\math@version\endcsname\@ne
⟨latexrelease⟩   \else
⟨latexrelease⟩     \let#1\relax
⟨latexrelease⟩     \@latex@error{Too many math alphabets used
⟨latexrelease⟩                   in version \math@version}%
⟨latexrelease⟩       \@eha
⟨latexrelease⟩  \fi
⟨latexrelease⟩ \else \expandafter\non@alpherr\fi
⟨latexrelease⟩ #1{#4}%
⟨latexrelease⟩ }%
```

```
⟨latexrelease⟩}
⟨latexrelease⟩\EndIncludeInRelease
⟨*2ekernel⟩
```

\process@table

```
\def\process@table{%
    \def\cdp@elt##1##2##3##4{%
        \@font@info{Checking defaults for
                  ##1/##2/##3/##4}%
        \expandafter
        \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.

```
          \begingroup
           \def\f@encoding{##1}\def\f@family{##2}%
           \try@load@fontshape
          \endgroup
        \fi
        \expandafter
        \ifx\csname##1/##2/##3/##4\endcsname\relax
             \@latex@error{This NFSS system isn't set up properly}%
                       {For encoding scheme ##1 the defaults
                        ##2/##3/##4 do not form a valid font shape}%
        \else
             \@font@info{... okay}%
        \fi}%
    \cdp@list
```

Now we make sure that \error@fontshape is okay.

```
    \begingroup
       \escapechar\m@ne
       \error@fontshape
       \expandafter\ifx\csname \curr@fontshape\endcsname\relax
          \begingroup
            \try@load@fontshape
           \endgroup
       \fi
       \expandafter\ifx\csname \curr@fontshape\endcsname\relax
         \@latex@error{This NFSS system isn't set up properly}%
            {The system maintainer forgot to specify a suitable
             substitution
             font shape using the \noexpand\DeclareErrorFont
             command}%
       \fi
    \endgroup
```

Set \select@group to its meaning used within the document body.

```
    \let\select@group\document@select@group
```

Install the default font attributes they are currently pointing to error font shape. Don't use \reset@font since that would trigger \selectfont.

```
    \fontencoding{\encodingdefault}%
    \fontfamily{\familydefault}%
    \fontseries{\seriesdefault}%
    \fontshape{\shapedefault}%
```

kill all macros not longer needed. we need to add many more!!!!!!

```
 \everyjob{}%
}
\@onlypreamble\process@table
```

```
%\@onlypreamble\set@mathradical
```

\DeclareMathVersion

```
\def\DeclareMathVersion#1{%
  \expandafter\new@mathversion\csname mv@#1\endcsname}
\@onlypreamble\DeclareMathVersion
```

\new@mathversion

```
\def\new@mathversion#1{%
  \expandafter\in@\expandafter#1\expandafter{\version@list}%
  \ifin@
    \@font@info{Redeclaring math version
               ‘\expandafter\@gobblefour\string#1’}%
  \else
    \expandafter\newcount\csname c@\expandafter
                                \@gobble\string#1\endcsname
    \def\version@elt{\noexpand\version@elt\noexpand}%
    \edef\version@list{\version@list\version@elt#1}%
  \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.

```
  \toks@{}%
  \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.

```
  \def\group@elt##1##2{%
       \advance\count@\@ne
       \addto@hook\toks@{\getanddefine@fonts##1##2}%
       }%
  \group@list
```

We set the counter for this math version to the number of math groups found in \group@list.

```
  \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.

```
  \def\alpha@elt##1##2##3{%
       \ifx##2\no@alphabet@error
         \toks@\expandafter{\the\toks@\install@mathalphabet##1%
             {\no@alphabet@error##1}}%
       \else
         \toks@\expandafter{\the\toks@\install@mathalphabet##1%
             {\select@group##1##2##3}}%
       \fi
          }%
  \alpha@list
```

Finally we define the math version to expand to the contents of `\toks@`.

```
  \xdef#1{\the\toks@}%
}
\@onlypreamble\new@mathversion
```

`\DeclareSymbolFont`

```
\def\DeclareSymbolFont#1#2#3#4#5{%
 \@tempswafalse
 \edef\reserved@b{#2}%
 \def\cdp@elt##1##2##3##4{\def\reserved@c{##1}%
      \ifx\reserved@b\reserved@c \@tempswatrue\fi}%
 \cdp@list
 \if@tempswa
   \@ifundefined{sym#1}{%
      \ifnum\count18<15 %
        \expandafter\new@mathgroup\csname sym#1\endcsname
        \expandafter\new@symbolfont\csname sym#1\endcsname
                        {#2}{#3}{#4}{#5}%
      \else
         \@latex@error{Too many symbol fonts declared}\@eha
      \fi
     }%
     {%
      \@font@info{Redeclaring symbol font `#1'}%
```

Update the group list.

```
      \def\group@elt##1##2{%
           \noexpand\group@elt\noexpand##1%
           \expandafter\ifx\csname sym#1\endcsname##1%
             \expandafter\noexpand\csname#2/#3/#4/#5\endcsname
           \else
               \noexpand##2%
           \fi}%
      \xdef\group@list{\group@list}%
```

Update the version list.

```
      \def\version@elt##1{%
          \expandafter
          \SetSymbolFont@\expandafter##1\csname#2/#3/#4/#5\expandafter
              \endcsname \csname sym#1\endcsname
          }%
      \version@list
     }%
  \else
    \@latex@error{Encoding scheme  `#2' unknown}\@eha
  \fi
  }
\@onlypreamble\DeclareSymbolFont
```

`\group@list`

```
\let\group@list\@empty
\@onlypreamble\group@list
```

\group@elt

```
\let\group@elt\relax
\@onlypreamble\group@elt
```

\new@symbolfont

```
\def\new@symbolfont#1#2#3#4#5{%
    \toks@\expandafter{\group@list}%
    \edef\group@list{\the\toks@\noexpand\group@elt\noexpand#1%
                     \expandafter\noexpand\csname#2/#3/#4/#5\endcsname}%
    \def\version@elt##1{\toks@\expandafter{##1}%
                   \edef##1{\the\toks@\noexpand\getanddefine@fonts
                   #1\expandafter\noexpand\csname#2/#3/#4/#5\endcsname}%
                  \global\advance\csname c@\expandafter
                                 \@gobble\string##1\endcsname\@ne
                 }%
    \version@list
}
\@onlypreamble\new@symbolfont
```

\SetSymbolFont

```
\def\SetSymbolFont#1#2#3#4#5#6{%
 \@tempswafalse
 \edef\reserved@b{#3}%
 \def\cdp@elt##1##2##3##4{\def\reserved@c{##1}%
      \ifx\reserved@b\reserved@c \@tempswatrue\fi}%
 \cdp@list
 \if@tempswa
  \expandafter\SetSymbolFont@
    \csname mv@#2\expandafter\endcsname\csname#3/#4/#5/#6\expandafter
    \endcsname \csname sym#1\endcsname
 \else
  \@latex@error{Encoding scheme  ‘#3’ unknown}\@eha
 \fi
}
\@onlypreamble\SetSymbolFont
```

\SetSymbolFont@

```
\def\SetSymbolFont@#1#2#3{%
  \expandafter\in@\expandafter#1\expandafter{\version@list}%
  \ifin@
    \expandafter\in@\expandafter#3\expandafter{\group@list}%
    \ifin@
      \begingroup
        \expandafter\get@cdp\string#2\@nil\reserved@a
        \toks@{}%
        \def\install@mathalphabet##1##2{%
             \addto@hook\toks@{\install@mathalphabet##1{##2}}%
            }%
        \def\getanddefine@fonts##1##2{%
          \ifnum##1=#3%
             \addto@hook\toks@{\getanddefine@fonts#3#2}%
             \expandafter\get@cdp\string##2\@nil\reserved@b
             \ifx\reserved@a\reserved@b\else
                \@font@info{Encoding `\reserved@b' has changed
                    to `\reserved@a' for symbol font\MessageBreak
                   `\expandafter\@gobblefour\string#3' in the
                    math version `\expandafter
                    \@gobblefour\string#1'}%
             \fi
             \@font@info{%
                Overwriting symbol font
                `\expandafter\@gobblefour\string#3' in
                 version `\expandafter
                \@gobblefour\string#1'\MessageBreak
                \@spaces \expandafter\@gobble\string##2 -->
                         \expandafter\@gobble\string#2}%
          \else
             \addto@hook\toks@{\getanddefine@fonts##1##2}%
          \fi}%
         #1%
         \xdef#1{\the\toks@}%
      \endgroup
    \else
       \@latex@error{Symbol font `\expandafter\@gobblefour\string#3'
                  not defined}\@eha
    \fi
  \else
    \@latex@error{Math version `\expandafter\@gobblefour\string#1'
       is not
       defined}{You probably misspelled the name of the math
       version.^^JOr you have to specify an additional package.}%
  \fi
}
\@onlypreamble\SetSymbolFont@
```

\get@cdp

```
\def\get@cdp#1#2/#3\@nil#4{\def#4{#2}}
\@onlypreamble\get@cdp
```

\DeclareMathAlphabet

```
\def\DeclareMathAlphabet#1#2#3#4#5{%
 \@tempswafalse
 \edef\reserved@b{#2}%
 \def\cdp@elt##1##2##3##4{\def\reserved@c{##1}%
      \ifx\reserved@b\reserved@c \@tempswatrue\fi}%
 \cdp@list
 \if@tempswa
   \expandafter\ifx
   \csname\expandafter\@gobble\string#1\endcsname
   \relax
      \new@mathalphabet#1{#2}{#3}{#4}{#5}%
   \else
```

Check if it is already a math alphabet.

```
     \edef\reserved@a{\noexpand\in@{\string\select@group}%
```

```
         {\expandafter\meaning\csname \expandafter
          \@gobble\string#1\space\endcsname}}%
     \reserved@a
     \ifin@
       \@font@info{Redeclaring math alphabet \string#1}%
       \def\version@elt##1{%
         \expandafter\SetMathAlphabet@\expandafter
            ##1\csname#2/#3/#4/#5\expandafter\endcsname

            \csname M@#2\expandafter\endcsname
            \csname \expandafter\@gobble\string#1\space\endcsname#1}%
       \version@list
     \else
```

Check if it is a math alphabet defined via `\DeclareSymbolFontAlphabet`.

```
       \edef\reserved@a{\noexpand\in@{\string\use@mathgroup}%
         {\expandafter\meaning\csname \expandafter
          \@gobble\string#1\space\endcsname}}%
       \reserved@a
       \ifin@
```

In that case overwriting is simple since there is nothing inserted in the math version macros.

```
         \@font@info{Redeclaring math alphabet \string#1}%
         \new@mathalphabet#1{#2}{#3}{#4}{#5}%
```

Otherwise panic.

```
       \else
         \@latex@error{Command `\string#1' already defined}\@eha
       \fi
     \fi
  \fi
 \else
  \@latex@error{Encoding scheme  `#2' unknown}\@eha
 \fi
  }
\@onlypreamble\DeclareMathAlphabet
```

`\new@mathalphabet`

```
\def\new@mathalphabet#1#2#3#4#5{%
    \toks@\expandafter{\alpha@list}%
    \edef#1{\expandafter\noexpand\csname \expandafter
            \@gobble\string#1\space\endcsname
            \if/#5/%
               \noexpand\no@alphabet@error
               \noexpand\no@alphabet@error
            \else
               \expandafter\noexpand\csname M@#2\endcsname
               \expandafter\noexpand\csname#2/#3/#4/#5\endcsname
            \fi
           }%
    \toks2\expandafter{#1}%
    \edef\alpha@list{\the\toks@\noexpand\alpha@elt\the\toks2}%
    \def\version@elt##1{\toks@\expandafter{##1}%
                   \edef##1{\the\toks@\install@mathalphabet
```

```
                            \expandafter\noexpand
                            \csname \expandafter\@gobble
                               \string#1\space\endcsname
                           {\if/#5/%
                             \noexpand\no@alphabet@error
                             \noexpand#1%
                            \else
                             \noexpand\select@group\the\toks2
                            \fi}}%
                 }%
    \version@list
    \expandafter\edef\csname \expandafter\@gobble
                \string#1\space\endcsname{\if/#5/%
              \noexpand\no@alphabet@error
              \noexpand#1%
            \else
              \noexpand\select@group\the\toks2
            \fi}%
    \edef#1{\noexpand\protect
            \expandafter\noexpand\csname \expandafter
            \@gobble\string#1\space\endcsname}%
}
\@onlypreamble\new@mathalphabet
```

\SetMathAlphabet

```
\def\SetMathAlphabet#1#2#3#4#5#6{%
 \@tempswafalse
 \edef\reserved@b{#3}%
 \def\cdp@elt##1##2##3##4{\def\reserved@c{##1}%
      \ifx\reserved@b\reserved@c \@tempswatrue\fi}%
 \cdp@list
 \if@tempswa
  \expandafter\SetMathAlphabet@
    \csname mv@#2\expandafter\endcsname\csname#3/#4/#5/#6\expandafter
    \endcsname \csname M@#3\expandafter\endcsname
    \csname \expandafter\@gobble\string#1\space\endcsname#1%
 \else
  \@latex@error{Encoding scheme  `#3' unknown}\@eha
 \fi
}
\@onlypreamble\SetMathAlphabet
```

\SetMathAlphabet@

```
\def\SetMathAlphabet@#1#2#3#4#5{%
  \expandafter\in@\expandafter#1\expandafter{\version@list}%
  \ifin@
    \expandafter\in@\expandafter#4\expandafter{\alpha@list}%
    \ifin@
      \begingroup
        \toks@{}%
        \def\getanddefine@fonts##1##2{%
             \addto@hook\toks@{\getanddefine@fonts##1##2}%
            }%
        \def\reserved@c##1##2##3##4{%                % for message below
```

```
            \expandafter\@gobble\string##4}%
        \def\install@mathalphabet##1##2{%
          \ifx##1#4%
             \addto@hook\toks@
                {\install@mathalphabet#4{\select@group#4#3#2}}%
             \@font@info{Overwriting math alphabet
                `\string#5' in version `\expandafter
                 \@gobblefour\string#1'\MessageBreak
                 \@spaces \reserved@c##2 -->
                        \expandafter\@gobble\string#2}%
          \else
             \addto@hook\toks@{\install@mathalphabet##1{##2}}%
          \fi
          }%
        #1%
        \xdef#1{\the\toks@}%
      \endgroup
    \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.

```
       \edef\reserved@a{%
         \noexpand\in@{\string\use@mathgroup}{\meaning#4}}%
       \reserved@a
       \ifin@
         \def\reserved@b##1\use@mathgroup##2##3{%
             \def\reserved@b{##3}\def\reserved@c{##2}}%
         \expandafter\reserved@b#4%
         \begingroup
           \def\install@mathalphabet##1##2{%
               \addto@hook\toks@{\install@mathalphabet##1{##2}}%
               }%
            \def\getanddefine@fonts##1##2{%
              \addto@hook\toks@{\getanddefine@fonts##1##2}%
              \ifnum##1=\reserved@b
                 \expandafter
                 \addto@hook\expandafter\toks@
                 \expandafter{\expandafter\install@mathalphabet
                 \expandafter#4\expandafter
                       {\expandafter\select@group\expandafter
                         #4\reserved@c##2}}%
              \fi
                      }%
           \def\version@elt##1{%
               \toks@{}%
               ##1%
               \xdef##1{\the\toks@}%
              }%
           \version@list
         \endgroup
```

Put it into the `\alpha@list` with default 'error'

```
         \expandafter\gdef\expandafter\alpha@list\expandafter
             {\alpha@list
```

```
              \alpha@elt #4\no@alphabet@error \no@alphabet@error}%
         \gdef#4{\no@alphabet@error #5}% fake things :-)
```

Then call the internal setting routine again:

```
         \SetMathAlphabet@{#1}{#2}{#3}#4#5%
       \else
         \@latex@error{Command `\string#5' not defined as a
                       math alphabet}%
            {Use \noexpand\DeclareMathAlphabet to define it.}%
       \fi
    \fi
  \else
    \@latex@error{Math version `\expandafter\@gobblefour\string#1'
       is not
       defined}{You probably misspelled the name of the math
       version.^^JOr you have to specify an additional package.}%
  \fi
}
\@onlypreamble\SetMathAlphabet@
```

\DeclareMathAccent

Could do with more checks like allowing single number in #4 lowercase in #4 etc

```
⟨/2ekernel⟩
⟨*2ekernel | latexrelease⟩
⟨latexrelease⟩\IncludeInRelease{2019/10/01}%
⟨latexrelease⟩                 {DeclareMathAccent}{Make math accents robust}%
\def\DeclareMathAccent#1#2#3#4{%
  \expandafter\in@\csname sym#3\expandafter\endcsname
    \expandafter{\group@list}%
  \ifin@
    \begingroup
      \count\z@=#4\relax
      \count\tw@\count\z@
      \divide\count\z@\sixt@@n
      \count@\count\z@
      \multiply\count@\sixt@@n
      \advance\count\tw@-\count@
      \if\relax\noexpand#1% is command?
        \edef\reserved@a{\noexpand\in@
          {\expandafter\@gobble\string\mathaccent}
          {\expandafter\meaning
           \csname\expandafter\@gobble\string#1\space\endcsname}}%
        \reserved@a
        \ifin@
          \expandafter\let
            \csname\expandafter\@gobble\string#1\space\endcsname
            \@undefined
          \expandafter\set@mathaccent
             \csname sym#3\endcsname#1#2%
             {\hexnumber@{\count\z@}\hexnumber@{\count\tw@}}%
          \@font@info{Redeclaring math accent \string#1}%
        \else
          \expandafter\ifx
          \csname\expandafter\@gobble\string#1\endcsname
          \relax
```

```
            \expandafter\set@mathaccent
               \csname sym#3\endcsname#1#2%
               {\hexnumber@{\count\z@}\hexnumber@{\count\tw@}}%
          \else
            \@latex@error{Command `\string#1' already defined}\@eha
          \fi
        \fi
      \else
       \@latex@error{Not a command name: `\noexpand#1'}\@eha
      \fi
    \endgroup
  \else
    \@latex@error{Symbol font `#3' is not defined}\@eha
  \fi
}
⟨/2ekernel | latexrelease⟩
⟨latexrelease⟩\EndIncludeInRelease
⟨latexrelease⟩\IncludeInRelease{0000/00/00}%
⟨latexrelease⟩                {DeclareMathAccent}{Make math accents robust}%
⟨latexrelease⟩\def\DeclareMathAccent#1#2#3#4{%
⟨latexrelease⟩  \expandafter\in@\csname sym#3\expandafter\endcsname
⟨latexrelease⟩     \expandafter{\group@list}%
⟨latexrelease⟩  \ifin@
⟨latexrelease⟩    \begingroup
⟨latexrelease⟩      \count\z@=#4\relax
⟨latexrelease⟩      \count\tw@\count\z@
⟨latexrelease⟩      \divide\count\z@\sixt@@n
⟨latexrelease⟩      \count@\count\z@
⟨latexrelease⟩      \multiply\count@\sixt@@n
⟨latexrelease⟩      \advance\count\tw@-\count@
⟨latexrelease⟩      \if\relax\noexpand#1% is command?
⟨latexrelease⟩        \edef\reserved@a{\noexpand\in@
⟨latexrelease⟩           {\expandafter\@gobble\string\mathaccent}{\meaning#1}}%
⟨latexrelease⟩        \reserved@a
⟨latexrelease⟩        \ifin@
⟨latexrelease⟩          \expandafter\set@mathaccent
⟨latexrelease⟩             \csname sym#3\endcsname#1#2%
⟨latexrelease⟩              {\hexnumber@{\count\z@}\hexnumber@{\count\tw@}}%
⟨latexrelease⟩          \@font@info{Redeclaring math accent \string#1}%
⟨latexrelease⟩        \else
⟨latexrelease⟩          \expandafter\ifx
⟨latexrelease⟩          \csname\expandafter\@gobble\string#1\endcsname
⟨latexrelease⟩          \relax
⟨latexrelease⟩            \expandafter\set@mathaccent
⟨latexrelease⟩               \csname sym#3\endcsname#1#2%
⟨latexrelease⟩               {\hexnumber@{\count\z@}\hexnumber@{\count\tw@}}%
⟨latexrelease⟩          \else
⟨latexrelease⟩            \@latex@error{Command `\string#1' already defined}\@eha
⟨latexrelease⟩          \fi
⟨latexrelease⟩        \fi
⟨latexrelease⟩      \else
⟨latexrelease⟩       \@latex@error{Not a command name: `\noexpand#1'}\@eha
⟨latexrelease⟩      \fi
⟨latexrelease⟩    \endgroup
```

```
⟨latexrelease⟩  \else
⟨latexrelease⟩    \@latex@error{Symbol font ‘#3’ is not defined}\@eha
⟨latexrelease⟩  \fi
⟨latexrelease⟩}
⟨latexrelease⟩\EndIncludeInRelease
⟨*2ekernel⟩
```

```
\@onlypreamble\DeclareMathAccent
```

\set@mathaccent

```
⟨/2ekernel⟩
⟨*2ekernel | latexrelease⟩
⟨latexrelease⟩\IncludeInRelease{2019/10/01}%
⟨latexrelease⟩                  {\set@mathaccent}{makemath accents robust}%
\def\set@mathaccent#1#2#3#4{%
  \xdef#2{\mathaccent"\mathchar@type#3\hexnumber@#1#4\relax}%
  \MakeRobust#2%
}
\@onlypreamble\set@mathaccent
⟨/2ekernel | latexrelease⟩
⟨latexrelease⟩\EndIncludeInRelease
⟨latexrelease⟩\IncludeInRelease{0000/00/00}%
⟨latexrelease⟩                  {\set@mathaccent}{makemath accents robust}%
⟨latexrelease⟩
⟨latexrelease⟩\def\set@mathaccent#1#2#3#4{%
⟨latexrelease⟩  \xdef#2{\mathaccent"\mathchar@type#3\hexnumber@#1#4\relax}}
⟨latexrelease⟩
⟨latexrelease⟩\EndIncludeInRelease
⟨*2ekernel⟩
```

\DeclareMathSymbol

```
\def\DeclareMathSymbol#1#2#3#4{%
  \expandafter\in@\csname sym#3\expandafter\endcsname
     \expandafter{\group@list}%
  \ifin@
    \begingroup
      \count\z@=#4\relax
      \count\tw@\count\z@
      \divide\count\z@\sixt@@n
      \count@\count\z@
      \multiply\count@\sixt@@n
      \advance\count\tw@-\count@
      \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.

```
        \edef\reserved@b{\expandafter\noexpand
                           \csname\expandafter\@gobble\string#1\space\endcsname}%
```

Test both `#1` and `#1␣` for containing `mathchar`.

```
        \edef\reserved@a
          {\noexpand\in@{\expandafter\@gobble\string\mathchar}%
                        {\meaning#1\expandafter\meaning\reserved@b}}%
        \reserved@a
```

Drop `#1␣` in case it was defined before.

```
        \global\expandafter\let\reserved@b\@undefined
        \ifin@
          \expandafter\set@mathsymbol
             \csname sym#3\endcsname#1#2%
             {\hexnumber@{\count\z@}\hexnumber@{\count\tw@}}%
          \@font@info{Redeclaring math symbol \string#1}%
        \else
          \expandafter\ifx
            \csname\expandafter\@gobble\string#1\endcsname
            \relax
            \expandafter\set@mathsymbol
               \csname sym#3\endcsname#1#2%
               {\hexnumber@{\count\z@}\hexnumber@{\count\tw@}}%
          \else
            \@latex@error{Command `\string#1' already defined}\@eha
          \fi
        \fi
      \else
        \expandafter\set@mathchar
          \csname sym#3\endcsname#1#2
          {\hexnumber@{\count\z@}\hexnumber@{\count\tw@}}%
      \fi
    \endgroup
  \else
    \@latex@error{Symbol font `#3' is not defined}\@eha
  \fi
}
\@onlypreamble\DeclareMathSymbol
```

\set@mathchar

```
\def\set@mathchar#1#2#3#4{%
  \global\mathcode`#2="\mathchar@type#3\hexnumber@#1#4\relax}
\@onlypreamble\set@mathchar
```

\set@mathsymbol

```
\def\set@mathsymbol#1#2#3#4{%
  \global\mathchardef#2"\mathchar@type#3\hexnumber@#1#4\relax}
\@onlypreamble\set@mathsymbol
```

```
%\def\mathsymbol#1#2#3{%
% \@tempcnta=#3\relax
% \@tempcntb\@tempcnta
% \divide\@tempcnta\sixt@@n
% \count@\@tempcnta
% \multiply\count@\sixt@@n
% \advance\@tempcntb-\count@
% \mathchar"\mathchar@type#1\hexnumber@#2%
%             \hexnumber@\@tempcnta\hexnumber@\@tempcntb\relax}
%
%\def\DeclareMathAlphabetCharacter#1#2#3{%
% \DeclareMathSymbol{#1}7{#2}{#3}}
```

\DeclareMathDelimiter

```
\def\DeclareMathDelimiter#1{%
  \if\relax\noexpand#1%
    \expandafter\@DeclareMathDelimiter
  \else
    \expandafter\@xxDeclareMathDelimiter
  \fi
  #1}
\@onlypreamble\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.

```
\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!

```
   \begingroup
    \let\mathalpha\mathord
    \ifnum7=\mathchar@type{#2}%
      \endgroup
```

If this branch is taken we have old syntax (5 arguments).

```
      \expandafter\@firstofone
    \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.

```
      \endgroup
      \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.

```
      \expandafter\@firstoftwo
    \fi
    {\@xDeclareMathDelimiter#1}{#2}{#3}{#4}}
\@onlypreamble\@xxDeclareMathDelimiter
```

\@DeclareMathDelimiter

```
\def\@DeclareMathDelimiter#1#2#3#4#5#6{%
  \expandafter\in@\csname sym#3\expandafter\endcsname
     \expandafter{\group@list}%
  \ifin@
    \expandafter\in@\csname sym#5\expandafter\endcsname
       \expandafter{\group@list}%
    \ifin@
      \begingroup
        \count\z@=#4\relax
        \count\tw@\count\z@
        \divide\count\z@\sixt@@n
        \count@\count\z@
        \multiply\count@\sixt@@n
        \advance\count\tw@-\count@
        \edef\reserved@c{\hexnumber@{\count\z@}\hexnumber@{\count\tw@}}%
      %
        \count\z@=#6\relax
        \count\tw@\count\z@
        \divide\count\z@\sixt@@n
        \count@\count\z@
        \multiply\count@\sixt@@n
        \advance\count\tw@-\count@
        \edef\reserved@d{\hexnumber@{\count\z@}\hexnumber@{\count\tw@}}%
      %
        \edef\reserved@a{\noexpand\in@
            {\expandafter\@gobble\string\delimiter}{\meaning#1}}%
        \reserved@a
        \ifin@
          \expandafter\set@mathdelimiter
             \csname sym#3\expandafter\endcsname
             \csname sym#5\endcsname#1#2%
             \reserved@c\reserved@d
          \@font@info{Redeclaring math delimiter \string#1}%
        \else
            \expandafter\ifx
            \csname\expandafter\@gobble\string#1\endcsname
            \relax
            \expandafter\set@mathdelimiter
              \csname sym#3\expandafter\endcsname
              \csname sym#5\endcsname#1#2%
              \reserved@c\reserved@d
          \else
            \@latex@error{Command `\string#1' already defined}\@eha
          \fi
        \fi
      \endgroup
    \else
      \@latex@error{Symbol font `#5' is not defined}\@eha
    \fi
  \else
    \@latex@error{Symbol font `#3' is not defined}\@eha
  \fi
}
```

```
\@onlypreamble\@DeclareMathDelimiter
```

\@xDeclareMathDelimiter

```
\def\@xDeclareMathDelimiter#1#2#3#4#5{%
  \expandafter\in@\csname sym#2\expandafter\endcsname
     \expandafter{\group@list}%
  \ifin@
    \expandafter\in@\csname sym#4\expandafter\endcsname
       \expandafter{\group@list}%
    \ifin@
      \begingroup
        \count\z@=#3\relax
        \count\tw@\count\z@
        \divide\count\z@\sixt@@n
        \count@\count\z@
        \multiply\count@\sixt@@n
        \advance\count\tw@-\count@
        \edef\reserved@c{\hexnumber@{\count\z@}\hexnumber@{\count\tw@}}%
      %
        \count\z@=#5\relax
        \count\tw@\count\z@
        \divide\count\z@\sixt@@n
        \count@\count\z@
        \multiply\count@\sixt@@n
        \advance\count\tw@-\count@
        \edef\reserved@d{\hexnumber@{\count\z@}\hexnumber@{\count\tw@}}%
        \expandafter\set@@mathdelimiter
           \csname sym#2\expandafter\endcsname\csname sym#4\endcsname#1%
           \reserved@c\reserved@d
      \endgroup
    \else
      \@latex@error{Symbol font `#4' is not defined}\@eha
    \fi
  \else
    \@latex@error{Symbol font `#2' is not defined}\@eha
  \fi
}
\@onlypreamble\@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)

```
⟨/2ekernel⟩
⟨*2ekernel | latexrelease⟩
⟨latexrelease⟩\IncludeInRelease{2019/10/01}%
⟨latexrelease⟩                  {\set@mathdelimiter}{make delimiters robust}%
\def\set@mathdelimiter#1#2#3#4#5#6{%
```

We use `\protected` not `\MakeRobust` so that `\bigl\lfoor` etc. works inside `\protected@edef`.

```
  \protected
  \xdef#3{\delimiter"\mathchar@type#4\hexnumber@#1#5%
    \hexnumber@#2#6 }%
%  \MakeRobust#3%
}
\@onlypreamble\set@mathdelimiter
⟨/2ekernel | latexrelease⟩
⟨latexrelease⟩\EndIncludeInRelease
⟨latexrelease⟩\IncludeInRelease{0000/00/00}%
⟨latexrelease⟩                  {\set@mathdelimiter}{make delimiters robust}%
⟨latexrelease⟩
⟨latexrelease⟩\def\set@mathdelimiter#1#2#3#4#5#6{%
⟨latexrelease⟩  \xdef#3{\delimiter"\mathchar@type#4\hexnumber@#1#5%
⟨latexrelease⟩    \hexnumber@#2#6 }}
⟨latexrelease⟩
⟨latexrelease⟩\EndIncludeInRelease
⟨*2ekernel⟩
```

\set@@mathdelimiter

```
\def\set@@mathdelimiter#1#2#3#4#5{%
  \global\delcode`#3="\hexnumber@#1#4\hexnumber@#2#5\relax}
\@onlypreamble\set@@mathdelimiter
```

\DeclareMathRadical

```
\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!

```
  \expandafter\ifx
       \csname\expandafter\@gobble\string#1\endcsname
       \relax
     \let#1\radical
  \fi
  \edef\reserved@a{\noexpand\in@
       {\expandafter\@gobble\string\radical}{\meaning#1}}%
  \reserved@a
  \ifin@
    \expandafter\in@\csname sym#2\expandafter\endcsname
       \expandafter{\group@list}%
    \ifin@
      \expandafter\in@\csname sym#4\expandafter\endcsname
         \expandafter{\group@list}%
      \ifin@
        \begingroup
          \count\z@=#3\relax
          \count\tw@\count\z@
          \divide\count\z@\sixt@@n
          \count@\count\z@
          \multiply\count@\sixt@@n
          \advance\count\tw@-\count@
          \edef\reserved@c{%
            \hexnumber@{\count\z@}\hexnumber@{\count\tw@}}%
          \count\z@=#5\relax
          \count\tw@\count\z@
          \divide\count\z@\sixt@@n
          \count@\count\z@
          \multiply\count@\sixt@@n
          \advance\count\tw@-\count@
          \edef\reserved@d{%
            \hexnumber@{\count\z@}\hexnumber@{\count\tw@}}%
```

Coded inline instead of using `\set@mathradical`

```
%          \expandafter\set@mathradical
%             \csname sym#2\expandafter\endcsname
%             \csname sym#4\endcsname#1%
%             \reserved@c\reserved@d
          \xdef#1{\radical"\expandafter\hexnumber@
                                 \csname sym#2\endcsname\reserved@c
                              \expandafter\hexnumber@
                                 \csname sym#4\endcsname\reserved@d
                  \relax}%
        \endgroup
```

```
      \else
        \@latex@error{Symbol font ‘#4’ is not defined}\@eha
      \fi
    \else
      \@latex@error{Symbol font ‘#2’ is not defined}\@eha
    \fi
  \else
    \@latex@error{Command ‘\string#1’ already defined}\@eha
  \fi
}
\@onlypreamble\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

```
\let\mathalpha\relax
```

`\mathchar@type`

```
\def\mathchar@type#1{%
  \ifodd 2#11 #1\else             % is this non-negative number?
    \ifx#1\mathord 0\else
     \ifx#1\mathop   1\else
       \ifx#1\mathbin 2\else
         \ifx#1\mathrel 3\else
           \ifx#1\mathopen 4\else
             \ifx#1\mathclose 5\else
               \ifx#1\mathpunct 6\else
                   7%             % anything else is variable ord
               \fi
             \fi
           \fi
         \fi
       \fi
     \fi
    \fi
  \fi}
\@onlypreamble\mathchar@type
```

`\DeclareSymbolFontAlphabet`

```
\def\DeclareSymbolFontAlphabet#1#2{%
    \expandafter\DeclareSymbolFontAlphabet@
      \csname \expandafter\@gobble\string#1\space\endcsname{#2}#1}
\@onlypreamble\DeclareSymbolFontAlphabet
```

`\DeclareSymbolFontAlphabet@`

```
\def\DeclareSymbolFontAlphabet@#1#2#3{%
```

We use the switch `\if@tempswa` to decide if we can declare this symbol font alphabet.

```
    \@tempswatrue
```

First check if #2 is known to be a symbol font

```
\expandafter\in@\csname sym#2\expandafter\endcsname
   \expandafter{\group@list}%
\ifin@
```

Check if #1 is defined as a math alphabet defined via \DeclareMathAlphabet:

```
  \expandafter\in@\expandafter#1\expandafter{\alpha@list}%
  \ifin@
```

If so remove it from the \alpha@list and from all math version macros.

```
    \@font@info{Redeclaring math alphabet \string#3}%
    \toks@{}%
    \def\alpha@elt##1##2##3{%
        \ifx##1#1\else\addto@hook\toks@{\alpha@elt##1##2##3}\fi}%
    \alpha@list
    \xdef\alpha@list{\the\toks@}%
```

Now we loop over all versions and remove the math alphabet:

```
    \def\version@elt##1{%
        \begingroup
          \toks@{}%
          \def\getanddefine@fonts####1####2{%
             \addto@hook\toks@{\getanddefine@fonts####1####2}}%
          \def\install@mathalphabet####1####2{%
             \ifx####1#1\else
               \addto@hook\toks@{\install@mathalphabet
                                  ####1{####2}}\fi}%
          ##1%
          \xdef##1{\the\toks@}%
        \endgroup
        }%
    \version@list
  \else
```

If #3 is not defined as a math alphabet check if it is defined at all:

```
    \expandafter\ifx
    \csname\expandafter\@gobble\string#1\space\endcsname
    \relax
```

If it is undefined, fine otherwise check if it is a math alphabet defined via \DeclareSymbolFontAlphabet:

```
    \else
      \edef\reserved@a{%
        \noexpand\in@{\string\use@mathgroup}{\meaning#1}}%
      \reserved@a
      \ifin@
        \@font@info{Redeclaring math alphabet \string#3}%
      \else
```

Since the command #3 is defined to be something which is not a math alphabet we have to skip redefining it.

```
        \@tempswafalse
        \@latex@error{Command `\string#3' already defined}\@eha
      \fi
    \fi
  \fi
 \else
```

Since the symbol font is not known we better skip defining this alphabet.

```
     \@tempswafalse
     \@latex@error{Unknown symbol font `#2'}\@eha
   \fi
   \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.

```
     \def\group@elt##1##2{%
        \expandafter\ifx\csname sym#2\endcsname##1%
        \expandafter\reserved@a\string##2\@nil
        \fi}%
     \def\reserved@a##1##2/##3\@nil{%
        \def\reserved@a{##2}}%
     \group@list
     \toks@{\relax\ifmmode \else \non@alpherr#1\fi}%
     \edef#1{\the\toks@
             \noexpand\use@mathgroup
             \expandafter\noexpand\csname M@\reserved@a\endcsname
             \csname sym#2\endcsname}%
     \def#3{\protect#1}%
   \fi
}
\@onlypreamble\DeclareSymbolFontAlphabet@
⟨/2ekernel⟩
```

File t
ltfssini.dtx

This file contains the top level LaTeX interface to the font selection scheme commands. See other parts of the LaTeX distribution, or *The LaTeX Companion* for higher level documentation of these commands.

38 NFSS Initialisation

Finally, there are six commands that are to be used in LaTeX and that we will therefore protect against expansion at the wrong point: `\fontfamily`, `\fontseries`, `\fontshape`, `\fontsize`, `\selectfont`, and `\mathversion`.

```
⟨*2ekernel⟩
```

38.1 Providing math *versions*

LaTeX provides two *versions*. We call them normal and bold, respectively.

```
\DeclareMathVersion{normal}
\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*:

```
%\DeclareRobustCommand\rmfamily
%        {\not@math@alphabet\rmfamily\mathrm
%         \fontfamily\rmdefault\selectfont}
%\DeclareRobustCommand\sffamily
%        {\not@math@alphabet\sffamily\mathsf
%         \fontfamily\sfdefault\selectfont}
%\DeclareRobustCommand\ttfamily
%        {\not@math@alphabet\ttfamily\mathtt
%         \fontfamily\ttdefault\selectfont}
```

Then the commands changing the *series*:

```
%\DeclareRobustCommand\bfseries
%        {\not@math@alphabet\bfseries\mathbf
%         \fontseries\bfdefault\selectfont}
%\DeclareRobustCommand\mdseries
%        {\not@math@alphabet\mdseries\relax
%         \fontseries\mddefault\selectfont}
\DeclareRobustCommand\upshape
        {\not@math@alphabet\upshape\relax
         \fontshape\updefault\selectfont}
```

Then the commands changing the *shape*:

```
\DeclareRobustCommand\slshape
        {\not@math@alphabet\slshape\relax
         \fontshape\sldefault\selectfont}
\DeclareRobustCommand\scshape
        {\not@math@alphabet\scshape\relax
         \fontshape\scdefault\selectfont}
\DeclareRobustCommand\itshape
        {\not@math@alphabet\itshape\mathit
         \fontshape\itdefault\selectfont}
```

39 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).

```
⟨/2ekernel⟩
⟨*2ekernel | latexrelease⟩
⟨latexrelease⟩\IncludeInRelease{2020/02/02}%
⟨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 LaTeX’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 LaTeX and for use in font packages (or the document preamble) we offer `\DeclareSeriesDefault`. 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` (semibold) 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 specify 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![7]

\DeclareFontSeriesDefault The command to declare font series defaults for the "rm", "sf" or "tt" family.

```
\newcommand\DeclareFontSeriesDefault[3][]{%
  \def\reserved@a{#1}%
```

No optional argument: set up general default.

```
  \ifx\reserved@a\@empty
    \ifcsname #2series\endcsname            % supported are
                                            % \[md/bf]default
      \expandafter\def
         \csname #2default\endcsname{#3}%
    \else
       \@latex@error{Wrong syntax for \string\DeclareFontSeriesDefault}%
          {Mandatory first argument must be ’md’  or ’bf’.}
    \fi
```

Optional argument given, set up specific default.

```
  \else
    \ifcsname #2series@#1\endcsname           % supported are
                                              % \[md/bf]series@[rm/sf/tt]
      \expandafter\edef
         \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).

```
      \expandafter\let
         \csname #2series@#1@kernel\endcsname\@undefined
    \else
       \@latex@error{Wrong syntax for \string\DeclareFontSeriesDefault}%
          {Optional argument must be ’rm’, ’sf’, or ’tt’. \MessageBreak
           Mandatory first argument must be ’md’  or ’bf’.}
    \fi
  \fi
}
```

\mdseries@rm
\mdseries@sf
\mdseries@tt
\bfseries@rm
\bfseries@sf
\bfseries@tt

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).

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.

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.

[7] I 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.

```
\def\bfseries@rm{bx}
\def\bfseries@sf{bx}
\def\bfseries@tt{bx}
```

Frozen version of the kernel defaults so we can see if they have changed.

```
\let\bfseries@rm@kernel\bfseries@rm
\let\bfseries@sf@kernel\bfseries@sf
\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.

```
\def\mdseries@rm{m}
\def\mdseries@sf{m}
\def\mdseries@tt{m}
```

`\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.

```
\def\expand@font@defaults{%
  \edef\rmdef@ult{\rmdefault}%
  \edef\sfdef@ult{\sfdefault}%
  \edef\ttdef@ult{\ttdefault}%
  \edef\bfdef@ult{\bfdefault}%
  \edef\mddef@ult{\mddefault}%
  \edef\famdef@ult{\familydefault}%
}
```

`\bfseries` This document command switches to the bold series.

```
\DeclareRobustCommand\bfseries{%
  \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 the select the correct family default in that case.

```
  \expand@font@defaults
    \ifx\f@family\rmdef@ult      \fontseries\bfseries@rm
    \else\ifx\f@family\sfdef@ult \fontseries\bfseries@sf
    \else\ifx\f@family\ttdef@ult \fontseries\bfseries@tt
```

If not `\bfdefault` is used.

```
    \else                        \fontseries\bfdefault
    \fi\fi\fi
  \selectfont
}
```

`\mdseries` This document command switches to the medium series.

```
\DeclareRobustCommand\mdseries{%
  \not@math@alphabet\mdseries\relax
  \expand@font@defaults
```

```
    \ifx\f@family\rmdef@ult      \fontseries\mdseries@rm
    \else\ifx\f@family\sfdef@ult \fontseries\mdseries@sf
    \else\ifx\f@family\ttdef@ult \fontseries\mdseries@tt
    \else                        \fontseries\mddefault
    \fi\fi\fi
  \selectfont
}
```

\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 has different sets ofshapes available and we certainly don't want to see substituation 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`.

```
\DeclareRobustCommand\rmfamily{%
   \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.

```
   \prepare@family@series@update{rm}\rmdefault
```

Then comes the hook code (by default a no-op) and finally the call to `\selectfont`.

```
   \@rmfamilyhook
   \selectfont}
```

`\sffamily` `\ttfamily` The definitions for `\sffamily` and `\ttfamily` are similar, the differences are only in what font families get checked.

```
\DeclareRobustCommand\sffamily{%
   \not@math@alphabet\sffamily\mathsf
```

```
  \prepare@family@series@update{sf}\sfdefault
  \@sffamilyhook
  \selectfont}
```

```
\DeclareRobustCommand\ttfamily{%
  \not@math@alphabet\ttfamily\mathtt
  \prepare@family@series@update{tt}\ttdefault
  \@ttfamilyhook
  \selectfont}
```

\@rmfamilyhook
\@sffamilyhook
\@ttfamilyhook

By default the hooks do nothing.

```
\let\@rmfamilyhook\@empty
\let\@sffamilyhook\@empty
\let\@ttfamilyhook\@empty
```

\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.

```
⟨*debug⟩
\let\series@change@debug\typeout
\let\series@change@debug\@gobble
⟨/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.

\@meta@family@list

Here is the current definition of that list:

```
\def\@meta@family@list{\@elt{rm}\@elt{sf}\@elt{tt}}
```

```
\def\prepare@family@series@update#1#2{%

 \if@forced@series
⟨+debug⟩ \series@change@debug{No series preparation (forced \f@series)\on@line}%
   \fontfamily#2%
 \else
⟨+debug⟩ \series@change@debug{Prepearing for switching to #1 (#2)\on@line}%
   \expand@font@defaults
```

We prepare for changing the current series. We have to find it before changing the family as discussed above.

```
   \let\target@series@value\@empty
   \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 the meta families, e.g., via `\fontfamily{ptm}\selectfont`
- 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.

```
   \expandafter\edef\csname ??def@ult\endcsname{\f@family}%
```

To find it we loop over the meta family list with a suitable definition of `\@elt`.

```
   \let\@elt\update@series@target@value
      \@meta@family@list
```

Last resort pseudo meta family. Will only be looked at f none of the real ones have matched.

```
      \@elt{??}%
   \let\@elt\relax
```

That will figure out the correct series value to use without updating it. Now we can change the family.

```
   \fontfamily#2%
```

After that we update the series. That code is again like the one above.

```
   \ifx\target@series@value\@empty
⟨+debug⟩ \series@change@debug{Target series still empty ...}%
   \else
     \ifx \f@series\target@series@value
⟨+debug⟩ \series@change@debug{Target series unchanged:
⟨+debug⟩                         \f@series \space = \target@series@value}%
     \else
       \maybe@load@fontshape
⟨+debug⟩ \series@change@debug{Target series:
⟨+debug⟩                         \f@series \space -> \target@series@value}%
       \let\f@series\target@series@value
     \fi
   \fi
 \fi
}
```

`\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 is it is parameterized instead of being written out and only for one block of tests because the code is called reatedly when looping over the meta family list. From the list we get each meta family name in turn.

```
\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.

```
  \def\reserved@a{#1}%
  \ifx\target@meta@family@value\reserved@a   % rm -> rm do nothing
  \else
⟨+debug⟩ \series@change@debug{Trying to match #1: \csname#1def@ult\endcsname
⟨+debug⟩                         \space = \f@family\space ?}%
```

We only "do" something if the current font family matches the current meta family.

```
     \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.

```
       \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.

```
    \expandafter\let\expandafter\reserved@b
                    \csname mdseries@\target@meta@family@value\endcsname
    \expandafter\let\expandafter\reserved@c
                    \csname bfseries@\target@meta@family@value\endcsname
⟨+debug⟩\series@change@debug{Targets for mdseries and bfseries:
⟨+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.

```
    \expandafter\ifx\csname mdseries@#1\endcsname\f@series
⟨+debug⟩  \series@change@debug{mdseries@#1 matched -> \reserved@b}%
                                  \let\target@series@value\reserved@b
    \else\expandafter\ifx\csname bfseries@#1\endcsname\f@series
⟨+debug⟩  \series@change@debug{bfseries@#1 matched -> \reserved@c}%
                                  \let\target@series@value\reserved@c
    \else\ifx\f@series\mddef@ult    \let\target@series@value\reserved@b
⟨+debug⟩  \series@change@debug{mddef@ult matched -> \reserved@b}%
    \else\ifx\f@series\bfdef@ult    \let\target@series@value\reserved@c
⟨+debug⟩  \series@change@debug{bfdef@ult matched -> \reserved@c}%
    \fi\fi\fi\fi
  \fi
\fi
}
```

`\init@series@setup`

This is code to be run at begin document ...

```
\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.

```
\ifx\bfseries@rm@kernel\bfseries@rm
  \expandafter\in@\expandafter{\rmdefault}{cmr,cmss,cmtt,lcmss,lcmtt,lmr,lmss,lmtt}%
  \ifin@ \else \def\bfseries@rm{b}\fi\fi
```

Same approach for `\bfseries@sf` and `\bfseries@tt`:

```
\ifx\bfseries@sf@kernel\bfseries@sf
  \expandafter\in@\expandafter{\sfdefault}{cmr,cmss,cmtt,lcmss,lcmtt,lmr,lmss,lmtt}%
  \ifin@ \else \def\bfseries@sf{b}\fi\fi
\ifx\bfseries@tt@kernel\bfseries@tt
```

```
    \expandafter\in@\expandafter{\ttdefault}{cmr,cmss,cmtt,lcmss,lcmtt,lmr,lmss,lmtt}%
    \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 have to adjust the series defaults accordingly, before starting typesetting.

On the other hand if we still typeset in CM or LM then `\bfdefault` is wrong since it is now saying `b` and not `bx`.

To fix this we run `\rmfamily`, `\sffamily` or `\ttfamily` depending on the situation and this will correct the setup for us.

```
  \expand@font@defaults
  \ifx\famdef@ult\rmdef@ult      \rmfamily
  \else\ifx\famdef@ult\sfdef@ult \sffamily
  \else\ifx\famdef@ult\ttdef@ult \ttfamily
  \fi\fi\fi
}%
```

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.

```
\expandafter\let\csname ver@mweights.sty\endcsname\fmtversion
```

```
⟨/2ekernel | latexrelease⟩
⟨latexrelease⟩\EndIncludeInRelease
⟨latexrelease⟩\IncludeInRelease{0000/00/00}%
⟨latexrelease⟩               {\DeclareFontSeriesDefault}{Custom series}%
⟨latexrelease⟩
⟨latexrelease⟩\let\DeclareFontSeriesDefault\@undefined
⟨latexrelease⟩\let\bfseries@rm\@undefined
⟨latexrelease⟩\let\bfseries@sf\@undefined
⟨latexrelease⟩\let\bfseries@tt\@undefined
⟨latexrelease⟩\let\bfseries@rm@kernel\@undefined
⟨latexrelease⟩\let\bfseries@sf@kernel\@undefined
⟨latexrelease⟩\let\bfseries@tt@kernel\@undefined
⟨latexrelease⟩\let\mdseries@rm\@undefined
⟨latexrelease⟩\let\mdseries@sf\@undefined
⟨latexrelease⟩\let\mdseries@tt\@undefined
⟨latexrelease⟩\let\expand@font@defaults\@undefined
⟨latexrelease⟩\expandafter\let\csname ver@mweights.sty\endcsname\@undefined
⟨latexrelease⟩
⟨latexrelease⟩\DeclareRobustCommand\bfseries
⟨latexrelease⟩       {\not@math@alphabet\bfseries\mathbf
⟨latexrelease⟩        \fontseries\bfdefault\selectfont}
⟨latexrelease⟩\DeclareRobustCommand\mdseries
⟨latexrelease⟩      {\not@math@alphabet\mdseries\relax
⟨latexrelease⟩        \fontseries\mddefault\selectfont}
⟨latexrelease⟩\DeclareRobustCommand\rmfamily
⟨latexrelease⟩      {\not@math@alphabet\rmfamily\mathrm
⟨latexrelease⟩        \fontfamily\rmdefault\selectfont}
⟨latexrelease⟩\DeclareRobustCommand\sffamily
```

```
⟨latexrelease⟩        {\not@math@alphabet\sffamily\mathsf
⟨latexrelease⟩         \fontfamily\sfdefault\selectfont}
⟨latexrelease⟩\DeclareRobustCommand\ttfamily
⟨latexrelease⟩        {\not@math@alphabet\ttfamily\mathtt
⟨latexrelease⟩         \fontfamily\ttdefault\selectfont}
⟨latexrelease⟩
⟨latexrelease⟩\let\@rmfamilyhook\@undefined
⟨latexrelease⟩\let\@sffamilyhook\@undefined
⟨latexrelease⟩\let\@ttfamilyhook\@undefined
⟨latexrelease⟩\let\@meta@family@list\@undefined
⟨latexrelease⟩\let\prepare@family@series@update\@undefined
⟨latexrelease⟩\let\update@series@target@value\@undefined
⟨latexrelease⟩
```

This is always called in `\document` so don't make it undefined.

```
⟨latexrelease⟩\let\init@series@setup\relax
⟨latexrelease⟩
⟨latexrelease⟩\EndIncludeInRelease
⟨*2ekernel⟩
```

40 Supporting nested emphasis

By default LaTeX 2$_\varepsilon$ supports two levels of nested emphasis: if the current font has an upright shape then it switches to `\itshape` otherwise to `\eminnershape` (which defaults to `\upshape`). This means nested emphasis will ocssilate between italic and upright shapes.

Sometimes it would be nice to allow for a more lengthly 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 clist of declaration. Spaces in the argument are dropped to avoid surious 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.[8] Further nested calls restart at the beginning.

```
⟨/2ekernel⟩
⟨*2ekernel | latexrelease⟩
```

[8] Maybe 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 ...

```
⟨latexrelease⟩\IncludeInRelease{2020/02/02}%
⟨latexrelease⟩                  {\DeclareEmphSequence}{Nested emph}%
\def\DeclareEmphSequence#1{%
  \protected@edef\emfontdeclare@clist{\zap@space#1, \@empty\emforce\emreset}%
}
```

By default the it is empty, in which case `\eminnershape` is used by LaTeX.

```
\let\emfontdeclare@clist\@empty
```

`\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:

```
\DeclareRobustCommand\emreset{\upshape\ulcshape}
```

`\em` The new definition for `\em` (and implicitly `\emph` is like it was before if `\emfontdeclare@clist` is empty.

```
\DeclareRobustCommand\em{%
  \@nomath\em
  \ifx\emfontdeclare@clist\@empty
    \ifdim \fontdimen\@ne\font >\z@
      \eminnershape \else \itshape \fi
  \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.

```
  \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.

```
    \expandafter\do@emfont@update\emfontdeclare@clist\do@emfont@update
  \fi
}
```

```
\def\eminnershape{\upshape}
```

`\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`.

```
\def\do@emfont@update#1,#2\do@emfont@update{%
```

First action is to alter the list and move the first entry to the end

```
  \def\emfontdeclare@clist{#2,#1}%
```

Then we execute current declaration. Appending `\selectfont` means one can write just `\fontshape{it}}` and that works then too.

```
%  \typeout{Use: \detokenize{#1}}%
  #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.

```
  \expandafter\ifx\csname \curr@fontshape/\f@size\em@force
```

For the comparison with \ifx we have to exand \em@currfont once as the relevant info is inside.

```
                          \expandafter\endcsname
                 \em@currfont
  \expandafter\do@emfont@update\emfontdeclare@clist\do@emfont@update
```

If \emforce was used, we have to undo its effect:

```
  \else
    \let\em@force\@empty
  \fi
}
```

\emforce
\em@force

The definition of \emforce is simple: change \em@force to make the above test always invalid.

```
\protected\def\emforce{\def\em@force{/}}
\let\em@force\@empty
⟨/2ekernel | latexrelease⟩
⟨latexrelease⟩\EndIncludeInRelease
```

\em
\eminnershape

These are the older definitions for \em, prior to 2020.

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.

```
⟨latexrelease⟩\IncludeInRelease{2015/01/01}{\DeclareEmphSequence}{Nested emph}%
⟨latexrelease⟩\let\DeclareEmphSequence\@undefined
⟨latexrelease⟩\let\emfontdeclare@clist\@undefined
⟨latexrelease⟩\let\emreset\@undefined
⟨latexrelease⟩\let\do@emfont@update\@undefined
⟨latexrelease⟩\let\emforce\@undefined
⟨latexrelease⟩\let\em@force\@undefined
⟨latexrelease⟩
⟨latexrelease⟩\DeclareRobustCommand\em
⟨latexrelease⟩        {\@nomath\em \ifdim \fontdimen\@ne\font >\z@
⟨latexrelease⟩                       \eminnershape \else \itshape \fi}%
⟨latexrelease⟩\EndIncludeInRelease
⟨latexrelease⟩
⟨latexrelease⟩\IncludeInRelease{0000/00/00}{\DeclareEmphSequence}{Nested emph}%
⟨latexrelease⟩\DeclareRobustCommand\em
⟨latexrelease⟩        {\@nomath\em \ifdim \fontdimen\@ne\font >\z@
⟨latexrelease⟩                       \upshape \else \itshape \fi}%
⟨latexrelease⟩\let\eminnershape\@undefined
⟨latexrelease⟩\EndIncludeInRelease
⟨*2ekernel⟩
```

\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.

```
\def\not@math@alphabet#1#2{%
  \relax
  \ifmmode
    \@latex@error{Command \noexpand#1invalid in math mode}%
       {%
        Please
        \ifx#2\relax
```

```
            define a new math alphabet^^J%
            if you want to use a special font in math mode%
          \else
```

We have to a `\noexpand` below to prevent expansion of `#2`. In case of `#1` we can omit this (due to the current definition of robust commands since they do come out right there :-).

```
            use the math alphabet \noexpand#2instead of
            the #1command%
         \fi
         .
        }%
   \fi}
```

Finally we provide two abbreviations to switch to the LaTeX *versions*.

```
\DeclareRobustCommand\boldmath{\@nomath\boldmath
             \mathversion{bold}}
\DeclareRobustCommand\unboldmath{\@nomath\unboldmath
             \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`.

```
\def\math@version{normal}
```

40.1 Miscellaneous

`\newfont` `\symbol` We start by defining a few macros that are part of standard LaTeX's user interface. The use of these functions is not encouraged, but they will allow to process older documents without changes to the source.

```
\def\newfont#1#2{\@ifdefinable#1{\font#1=#2\relax}}
```

```
\DeclareRobustCommand\symbol[1]{\char #1\relax}
```

`\@setfontsize` `\@setsize` This abbreviation is used by LaTeX's user level size changing commands, such as `\large`.

```
\def\@setfontsize#1#2#3{\@nomath#1%
```

For the benefit of people relying on keeping the name of the current font command saved in `\@currsize` we define it. To ensure that `\@setfontsize` keeps being robust we omit this assignment during times where `\protect` differs from `\@typeset@protect`.

```
    \ifx\protect\@typeset@protect
      \let\@currsize#1%
    \fi
    \fontsize{#2}{#3}\selectfont}
```

For compatibility we also define `\@setsize` the 209 command

```
⟨*compat⟩
\def\@setsize#1#2#3#4{\@setfontsize#1{#4}{#2}}
⟨/compat⟩
```

`\hexnumber@` To set up LaTeX's special math character definitions we first provide a macro to generate hexadecimal numbers. It is a rather simple `\ifcase`.

```
\def\hexnumber@#1{\ifcase\number#1
 0\or 1\or 2\or 3\or 4\or 5\or 6\or 7\or 8\or
 9\or A\or B\or C\or D\or E\or F\fi}
```

\nfss@text In it 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.

```
\def\nfss@text#1{{\mbox{#1}}}
```

\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.

```
%\DeclareRobustCommand\copyright
%    {{\ooalign{\hfil
%     \raise.07ex\hbox{\mdseries\upshape c}\hfil\crcr
%     \mathhexbox20D}}}
```

\normalfont
\reset@font The macro `\reset@font` is used in LaTeX 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 LaTeX version above but nevertheless are able to run all kind of mixtures.

The user interface name for `\reset@font` is `\normalfont`:

```
⟨/2ekernel⟩
⟨*2ekernel | latexrelease⟩
⟨latexrelease⟩\IncludeInRelease{2020/02/02}%
⟨latexrelease⟩                    {\normalfont}{Add hook to \normalfont}%
\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.

```
   \fontencoding\encodingdefault
   \edef\f@family{\familydefault}%
   \edef\f@series{\seriesdefault}%
   \edef\f@shape{\shapedefault}%
   \@defaultfamilyhook
   \selectfont}
\let\reset@font\normalfont
```

\@defaultfamilyhook By default the hooks do nothing.

```
\let\@defaultfamilyhook\@empty
⟨/2ekernel | latexrelease⟩
⟨latexrelease⟩\EndIncludeInRelease
⟨latexrelease⟩\IncludeInRelease{0000/00/00}%
```

```
⟨latexrelease⟩                 {\normalfont}{Add hook to \normalfont}%
⟨latexrelease⟩
⟨latexrelease⟩\DeclareRobustCommand\normalfont
⟨latexrelease⟩         {\usefont\encodingdefault
⟨latexrelease⟩                  \familydefault
⟨latexrelease⟩                  \seriesdefault
⟨latexrelease⟩                  \shapedefault
⟨latexrelease⟩                  \relax}
⟨latexrelease⟩\let\reset@font\normalfont
⟨latexrelease⟩
⟨latexrelease⟩\let\@defaultfamilyhook\@undefined
⟨latexrelease⟩
⟨latexrelease⟩\EndIncludeInRelease
⟨*2ekernel⟩
```

We left out the special LaTeX 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.

```
\def\not@base#1{\@latex@error
   {Command \noexpand#1not provided in base LaTeX2e}%
   {Load the latexsym or the amsfonts package to
    define this symbol}}
\def\mho{\not@base\mho}
\def\Join{\not@base\Join}
\def\Box{\not@base\Box}
\def\Diamond{\not@base\Diamond}
\def\leadsto{\not@base\leadsto}
\def\sqsubset{\not@base\sqsubset}
\def\sqsupset{\not@base\sqsupset}
\def\lhd{\not@base\lhd}
\def\unlhd{\not@base\unlhd}
\def\rhd{\not@base\rhd}
\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.

```
\DeclareErrorFont{OT1}{cmr}{m}{n}{10}  %% don't modify this setting
                                       %% overwrite it in fontdef.cfg
                                       %% 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`.

```
\fontfamily{cmr}
\fontseries{m}
\fontshape{n}
\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:

```
\def\@fontenc@load@list{\@elt{T1,OT1}}
```

We now load the customizable parts of NFSS.

```
\InputIfFileExists{fonttext.cfg}
           {\typeout{====================================^^J%
                     ^^J%
                      Local config file fonttext.cfg used^^J%
                     ^^J%
                     ====================================}%
             \def\@addtofilelist##1{\xdef\@filelist{\@filelist,##1}}%
            }
           {\input{fonttext.ltx}}
\let\@addtofilelist\@gobble
```

Ditto for math although I don't think that we will get a lot of customisation :-)

```
\InputIfFileExists{fontmath.cfg}
           {\typeout{====================================^^J%
                     ^^J%
                      Local config file fontmath.cfg used^^J%
                     ^^J%
                     ====================================}%
             \def\@addtofilelist##1{\xdef\@filelist{\@filelist,##1}}%
            }
           {\input{fontmath.ltx}}
\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`.

```
\InputIfFileExists{preload.cfg}
           {\typeout{====================================^^J%
                     ^^J%
                      Local config file preload.cfg used^^J%
                     ^^J%
                     =====================================}%
             \def\@addtofilelist##1{\xdef\@filelist{\@filelist,##1}}%
            }
           {\input{preload.ltx}}
\let\@addtofilelist\@gobble
```

`\@acci` `\@accii` `\@acciii` We also save the values of some accents in `\@acci`, `\@accii` and `\@acciii` so they can be restored by a `minipage` inside a `tabbing` environment.

```
\let\@acci\' \let\@accii\` \let\@acciii\=
```

`\cal` `\mit` Here were the two old ⟨*alphabet identifiers*⟩.

⟨/2ekernel⟩

File u
fontdef.dtx

¡-latexrelease¿ [2020/02/11 v3.0g LaTeX Kernel (¡-latexrelease¿ font setup)]

41 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.

42 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 LaTeX 2ε 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 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 TEX 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 TEX 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 LATEX 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.

43 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}}
```

44 A driver for this document

The next bit of code contains the documentation driver file for TEX, i.e. the file that will produce the documentation you are currently reading. It will be extracted from this file by the DOCSTRIP program.

```
⟨*driver⟩
\documentclass{ltxdoc}
\GetFileInfo{fontdef.dtx}
```

```
\begin{document}
  \DocInput{fontdef.dtx}
\end{document}
⟨/driver⟩
```

45 The fonttext.ltx file

The identification is done earlier on with a `\ProvidesFile` declaration.

```
⟨*text⟩
\typeout{=== Don't modify this file, use a .cfg file instead ===^^J}
```

45.1 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 declaration `\DeclareFontEncoding`, 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.

```
\input {omlenc.def}
\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`.

```
\input {ot1enc.def}
\input  {t1enc.def}
```

```
\input{ts1enc.def}
```

```
\ifx\Umathchar\@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.

```
\fontencoding{OT1}
```

The initial `fontenc` package load list if an 8-bit TEX engine is used:

```
\def\@fontenc@load@list{\@elt{T1,OT1}}
```

```
\def\rmsubstdefault{cmr}
\def\sfsubstdefault{cmss}
\def\ttsubstdefault{cmtt}
\LoadFontDefinitionFile{TS1}{cmr}
```

```
\else
```

Unicode.

```
\input {tuenc.def}
\fontencoding{TU}
```

The initial `fontenc` package load list if a Unicode engine is used:

```
\def\@fontenc@load@list{\@elt{TU}}
```

```
\DeclareFontSubstitution{TU}{lmr}{m}{n}
\LoadFontDefinitionFile{TU}{lmr}
\LoadFontDefinitionFile{TU}{lmss}
\LoadFontDefinitionFile{TU}{lmtt}
```

```
\def\rmsubstdefault{lmr}
\def\sfsubstdefault{lmss}
\def\ttsubstdefault{lmtt}
\LoadFontDefinitionFile{TS1}{lmr}
```

```
\DeclareFontSubstitution{TU}{lmr}{m}{n}
```

End of Unicode branch.

```
\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!

```
\DeclareFontEncodingDefaults{}{}
```

Then we define the default substitution for every encoding. This release of LaTeX 2ε 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.

```
\DeclareFontSubstitution{T1}{cmr}{m}{n}
\DeclareFontSubstitution{OT1}{cmr}{m}{n}
```

For every encoding declaration, LaTeX 2ε 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. LaTeX 2ε 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 LaTeX 2ε 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.

```
\begingroup
\nfss@catcodes
\input  {t1cmr.fd}
```

```
\input  {ot1cmr.fd}
\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.)

```
\begingroup
\nfss@catcodes
\input {ot1cmss.fd}
\input {ot1cmtt.fd}
\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.

```
\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.

45.2 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
\rmdefault
\sfdefault
\ttdefault

The following three definitions set up the meaning for \rmfamily, \sffamily, and \ttfamily.

```
\ifx\Umathchar\@undefined
\newcommand\encodingdefault{OT1}
\newcommand\rmdefault{cmr}
\newcommand\sfdefault{cmss}
\newcommand\ttdefault{cmtt}
\else
\newcommand\encodingdefault{TU}
\newcommand\rmdefault{lmr}
\fontfamily{\rmdefault}
\newcommand\sfdefault{lmss}
\newcommand\ttdefault{lmtt}
\fi
⟨/text⟩
⟨latexrelease⟩\IncludeInRelease{2017/01/01}%
⟨latexrelease⟩                  {\encodingdefault}{TU encoding default}%
⟨latexrelease⟩\ifx\Umathchar\@undefined
```

```
⟨latexrelease⟩\renewcommand\encodingdefault{OT1}
⟨latexrelease⟩\fontencoding{\encodingdefault}
⟨latexrelease⟩\renewcommand\rmdefault{cmr}
⟨latexrelease⟩\fontfamily{\rmdefault}
⟨latexrelease⟩\renewcommand\sfdefault{cmss}
⟨latexrelease⟩\renewcommand\ttdefault{cmtt}
⟨latexrelease⟩\else
⟨latexrelease⟩\renewcommand\encodingdefault{TU}
⟨latexrelease⟩%done in everyjob\fontencoding{\encodingdefault}
⟨latexrelease⟩\renewcommand\rmdefault{lmr}
⟨latexrelease⟩\fontfamily{\rmdefault}
⟨latexrelease⟩\renewcommand\sfdefault{lmss}
⟨latexrelease⟩\renewcommand\ttdefault{lmtt}
⟨latexrelease⟩\fi
⟨latexrelease⟩\EndIncludeInRelease
⟨latexrelease⟩\IncludeInRelease{0000/00/00}%
⟨latexrelease⟩                 {\encodingdefault}{TU encoding default}%
⟨latexrelease⟩\fontencoding{OT1}
⟨latexrelease⟩\renewcommand\encodingdefault{OT1}
⟨latexrelease⟩\fontencoding{\encodingdefault}
⟨latexrelease⟩\renewcommand\rmdefault{cmr}
⟨latexrelease⟩\fontfamily{\rmdefault}
⟨latexrelease⟩\renewcommand\sfdefault{cmss}
⟨latexrelease⟩\renewcommand\ttdefault{cmtt}
⟨latexrelease⟩\EndIncludeInRelease
⟨*text⟩
```

\bfdefault
\mddefault

Series changing commands are influenced by the following hooks.

```
\newcommand\bfdefault{b}  % overwritten below
\newcommand\mddefault{m}
```

\itdefault
\sldefault
\scdefault
\updefault

Shape changing commands use the following hooks.

```
\newcommand\itdefault{it}
\newcommand\sldefault{sl}
\newcommand\scdefault{sc}
\newcommand\updefault{up}  % overwritten below
```

```
⟨/text⟩
⟨*text | latexrelease⟩
⟨latexrelease⟩\IncludeInRelease{2020/02/02}%
⟨latexrelease⟩                 {\updefault}{font defaults change}%
\renewcommand\updefault{up}
\renewcommand\bfdefault{b}
⟨/text | latexrelease⟩
⟨latexrelease⟩\EndIncludeInRelease
⟨latexrelease⟩\IncludeInRelease{0000/00/00}%
⟨latexrelease⟩                 {\updefault}{font defaults change}%
⟨latexrelease⟩
⟨latexrelease⟩\renewcommand\updefault{n}
⟨latexrelease⟩\renewcommand\bfdefault{bx}
⟨latexrelease⟩\EndIncludeInRelease
⟨*text⟩
```

\familydefault
\seriesdefault
\shapedefault

Finally we have the hooks that describe the behaviour of the `\normalfont` command. To stay portable, the definition of `\encodingdefault` should *not* be changed and should match the setting above for \fontencoding. All other values can be set according to your taste.

```
\newcommand\familydefault{\rmdefault}
\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.

```
% \changes{v3.0e}{2019/12/17}{Set \cs{shapedefault} explicitly to ‘‘n’’}
\newcommand\shapedefault{n}
```

This finishes the low-level setup in fonttext.ltx.

```
⟨/text⟩
```

46 The fontmath.ltx file

The identification is done earlier on with a \ProvidesFile declaration.

```
⟨*math⟩
\typeout{=== Don’t modify this file, use a .cfg file instead ===^^J}
```

46.1 The font encodings used

```
\DeclareFontEncoding{OML}{}{}
\DeclareFontEncoding{OMS}{}{}
\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.

```
\DeclareFontEncoding{U}{}{\noaccents@}
```

The encodings for math are next:

```
\DeclareFontSubstitution{OML}{cmm}{m}{it}
\DeclareFontSubstitution{OMS}{cmsy}{m}{n}
\DeclareFontSubstitution{OMX}{cmex}{m}{n}
\DeclareFontSubstitution{U}{cmr}{m}{n}
```

```
\begingroup
\nfss@catcodes
\input  {omlcmm.fd}
\input  {omscmsy.fd}
\input  {omxcmex.fd}
\input  {ucmr.fd}
\endgroup
```

46.1.1 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.

```
\DeclareSymbolFont{operators}   {OT1}{cmr} {m}{n}
\DeclareSymbolFont{letters}     {OML}{cmm} {m}{it}
\DeclareSymbolFont{symbols}     {OMS}{cmsy}{m}{n}
\DeclareSymbolFont{largesymbols}{OMX}{cmex}{m}{n}
```

```
\SetSymbolFont{operators}{bold}{OT1}{cmr} {bx}{n}
\SetSymbolFont{letters}  {bold}{OML}{cmm} {b}{it}
\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.

```
\DeclareSymbolFontAlphabet{\mathrm}    {operators}
\DeclareSymbolFontAlphabet{\mathnormal}{letters}
\DeclareSymbolFontAlphabet{\mathcal}   {symbols}
\DeclareMathAlphabet      {\mathbf}{OT1}{cmr}{bx}{n}
\DeclareMathAlphabet      {\mathsf}{OT1}{cmss}{m}{n}
\DeclareMathAlphabet      {\mathit}{OT1}{cmr}{m}{it}
\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.

```
\SetMathAlphabet\mathsf{bold}{OT1}{cmss}{bx}{n}
\SetMathAlphabet\mathit{bold}{OT1}{cmr}{bx}{it}
```

46.2 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).

```
  \DeclareMathSizes{5}{5}{5}{5}
  \DeclareMathSizes{6}{6}{5}{5}
  \DeclareMathSizes{7}{7}{5}{5}
  \DeclareMathSizes{8}{8}{6}{5}
  \DeclareMathSizes{9}{9}{6}{5}
  \DeclareMathSizes{\@xpt}{\@xpt}{7}{5}
  \DeclareMathSizes{\@xipt}{\@xipt}{8}{6}
  \DeclareMathSizes{\@xiipt}{\@xiipt}{8}{6}
  \DeclareMathSizes{\@xivpt}{\@xivpt}{\@xpt}{7}
  \DeclareMathSizes{\@xviipt}{\@xviipt}{\@xiipt}{\@xpt}
  \DeclareMathSizes{\@xxpt}{\@xxpt}{\@xivpt}{\@xiipt}
  \DeclareMathSizes{\@xxvpt}{\@xxvpt}{\@xxpt}{\@xviipt}
```

46.3 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.

46.3.1 The letters

```
\DeclareMathSymbol{a}{\mathalpha}{letters}{`a}
\DeclareMathSymbol{b}{\mathalpha}{letters}{`b}
\DeclareMathSymbol{c}{\mathalpha}{letters}{`c}
\DeclareMathSymbol{d}{\mathalpha}{letters}{`d}
\DeclareMathSymbol{e}{\mathalpha}{letters}{`e}
\DeclareMathSymbol{f}{\mathalpha}{letters}{`f}
\DeclareMathSymbol{g}{\mathalpha}{letters}{`g}
\DeclareMathSymbol{h}{\mathalpha}{letters}{`h}
\DeclareMathSymbol{i}{\mathalpha}{letters}{`i}
\DeclareMathSymbol{j}{\mathalpha}{letters}{`j}
\DeclareMathSymbol{k}{\mathalpha}{letters}{`k}
\DeclareMathSymbol{l}{\mathalpha}{letters}{`l}
\DeclareMathSymbol{m}{\mathalpha}{letters}{`m}
\DeclareMathSymbol{n}{\mathalpha}{letters}{`n}
\DeclareMathSymbol{o}{\mathalpha}{letters}{`o}
\DeclareMathSymbol{p}{\mathalpha}{letters}{`p}
\DeclareMathSymbol{q}{\mathalpha}{letters}{`q}
\DeclareMathSymbol{r}{\mathalpha}{letters}{`r}
\DeclareMathSymbol{s}{\mathalpha}{letters}{`s}
\DeclareMathSymbol{t}{\mathalpha}{letters}{`t}
\DeclareMathSymbol{u}{\mathalpha}{letters}{`u}
\DeclareMathSymbol{v}{\mathalpha}{letters}{`v}
\DeclareMathSymbol{w}{\mathalpha}{letters}{`w}
\DeclareMathSymbol{x}{\mathalpha}{letters}{`x}
\DeclareMathSymbol{y}{\mathalpha}{letters}{`y}
\DeclareMathSymbol{z}{\mathalpha}{letters}{`z}

\DeclareMathSymbol{A}{\mathalpha}{letters}{`A}
\DeclareMathSymbol{B}{\mathalpha}{letters}{`B}
\DeclareMathSymbol{C}{\mathalpha}{letters}{`C}
\DeclareMathSymbol{D}{\mathalpha}{letters}{`D}
\DeclareMathSymbol{E}{\mathalpha}{letters}{`E}
\DeclareMathSymbol{F}{\mathalpha}{letters}{`F}
\DeclareMathSymbol{G}{\mathalpha}{letters}{`G}
\DeclareMathSymbol{H}{\mathalpha}{letters}{`H}
\DeclareMathSymbol{I}{\mathalpha}{letters}{`I}
\DeclareMathSymbol{J}{\mathalpha}{letters}{`J}
\DeclareMathSymbol{K}{\mathalpha}{letters}{`K}
\DeclareMathSymbol{L}{\mathalpha}{letters}{`L}
\DeclareMathSymbol{M}{\mathalpha}{letters}{`M}
\DeclareMathSymbol{N}{\mathalpha}{letters}{`N}
\DeclareMathSymbol{O}{\mathalpha}{letters}{`O}
\DeclareMathSymbol{P}{\mathalpha}{letters}{`P}
\DeclareMathSymbol{Q}{\mathalpha}{letters}{`Q}
\DeclareMathSymbol{R}{\mathalpha}{letters}{`R}
\DeclareMathSymbol{S}{\mathalpha}{letters}{`S}
\DeclareMathSymbol{T}{\mathalpha}{letters}{`T}
\DeclareMathSymbol{U}{\mathalpha}{letters}{`U}
\DeclareMathSymbol{V}{\mathalpha}{letters}{`V}
\DeclareMathSymbol{W}{\mathalpha}{letters}{`W}
```

```
\DeclareMathSymbol{X}{\mathalpha}{letters}{`X}
\DeclareMathSymbol{Y}{\mathalpha}{letters}{`Y}
\DeclareMathSymbol{Z}{\mathalpha}{letters}{`Z}
```

46.3.2 The digits

```
\DeclareMathSymbol{0}{\mathalpha}{operators}{`0}
\DeclareMathSymbol{1}{\mathalpha}{operators}{`1}
\DeclareMathSymbol{2}{\mathalpha}{operators}{`2}
\DeclareMathSymbol{3}{\mathalpha}{operators}{`3}
\DeclareMathSymbol{4}{\mathalpha}{operators}{`4}
\DeclareMathSymbol{5}{\mathalpha}{operators}{`5}
\DeclareMathSymbol{6}{\mathalpha}{operators}{`6}
\DeclareMathSymbol{7}{\mathalpha}{operators}{`7}
\DeclareMathSymbol{8}{\mathalpha}{operators}{`8}
\DeclareMathSymbol{9}{\mathalpha}{operators}{`9}
```

46.3.3 Punctuation, brace, etc. keys

```
\DeclareMathSymbol{!}{\mathclose}{operators}{"21}
\DeclareMathSymbol{*}{\mathbin}{symbols}{"03} % \ast
\DeclareMathSymbol{+}{\mathbin}{operators}{"2B}
\DeclareMathSymbol{,}{\mathpunct}{letters}{"3B}
\DeclareMathSymbol{-}{\mathbin}{symbols}{"00}
\DeclareMathSymbol{.}{\mathord}{letters}{"3A}
\DeclareMathSymbol{:}{\mathrel}{operators}{"3A}
\DeclareMathSymbol{;}{\mathpunct}{operators}{"3B}
\DeclareMathSymbol{=}{\mathrel}{operators}{"3D}
\DeclareMathSymbol{?}{\mathclose}{operators}{"3F}
```

The following symbols are defined as delimiters below which automatically defines them as math symbols.

```
%\DeclareMathSymbol{(}{\mathopen}{operators}{"28}
%\DeclareMathSymbol{)}{\mathclose}{operators}{"29}
%\DeclareMathSymbol{/}{\mathord}{letters}{"3D}
%\DeclareMathSymbol{[}{\mathopen}{operators}{"5B}
%\DeclareMathSymbol{]}{\mathclose}{operators}{"5D}
%\DeclareMathSymbol{|}{\mathord}{symbols}{"6A}
%\DeclareMathSymbol{<}{\mathrel}{letters}{"3C}
%\DeclareMathSymbol{>}{\mathrel}{letters}{"3E}
```

Should all of the following being activated by default? Probably not.

```
%\DeclareMathSymbol{`\{}{\mathopen}{symbols}{"66}
%\DeclareMathSymbol{`\}}{\mathclose}{symbols}{"67}
%\DeclareMathSymbol{`\\}{\mathord}{symbols}{"6E} % \backslash
\mathcode`\ ="8000 % \space
\mathcode`\'="8000 % ^\prime
\mathcode`\_="8000 % \_
```

46.3.4 Delimitercodes for characters

[to be completed]

Finally, IniTEX sets all `\delcode` values to -1, except `\delcode`.=0`

```
\DeclareMathDelimiter{(}{\mathopen} {operators}{"28}{largesymbols}{"00}
\DeclareMathDelimiter{)}{\mathclose}{operators}{"29}{largesymbols}{"01}
\DeclareMathDelimiter{[}{\mathopen} {operators}{"5B}{largesymbols}{"02}
\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 TEX. Therefore we explicitly redefine the math symbol definitions for these symbols afterwards.

```
\DeclareMathDelimiter{<}{\mathopen}{symbols}{"68}{largesymbols}{"0A}
\DeclareMathDelimiter{>}{\mathclose}{symbols}{"69}{largesymbols}{"0B}
\DeclareMathSymbol{<}{\mathrel}{letters}{"3C}
\DeclareMathSymbol{>}{\mathrel}{letters}{"3E}
```

And here is another case where the non-delimiter version produces a glyph different from the delimiter version.

```
\DeclareMathDelimiter{/}{\mathord}{operators}{"2F}{largesymbols}{"0E}
\DeclareMathSymbol{/}{\mathord}{letters}{"3D}
```

```
\DeclareMathDelimiter{|}{\mathord}{symbols}{"6A}{largesymbols}{"0C}
```

```
\expandafter\DeclareMathDelimiter\@backslashchar
                       {\mathord}{symbols}{"6E}{largesymbols}{"0F}
```

N.B. { and } should NOT get delcodes; otherwise parameter grouping fails!

46.4 Symbols accessed via control sequences

46.4.1 Greek letters

```
\DeclareMathSymbol{\alpha}{\mathord}{letters}{"0B}
\DeclareMathSymbol{\beta}{\mathord}{letters}{"0C}
\DeclareMathSymbol{\gamma}{\mathord}{letters}{"0D}
\DeclareMathSymbol{\delta}{\mathord}{letters}{"0E}
\DeclareMathSymbol{\epsilon}{\mathord}{letters}{"0F}
\DeclareMathSymbol{\zeta}{\mathord}{letters}{"10}
\DeclareMathSymbol{\eta}{\mathord}{letters}{"11}
\DeclareMathSymbol{\theta}{\mathord}{letters}{"12}
\DeclareMathSymbol{\iota}{\mathord}{letters}{"13}
\DeclareMathSymbol{\kappa}{\mathord}{letters}{"14}
\DeclareMathSymbol{\lambda}{\mathord}{letters}{"15}
\DeclareMathSymbol{\mu}{\mathord}{letters}{"16}
\DeclareMathSymbol{\nu}{\mathord}{letters}{"17}
\DeclareMathSymbol{\xi}{\mathord}{letters}{"18}
\DeclareMathSymbol{\pi}{\mathord}{letters}{"19}
\DeclareMathSymbol{\rho}{\mathord}{letters}{"1A}
\DeclareMathSymbol{\sigma}{\mathord}{letters}{"1B}
\DeclareMathSymbol{\tau}{\mathord}{letters}{"1C}
\DeclareMathSymbol{\upsilon}{\mathord}{letters}{"1D}
\DeclareMathSymbol{\phi}{\mathord}{letters}{"1E}
\DeclareMathSymbol{\chi}{\mathord}{letters}{"1F}
\DeclareMathSymbol{\psi}{\mathord}{letters}{"20}
\DeclareMathSymbol{\omega}{\mathord}{letters}{"21}
\DeclareMathSymbol{\varepsilon}{\mathord}{letters}{"22}
\DeclareMathSymbol{\vartheta}{\mathord}{letters}{"23}
\DeclareMathSymbol{\varpi}{\mathord}{letters}{"24}
\DeclareMathSymbol{\varrho}{\mathord}{letters}{"25}
\DeclareMathSymbol{\varsigma}{\mathord}{letters}{"26}
\DeclareMathSymbol{\varphi}{\mathord}{letters}{"27}
\DeclareMathSymbol{\Gamma}{\mathalpha}{operators}{"00}
```

```
\DeclareMathSymbol{\Delta}{\mathalpha}{operators}{"01}
\DeclareMathSymbol{\Theta}{\mathalpha}{operators}{"02}
\DeclareMathSymbol{\Lambda}{\mathalpha}{operators}{"03}
\DeclareMathSymbol{\Xi}{\mathalpha}{operators}{"04}
\DeclareMathSymbol{\Pi}{\mathalpha}{operators}{"05}
\DeclareMathSymbol{\Sigma}{\mathalpha}{operators}{"06}
\DeclareMathSymbol{\Upsilon}{\mathalpha}{operators}{"07}
\DeclareMathSymbol{\Phi}{\mathalpha}{operators}{"08}
\DeclareMathSymbol{\Psi}{\mathalpha}{operators}{"09}
\DeclareMathSymbol{\Omega}{\mathalpha}{operators}{"0A}
```

46.4.2 Ordinary symbols

```
\DeclareMathSymbol{\aleph}{\mathord}{symbols}{"40}
\DeclareMathSymbol{\imath}{\mathord}{letters}{"7B}
\DeclareMathSymbol{\jmath}{\mathord}{letters}{"7C}
\DeclareMathSymbol{\ell}{\mathord}{letters}{"60}
\DeclareMathSymbol{\wp}{\mathord}{letters}{"7D}
\DeclareMathSymbol{\Re}{\mathord}{symbols}{"3C}
\DeclareMathSymbol{\Im}{\mathord}{symbols}{"3D}
\DeclareMathSymbol{\partial}{\mathord}{letters}{"40}
\DeclareMathSymbol{\infty}{\mathord}{symbols}{"31}
\DeclareMathSymbol{\prime}{\mathord}{symbols}{"30}
\DeclareMathSymbol{\emptyset}{\mathord}{symbols}{"3B}
\DeclareMathSymbol{\nabla}{\mathord}{symbols}{"72}
\DeclareMathSymbol{\top}{\mathord}{symbols}{"3E}
\DeclareMathSymbol{\bot}{\mathord}{symbols}{"3F}
\DeclareMathSymbol{\triangle}{\mathord}{symbols}{"34}
\DeclareMathSymbol{\forall}{\mathord}{symbols}{"38}
\DeclareMathSymbol{\exists}{\mathord}{symbols}{"39}
\DeclareMathSymbol{\neg}{\mathord}{symbols}{"3A}
```

Alias:

```
%    \let\lnot=\neg
\DeclareMathSymbol{\lnot}{\mathord}{symbols}{"3A}
```

```
\DeclareMathSymbol{\flat}{\mathord}{letters}{"5B}
\DeclareMathSymbol{\natural}{\mathord}{letters}{"5C}
\DeclareMathSymbol{\sharp}{\mathord}{letters}{"5D}
\DeclareMathSymbol{\clubsuit}{\mathord}{symbols}{"7C}
\DeclareMathSymbol{\diamondsuit}{\mathord}{symbols}{"7D}
\DeclareMathSymbol{\heartsuit}{\mathord}{symbols}{"7E}
\DeclareMathSymbol{\spadesuit}{\mathord}{symbols}{"7F}
```

```
\DeclareRobustCommand\hbar{{\mathchar'26\mkern-9muh}}
\DeclareRobustCommand\surd{{\mathchar"1270}}
\DeclareRobustCommand\angle{{\vbox{\ialign{$\m@th\scriptstyle##$\crcr
      \not\mathrel{\mkern14mu}\crcr
      \noalign{\nointerlineskip}
      \mkern2.5mu\leaders\hrule \@height.34pt\hfill\mkern2.5mu\crcr}}}}
```

46.4.3 Large Operators

```
\DeclareMathSymbol{\coprod}{\mathop}{largesymbols}{"60}
\DeclareMathSymbol{\bigvee}{\mathop}{largesymbols}{"57}
\DeclareMathSymbol{\bigwedge}{\mathop}{largesymbols}{"56}
\DeclareMathSymbol{\biguplus}{\mathop}{largesymbols}{"55}
\DeclareMathSymbol{\bigcap}{\mathop}{largesymbols}{"54}
```

```
\DeclareMathSymbol{\bigcup}{\mathop}{largesymbols}{"53}
\DeclareMathSymbol{\intop}{\mathop}{largesymbols}{"52}
    \DeclareRobustCommand\int{\intop\nolimits}
\DeclareMathSymbol{\prod}{\mathop}{largesymbols}{"51}
\DeclareMathSymbol{\sum}{\mathop}{largesymbols}{"50}
\DeclareMathSymbol{\bigotimes}{\mathop}{largesymbols}{"4E}
\DeclareMathSymbol{\bigoplus}{\mathop}{largesymbols}{"4C}
\DeclareMathSymbol{\bigodot}{\mathop}{largesymbols}{"4A}
\DeclareMathSymbol{\ointop}{\mathop}{largesymbols}{"48}
    \DeclareRobustCommand\oint{\ointop\nolimits}
\DeclareMathSymbol{\bigsqcup}{\mathop}{largesymbols}{"46}
\DeclareMathSymbol{\smallint}{\mathop}{symbols}{"73}
```

46.4.4 Binary symbols

```
\DeclareMathSymbol{\triangleleft}{\mathbin}{letters}{"2F}
\DeclareMathSymbol{\triangleright}{\mathbin}{letters}{"2E}
\DeclareMathSymbol{\bigtriangleup}{\mathbin}{symbols}{"34}
\DeclareMathSymbol{\bigtriangledown}{\mathbin}{symbols}{"35}
```

Alias:

```
%   \let \varbigtriangledown \bigtriangledown
%   \let \varbigtriangleup \bigtriangleup
\DeclareMathSymbol{\varbigtriangleup}{\mathbin}{symbols}{"34}
\DeclareMathSymbol{\varbigtriangledown}{\mathbin}{symbols}{"35}
```

These last two synonyms are needed because the stmaryrd package redefines them as Operators.

```
\DeclareMathSymbol{\wedge}{\mathbin}{symbols}{"5E}
\DeclareMathSymbol{\vee}{\mathbin}{symbols}{"5F}
```

Alias:

```
%   \let\land=\wedge
%   \let\lor=\vee
\DeclareMathSymbol{\land}{\mathbin}{symbols}{"5E}
\DeclareMathSymbol{\lor}{\mathbin}{symbols}{"5F}
```

```
\DeclareMathSymbol{\cap}{\mathbin}{symbols}{"5C}
\DeclareMathSymbol{\cup}{\mathbin}{symbols}{"5B}
\DeclareMathSymbol{\ddagger}{\mathbin}{symbols}{"7A}
\DeclareMathSymbol{\dagger}{\mathbin}{symbols}{"79}
\DeclareMathSymbol{\sqcap}{\mathbin}{symbols}{"75}
\DeclareMathSymbol{\sqcup}{\mathbin}{symbols}{"74}
\DeclareMathSymbol{\uplus}{\mathbin}{symbols}{"5D}
\DeclareMathSymbol{\amalg}{\mathbin}{symbols}{"71}
\DeclareMathSymbol{\diamond}{\mathbin}{symbols}{"05}
\DeclareMathSymbol{\bullet}{\mathbin}{symbols}{"0F}
\DeclareMathSymbol{\wr}{\mathbin}{symbols}{"6F}
\DeclareMathSymbol{\div}{\mathbin}{symbols}{"04}
\DeclareMathSymbol{\odot}{\mathbin}{symbols}{"0C}
\DeclareMathSymbol{\oslash}{\mathbin}{symbols}{"0B}
\DeclareMathSymbol{\otimes}{\mathbin}{symbols}{"0A}
\DeclareMathSymbol{\ominus}{\mathbin}{symbols}{"09}
\DeclareMathSymbol{\oplus}{\mathbin}{symbols}{"08}
\DeclareMathSymbol{\mp}{\mathbin}{symbols}{"07}
\DeclareMathSymbol{\pm}{\mathbin}{symbols}{"06}
\DeclareMathSymbol{\circ}{\mathbin}{symbols}{"0E}
```

```
\DeclareMathSymbol{\bigcirc}{\mathbin}{symbols}{"0D}
\DeclareMathSymbol{\setminus}{\mathbin}{symbols}{"6E}
\DeclareMathSymbol{\cdot}{\mathbin}{symbols}{"01}
\DeclareMathSymbol{\ast}{\mathbin}{symbols}{"03}
\DeclareMathSymbol{\times}{\mathbin}{symbols}{"02}
\DeclareMathSymbol{\star}{\mathbin}{letters}{"3F}
```

46.4.5 Relations

```
\DeclareMathSymbol{\propto}{\mathrel}{symbols}{"2F}
\DeclareMathSymbol{\sqsubseteq}{\mathrel}{symbols}{"76}
\DeclareMathSymbol{\sqsupseteq}{\mathrel}{symbols}{"77}
\DeclareMathSymbol{\parallel}{\mathrel}{symbols}{"6B}
\DeclareMathSymbol{\mid}{\mathrel}{symbols}{"6A}
\DeclareMathSymbol{\dashv}{\mathrel}{symbols}{"61}
\DeclareMathSymbol{\vdash}{\mathrel}{symbols}{"60}
\DeclareMathSymbol{\nearrow}{\mathrel}{symbols}{"25}
\DeclareMathSymbol{\searrow}{\mathrel}{symbols}{"26}
\DeclareMathSymbol{\nwarrow}{\mathrel}{symbols}{"2D}
\DeclareMathSymbol{\swarrow}{\mathrel}{symbols}{"2E}
\DeclareMathSymbol{\Leftrightarrow}{\mathrel}{symbols}{"2C}
\DeclareMathSymbol{\Leftarrow}{\mathrel}{symbols}{"28}
\DeclareMathSymbol{\Rightarrow}{\mathrel}{symbols}{"29}
    \DeclareRobustCommand\neq{\not=}
```

As `\neq` is robust we should not use `\let` to define `\ne` as then it would change if `\neq` changes.

```
    \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).

```
\DeclareMathSymbol{\leq}{\mathrel}{symbols}{"14}
\DeclareMathSymbol{\geq}{\mathrel}{symbols}{"15}
```

Alias:

```
%    \let\le=\leq
%    \let\ge=\geq
\DeclareMathSymbol{\le}{\mathrel}{symbols}{"14}
\DeclareMathSymbol{\ge}{\mathrel}{symbols}{"15}
\DeclareMathSymbol{\succ}{\mathrel}{symbols}{"1F}
\DeclareMathSymbol{\prec}{\mathrel}{symbols}{"1E}
\DeclareMathSymbol{\approx}{\mathrel}{symbols}{"19}
\DeclareMathSymbol{\succeq}{\mathrel}{symbols}{"17}
\DeclareMathSymbol{\preceq}{\mathrel}{symbols}{"16}
\DeclareMathSymbol{\supset}{\mathrel}{symbols}{"1B}
\DeclareMathSymbol{\subset}{\mathrel}{symbols}{"1A}
\DeclareMathSymbol{\supseteq}{\mathrel}{symbols}{"13}
\DeclareMathSymbol{\subseteq}{\mathrel}{symbols}{"12}
\DeclareMathSymbol{\in}{\mathrel}{symbols}{"32}
\DeclareMathSymbol{\ni}{\mathrel}{symbols}{"33}
```

Alias:

```
%     \let\owns=\ni
\DeclareMathSymbol{\owns}{\mathrel}{symbols}{"33}
\DeclareMathSymbol{\gg}{\mathrel}{symbols}{"1D}
\DeclareMathSymbol{\ll}{\mathrel}{symbols}{"1C}
\DeclareMathSymbol{\not}{\mathrel}{symbols}{"36}
```

```
\DeclareMathSymbol{\leftrightarrow}{\mathrel}{symbols}{"24}
\DeclareMathSymbol{\leftarrow}{\mathrel}{symbols}{"20}
\DeclareMathSymbol{\rightarrow}{\mathrel}{symbols}{"21}
```

Alias:

```
%   \let\gets=\leftarrow
%   \let\to=\rightarrow
\DeclareMathSymbol{\gets}{\mathrel}{symbols}{"20}
\DeclareMathSymbol{\to}{\mathrel}{symbols}{"21}
```

```
\DeclareMathSymbol{\mapstochar}{\mathrel}{symbols}{"37}
   \DeclareRobustCommand\mapsto{\mapstochar\rightarrow}
\DeclareMathSymbol{\sim}{\mathrel}{symbols}{"18}
\DeclareMathSymbol{\simeq}{\mathrel}{symbols}{"27}
\DeclareMathSymbol{\perp}{\mathrel}{symbols}{"3F}
\DeclareMathSymbol{\equiv}{\mathrel}{symbols}{"11}
\DeclareMathSymbol{\asymp}{\mathrel}{symbols}{"10}
\DeclareMathSymbol{\smile}{\mathrel}{letters}{"5E}
\DeclareMathSymbol{\frown}{\mathrel}{letters}{"5F}
\DeclareMathSymbol{\leftharpoonup}{\mathrel}{letters}{"28}
\DeclareMathSymbol{\leftharpoondown}{\mathrel}{letters}{"29}
\DeclareMathSymbol{\rightharpoonup}{\mathrel}{letters}{"2A}
\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.

```
\DeclareRobustCommand
  \cong{\mathrel{\mathpalette\@vereq\sim}} % congruence sign
\def\@vereq#1#2{\lower.5\p@\vbox{\lineskiplimit\maxdimen\lineskip-.5\p@
    \ialign{$\m@th#1\hfil##\hfil$\crcr#2\crcr=\crcr}}}
\DeclareRobustCommand
  \notin{\mathrel{\m@th\mathpalette\c@ncel\in}}
\def\c@ncel#1#2{\m@th\ooalign{$\hfil#1\mkern1mu/\hfil$\crcr$#1#2$}}
\DeclareRobustCommand
  \rightleftharpoons{\mathrel{\mathpalette\rlh@{}}}
\def\rlh@#1{\vcenter{\m@th\hbox{\ooalign{\raise2pt
          \hbox{$#1\rightharpoonup$}\crcr
        $#1\leftharpoondown$}}}}
\DeclareRobustCommand
  \doteq{\buildrel\textstyle.\over=}
```

46.4.6 Arrows

```
\DeclareRobustCommand
  \joinrel{\mathrel{\mkern-3mu}}
\DeclareRobustCommand
  \relbar{\mathrel{\smash-}} % \smash, because -
                             % 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).

```
\DeclareRobustCommand
  \Relbar{\mathrel{=}}
\DeclareMathSymbol{\lhook}{\mathrel}{letters}{"2C}
   \DeclareRobustCommand\hookrightarrow{\lhook\joinrel\rightarrow}
\DeclareMathSymbol{\rhook}{\mathrel}{letters}{"2D}
   \DeclareRobustCommand\hookleftarrow{\leftarrow\joinrel\rhook}
\DeclareRobustCommand
  \bowtie{\mathrel\triangleright\joinrel\mathrel\triangleleft}
```

```
\DeclareRobustCommand
  \models{\mathrel{|}\joinrel\Relbar}
\DeclareRobustCommand
  \Longrightarrow{\Relbar\joinrel\Rightarrow}
```

LaTeX Change: `\longrightarrow` and `\longleftarrow` redefined to make then robust.

```
\DeclareRobustCommand\longrightarrow
     {\relbar\joinrel\rightarrow}
\DeclareRobustCommand\longleftarrow
     {\leftarrow\joinrel\relbar}
```

```
\DeclareRobustCommand
  \Longleftarrow{\Leftarrow\joinrel\Relbar}
\DeclareRobustCommand
  \longmapsto{\mapstochar\longrightarrow}
\DeclareRobustCommand
  \longleftrightarrow{\leftarrow\joinrel\rightarrow}
\DeclareRobustCommand
  \Longleftrightarrow{\Leftarrow\joinrel\Rightarrow}
\DeclareRobustCommand
  \iff{\;\Longleftrightarrow\;}
```

46.4.7 Punctuation symbols

```
\DeclareMathSymbol{\ldotp}{\mathpunct}{letters}{"3A}
\DeclareMathSymbol{\cdotp}{\mathpunct}{symbols}{"01}
\DeclareMathSymbol{\colon}{\mathpunct}{operators}{"3A}
```

This is commented out, since `\ldots` is now defined in ltoutenc.dtx.

```
%\def\@ldots{\mathinner{\ldotp\ldotp\ldotp}}
%\DeclareRobustCommand\ldots
%          {\relax\ifmmode\@ldots\else\mbox{$\m@th\@ldots\,$}\fi}
```

```
\DeclareRobustCommand
  \cdots{\mathinner{\cdotp\cdotp\cdotp}}
\DeclareRobustCommand
  \vdots{\vbox{\baselineskip4\p@ \lineskiplimit\z@
    \kern6\p@\hbox{.}\hbox{.}\hbox{.}}}
\DeclareRobustCommand
  \ddots{\mathinner{\mkern1mu\raise7\p@
    \vbox{\kern7\p@\hbox{.}}\mkern2mu
    \raise4\p@\hbox{.}\mkern2mu\raise\p@\hbox{.}\mkern1mu}}
```

46.4.8 Math accents

```
\DeclareMathAccent{\acute}{\mathalpha}{operators}{"13}
\DeclareMathAccent{\grave}{\mathalpha}{operators}{"12}
```

```
\DeclareMathAccent{\ddot}{\mathalpha}{operators}{"7F}
\DeclareMathAccent{\tilde}{\mathalpha}{operators}{"7E}
\DeclareMathAccent{\bar}{\mathalpha}{operators}{"16}
\DeclareMathAccent{\breve}{\mathalpha}{operators}{"15}
\DeclareMathAccent{\check}{\mathalpha}{operators}{"14}
\DeclareMathAccent{\hat}{\mathalpha}{operators}{"5E}
\DeclareMathAccent{\vec}{\mathord}{letters}{"7E}
\DeclareMathAccent{\dot}{\mathalpha}{operators}{"5F}
\DeclareMathAccent{\widetilde}{\mathord}{largesymbols}{"65}
\DeclareMathAccent{\widehat}{\mathord}{largesymbols}{"62}
```

For some reason plain TEX 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:

```
\DeclareMathAccent{\mathring}{\mathalpha}{operators}{"17}
```

46.4.9 Radicals

```
\DeclareMathRadical{\sqrtsign}{symbols}{"70}{largesymbols}{"70}
```

46.4.10 Over and under something, etc

```
\DeclareRobustCommand\overrightarrow[1]{\vbox{\m@th\ialign{##\crcr
      \rightarrowfill\crcr\noalign{\kern-\p@\nointerlineskip}
      $\hfil\displaystyle{#1}\hfil$\crcr}}}
\DeclareRobustCommand\overleftarrow[1]{\vbox{\m@th\ialign{##\crcr
      \leftarrowfill\crcr\noalign{\kern-\p@\nointerlineskip}%
      $\hfil\displaystyle{#1}\hfil$\crcr}}}
\DeclareRobustCommand\overbrace[1]
     {\mathop{\vbox{\m@th\ialign{##\crcr\noalign{\kern3\p@}%
      \downbracefill\crcr\noalign{\kern3\p@\nointerlineskip}%
      $\hfil\displaystyle{#1}\hfil$\crcr}}}\limits}
\DeclareRobustCommand\underbrace[1]{\mathop{\vtop{\m@th\ialign{##\crcr
   $\hfil\displaystyle{#1}\hfil$\crcr
   \noalign{\kern3\p@\nointerlineskip}%
   \upbracefill\crcr\noalign{\kern3\p@}}}}\limits}
```

(quite a waste of tokens, IMHO — Frank)

```
\DeclareRobustCommand\skew[3]
  {{\muskip\z@#1mu\divide\muskip\z@\tw@ \mkern\muskip\z@
    #2{\mkern-\muskip\z@{#3}\mkern\muskip\z@}\mkern-\muskip\z@}{}}
\DeclareRobustCommand\rightarrowfill{$\m@th\smash-\mkern-7mu%
  \cleaders\hbox{$\mkern-2mu\smash-\mkern-2mu$}\hfill
  \mkern-7mu\mathord\rightarrow$}
\DeclareRobustCommand\leftarrowfill{$\m@th\mathord\leftarrow\mkern-7mu%
  \cleaders\hbox{$\mkern-2mu\smash-\mkern-2mu$}\hfill
  \mkern-7mu\smash-$}
\DeclareMathSymbol{\braceld}{\mathord}{largesymbols}{"7A}
\DeclareMathSymbol{\bracerd}{\mathord}{largesymbols}{"7B}
\DeclareMathSymbol{\bracelu}{\mathord}{largesymbols}{"7C}
\DeclareMathSymbol{\braceru}{\mathord}{largesymbols}{"7D}
\DeclareRobustCommand\downbracefill{$\m@th \setbox\z@\hbox{$\braceld$}%
  \braceld\leaders\vrule \@height\ht\z@ \@depth\z@\hfill\braceru
  \bracelu\leaders\vrule \@height\ht\z@ \@depth\z@\hfill\bracerd$}
\DeclareRobustCommand\upbracefill{$\m@th \setbox\z@\hbox{$\braceld$}%
  \bracelu\leaders\vrule \@height\ht\z@ \@depth\z@\hfill\bracerd
  \braceld\leaders\vrule \@height\ht\z@ \@depth\z@\hfill\braceru$}
```

46.4.11 Delimiters

```
\DeclareMathDelimiter{\lmoustache}   % top from (, bottom from )
   {\mathopen}{largesymbols}{"7A}{largesymbols}{"40}
\DeclareMathDelimiter{\rmoustache}   % top from ), bottom from (
   {\mathclose}{largesymbols}{"7B}{largesymbols}{"41}
\DeclareMathDelimiter{\arrowvert}    % arrow without arrowheads
   {\mathord}{symbols}{"6A}{largesymbols}{"3C}
\DeclareMathDelimiter{\Arrowvert}    % double arrow without arrowheads
   {\mathord}{symbols}{"6B}{largesymbols}{"3D}
\DeclareMathDelimiter{\Vert}
   {\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

```
%\let\|=\Vert
\DeclareMathDelimiter{\|}
   {\mathord}{symbols}{"6B}{largesymbols}{"0D}
```

```
\DeclareMathDelimiter{\vert}
   {\mathord}{symbols}{"6A}{largesymbols}{"0C}
\DeclareMathDelimiter{\uparrow}
   {\mathrel}{symbols}{"22}{largesymbols}{"78}
\DeclareMathDelimiter{\downarrow}
   {\mathrel}{symbols}{"23}{largesymbols}{"79}
\DeclareMathDelimiter{\updownarrow}
   {\mathrel}{symbols}{"6C}{largesymbols}{"3F}
\DeclareMathDelimiter{\Uparrow}
   {\mathrel}{symbols}{"2A}{largesymbols}{"7E}
\DeclareMathDelimiter{\Downarrow}
   {\mathrel}{symbols}{"2B}{largesymbols}{"7F}
\DeclareMathDelimiter{\Updownarrow}
   {\mathrel}{symbols}{"6D}{largesymbols}{"77}
\DeclareMathDelimiter{\backslash}    % for double coset G\backslash H
   {\mathord}{symbols}{"6E}{largesymbols}{"0F}
\DeclareMathDelimiter{\rangle}
   {\mathclose}{symbols}{"69}{largesymbols}{"0B}
\DeclareMathDelimiter{\langle}
   {\mathopen}{symbols}{"68}{largesymbols}{"0A}
\DeclareMathDelimiter{\rbrace}
   {\mathclose}{symbols}{"67}{largesymbols}{"09}
\DeclareMathDelimiter{\lbrace}
   {\mathopen}{symbols}{"66}{largesymbols}{"08}
\DeclareMathDelimiter{\rceil}
   {\mathclose}{symbols}{"65}{largesymbols}{"07}
\DeclareMathDelimiter{\lceil}
   {\mathopen}{symbols}{"64}{largesymbols}{"06}
\DeclareMathDelimiter{\rfloor}
   {\mathclose}{symbols}{"63}{largesymbols}{"05}
\DeclareMathDelimiter{\lfloor}
   {\mathopen}{symbols}{"62}{largesymbols}{"04}
```

\lgroup
\rgroup
\bracevert

There are three plain TEX delimiters which are not fully supported by NFSS, since they partly point into a bold cmr font. Allocating a full symbol font, just to have three 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.)

```
\DeclareMathDelimiter{\lgroup} % extensible ( with sharper tips
     {\mathopen}{largesymbols}{"3A}{largesymbols}{"3A}
\DeclareMathDelimiter{\rgroup} % extensible ) with sharper tips
     {\mathclose}{largesymbols}{"3B}{largesymbols}{"3B}
\DeclareMathDelimiter{\bracevert} % the vertical bar that extends braces
     {\mathord}{largesymbols}{"3E}{largesymbols}{"3E}
```

46.5 Math versions of text commands

The \mathunderscore here is really a text definition, so it has been put back into ltoutenc.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 \mathsection \mathdollar \mathsterling \mathunderscore

These math symbols are not in plain TEX.

```
\DeclareMathSymbol{\mathparagraph}{\mathord}{symbols}{"7B}
\DeclareMathSymbol{\mathsection}{\mathord}{symbols}{"78}
\DeclareMathSymbol{\mathdollar}{\mathord}{operators}{"24}
```

```
\DeclareRobustCommand\mathsterling{\mathit{\mathchar"7024}}
\DeclareRobustCommand\mathunderscore{\kern.06em\vbox{\hrule\@width.3em}}
```

\mathellipsis

This is plain TEX's \ldots.

```
\DeclareRobustCommand\mathellipsis{\mathinner{\ldotp\ldotp\ldotp}}%
```

46.6 Other special functions and parameters

46.6.1 Bigggg

```
⟨/math⟩
⟨*math | latexrelease⟩
⟨latexrelease⟩\IncludeInRelease{2018/12/01}%
⟨latexrelease⟩                    {\Big}{Start LR-mode}%
\DeclareRobustCommand\big[1]{\leavevmode@ifvmode
   {\hbox{$\left#1\vbox to8.5\p@{}\right.\n@space$}}}
\DeclareRobustCommand\Big[1]{\leavevmode@ifvmode
   {\hbox{$\left#1\vbox to11.5\p@{}\right.\n@space$}}}
\DeclareRobustCommand\bigg[1]{\leavevmode@ifvmode
   {\hbox{$\left#1\vbox to14.5\p@{}\right.\n@space$}}}
\DeclareRobustCommand\Bigg[1]{\leavevmode@ifvmode
   {\hbox{$\left#1\vbox to17.5\p@{}\right.\n@space$}}}
⟨/math | latexrelease⟩
⟨latexrelease⟩\EndIncludeInRelease
⟨latexrelease⟩\IncludeInRelease{0000/00/00}%
⟨latexrelease⟩                    {\Big}{Start LR-mode}%
⟨latexrelease⟩\def\big#1{{\hbox{$\left#1\vbox to8.5\p@{}\right.\n@space$}}}
⟨latexrelease⟩\def\Big#1{{\hbox{$\left#1\vbox to11.5\p@{}\right.\n@space$}}}
⟨latexrelease⟩\def\bigg#1{{\hbox{$\left#1\vbox to14.5\p@{}\right.\n@space$}}}
⟨latexrelease⟩\def\Bigg#1{{\hbox{$\left#1\vbox to17.5\p@{}\right.\n@space$}}}
⟨latexrelease⟩\EndIncludeInRelease
⟨*math⟩
```

```
\def\n@space{\nulldelimiterspace\z@ \m@th}
```

46.6.2 The log-like functions

\operator@font

The \operator@font determines the symbol font used for log-like functions.

```
\def\operator@font{\mathgroup\symoperators}
```

46.6.3 Parameters

```
\thinmuskip=3mu
\medmuskip=4mu plus 2mu minus 4mu
\thickmuskip=5mu plus 5mu
```

This finishes the low-level setup in `fontmath.ltx`.

```
⟨/math⟩
```

47 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.

```
⟨*cfgtext | cfgmath | cfgprel⟩
%%
%%
%%
%% Load the standard setup:
%%
⟨+cfgtext⟩\input{fonttext.ltx}
⟨+cfgmath⟩\input{fontmath.ltx}
⟨+cfgprel⟩\input{preload.ltx}
%%
%% Small changes could go here; see documentation in cfgguide.tex for
%% allowed modifications.
%%
%% In particular it is not allowed to misuse this configuration file
%% to modify internal LaTeX commands!
%%
%% If you use this file as the basis for configuration please change
%% the \ProvidesFile lines to clearly identify your modification, e.g.,
%%
⟨+cfgtext⟩%%  \ProvidesFile{fonttext.cfg}[2001/06/01
⟨+cfgmath⟩%%  \ProvidesFile{fonttext.cfg}[2001/06/01
⟨+cfgprel⟩%%   \ProvidesFile{preload.cfg}[2001/06/01
%%                               Customised local font setup]
%%
%%
⟨/cfgtext | cfgmath | cfgprel⟩
```

File v

preload.dtx

48 Overview

This file contains an number of possible settings for preloading fonts during installation of NFSS2 (which is used by LaTeX 2ε). 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 `lfonts.tex`
preload.ltx	The standard selection of preloads
cmpreloa.xpt	preload of CM fonts for 10pt document size
cmpreloa.xip	preload of CM fonts for 11pt document size
cmpreloa.xii	preload of CM fonts for 12pt document size
dcpreloa.xpt	preload of DC fonts for 10pt size
dcpreloa.xip	preload of DC fonts for 11pt size
dcpreloa.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.

49 Customization

You can customize the preloaded fonts in your LaTeX 2ε 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* LaTeX 2ε 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 LaTeX 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.

50 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
xiipt	produce 12pt preloads
ori	produce preloads similar to old `lfonts.tex`
tex	produce preload.ltx

A typical DOCSTRIP command file would then have entries like:

```
\generateFile{preload.min}{t}{\from{preload.dtx}{preload,min}}
```

for generating preload files.

51 A driver for this document

The next bit of code contains the documentation driver file for TEX, i.e., the file that will produce the documentation you are currently reading. It will be extracted from this file by the DOCSTRIP program.

```
⟨*driver⟩
\documentclass{ltxdoc}
%\OnlyDescription  % comment out for implementation details
\begin{document}
   \DocInput{preload.dtx}
\end{document}
⟨/driver⟩
```

52 The code

We begin by loading the math extension font (cmex10) and the LATEX 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.

```
\expandafter\font\csname OMX/cmex/m/n/10\endcsname=cmex10\relax
\font\tenln  =line10   \font\tenlnw  =linew10\relax
\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.

```
⟨-tex⟩%*****************************************
```

```
⟨-tex⟩% Start any modification below this point **
⟨-tex⟩%*****************************************
⟨-tex⟩
%%
%% Computer Modern Roman:
%%----------------------
⟨*ori⟩
\DeclarePreloadSizes{OT1}{cmr}{m}{n}
        {5,6,7,8,9,10,10.95,12,14.4,17.28,20.74,24.88}
\DeclarePreloadSizes{OT1}{cmr}{bx}{n}{9,10,10.95,12,14.4,17.28}
\DeclarePreloadSizes{OT1}{cmr}{m}{sl}{10,10.95,12}
\DeclarePreloadSizes{OT1}{cmr}{m}{it}{7,8,9,10,10.95,12}
⟨/ori⟩
⟨+xpt & cm⟩ \DeclarePreloadSizes{OT1}{cmr}{m}{n}{5,7,10}
⟨+xpt & dc⟩ \DeclarePreloadSizes{T1}{cmr}{m}{n}{5,7,10}
⟨+xipt & cm⟩ \DeclarePreloadSizes{OT1}{cmr}{m}{n}{6,8,10.95}
⟨+xipt & dc⟩ \DeclarePreloadSizes{T1}{cmr}{m}{n}{6,8,10.95}
⟨+xiipt & cm⟩ \DeclarePreloadSizes{OT1}{cmr}{m}{n}{6,8,12}
⟨+xiipt & dc⟩ \DeclarePreloadSizes{T1}{cmr}{m}{n}{6,8,12}
%%
%% Computer Modern Sans:
%%---------------------
⟨+ori⟩ \DeclarePreloadSizes{OT1}{cmss}{m}{n}{10,10.95,12}
%%
%% Computer Modern Typewriter:
%%---------------------------
⟨+ori⟩ \DeclarePreloadSizes{OT1}{cmtt}{m}{n}{9,10,10.95,12}
%%
%% Computer Modern Math:
%%---------------------
⟨*ori⟩
\DeclarePreloadSizes{OML}{cmm}{m}{it}
         {5,6,7,8,9,10,10.95,12,14.4,17.28,20.74}
\DeclarePreloadSizes{OMS}{cmsy}{m}{n}
         {5,6,7,8,9,10,10.95,12,14.4,17.28,20.74}
⟨/ori⟩
```

The math fonts are the same for both DC and CM fonts. So far there isn't an agreed on standard.

```
⟨*xpt⟩
\DeclarePreloadSizes{OML}{cmm}{m}{it}{5,7,10}
\DeclarePreloadSizes{OMS}{cmsy}{m}{n}{5,7,10}
⟨/xpt⟩
⟨*xipt⟩
\DeclarePreloadSizes{OML}{cmm}{m}{it}{6,8,10.95}
\DeclarePreloadSizes{OMS}{cmsy}{m}{n}{6,8,10.95}
⟨/xipt⟩
⟨*xiipt⟩
\DeclarePreloadSizes{OML}{cmm}{m}{it}{6,8,12}
\DeclarePreloadSizes{OMS}{cmsy}{m}{n}{6,8,12}
⟨/xiipt⟩
%%
%% LaTeX symbol fonts:
%%-------------------
```

```
⟨*ori⟩
\DeclarePreloadSizes{U}{lasy}{m}{n}
          {5,6,7,8,9,10,10.95,12,14.4,17.28,20.74}
⟨/ori⟩
⟨/preload⟩
```

File w

ltfntcmd.dtx

Abstract

The commands defined in this file `ltfntcmd` are part of the kernel code for LaTeX 2ε/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.

53 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 LaTeX3 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`.[9] The default definition for this list is produced by the following.

```
\newcommand \nocorrlist {,.}
```

[9] Any 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.

Command	*Corresponds to*	*Action*
`\textnormal{..}`	`\normalfont`	Typeset argument in normal family
`\textrm{..}`	`\rmfamily`	Typeset argument in roman family
`\textsf{..}`	`\sffamily`	Typeset argument in sans serif family
`\texttt{..}`	`\ttfamily`	Typeset argument in `typewriter` family
`\textmd{..}`	`\mdseries`	Typeset argument in medium series
`\textbf{..}`	`\bfseries`	Typeset argument in **bold** series
`\textup{..}`	`\upshape`	Typeset argument in normal shape
`\textit{..}`	`\itshape`	Typeset argument in *italic* shape
`\textsl{..}`	`\slshape`	Typeset argument in *slanted* shape
`\textsc{..}`	`\scshape`	Typeset argument in SMALL CAPS shape
`\emph{..}`	`\em`	Typeset argument *emphasized*

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.

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
\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 LaTeX 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 LaTeX'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 `\/`.

54 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 optimises 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.

```
⟨*2ekernel⟩
\def \DeclareTextFontCommand #1#2{%
  \DeclareRobustCommand#1[1]{%
    \ifmmode
      \nfss@text{#2##1}%
    \else
      \hmode@bgroup
       \text@command{##1}%
       #2\check@icl ##1\check@icr
       \expandafter
      \egroup
    \fi
                       }%
}
```

\textrm \textsf \texttt \textnormal

Now we define the `\text`⟨*family*⟩ commands in terms of the above; `\texttt` does not look very nice!

```
\DeclareTextFontCommand{\textrm}{\rmfamily}
\DeclareTextFontCommand{\textsf}{\sffamily}
\DeclareTextFontCommand{\texttt}{\ttfamily}
\DeclareTextFontCommand{\textnormal}{\normalfont}
```

\textbf \textmd

For the series attribute:

```
\DeclareTextFontCommand{\textbf}{\bfseries}
\DeclareTextFontCommand{\textmd}{\mdseries}
```

\textit \textsl \textsc \textup

And for the shapes:

```
\DeclareTextFontCommand{\textit}{\itshape}
\DeclareTextFontCommand{\textsl}{\slshape}
\DeclareTextFontCommand{\textsc}{\scshape}
\DeclareTextFontCommand{\textup}{\upshape}
```

textulc textsw textssc

```
⟨/2ekernel⟩
⟨*2ekernel | latexrelease⟩
⟨latexrelease⟩\IncludeInRelease{2020/02/02}%
⟨latexrelease⟩                 {\textulc}{Additional text commands}%
\DeclareTextFontCommand{\textulc}{\ulcshape}
\DeclareTextFontCommand{\textsw}{\swshape}
\DeclareTextFontCommand{\textssc}{\sscshape}
⟨/2ekernel | latexrelease⟩
⟨latexrelease⟩\EndIncludeInRelease
⟨latexrelease⟩\IncludeInRelease{0000/00/00}%
⟨latexrelease⟩                 {\textulc}{Additional text commands}%
⟨latexrelease⟩
⟨latexrelease⟩\let\textulc\@undefined
⟨latexrelease⟩\let\textsw\@undefined
⟨latexrelease⟩\let\textssc\@undefined
⟨latexrelease⟩\EndIncludeInRelease
⟨*2ekernel⟩
```

\emph

Finally we have the `\em` font change declaration of LaTeX. The corresponding definition with argument is

```
\DeclareTextFontCommand{\emph}{\em}
```

\nocorr

This is just a label, so it does nothing; it should also be unexpandable.

```
\let \nocorr \relax
```

\check@icl
\check@icr

We define these defaults in case some error causes them to be expanded at the wrong time.

```
\let \check@icl \@empty
\let \check@icr \@empty
```

\text@command
\check@nocorr@

This checks for a `\nocorr` as the first token in its argument and also for one in any other 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!

```
\def \text@command #1{%
  \def \reserved@a {#1}%
  \ifx \reserved@a \@empty
    \let \check@icl \@empty
    \let \check@icr \@empty
  \else
```

`\space` is a reserved word in LaTeX or actually already in plain TeX. 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.

```
%    \def \reserved@b { }%
%    \ifx \reserved@a \reserved@b
    \ifx \reserved@a \space
      \let \check@icl \@empty
      \let \check@icr \@empty
    \else
      \check@nocorr@ #1\nocorr\@nil
    \fi
  \fi
}
\def \check@nocorr@ #1#2\nocorr#3\@nil {%
```

The two checks are initialised here to their values in the normal case.

```
  \let \check@icl \maybe@ic
  \def \check@icr {\ifvmode \else \aftergroup \maybe@ic \fi}%
  \def \reserved@a {\nocorr}%
  \def \reserved@b {#1}%
  \def \reserved@c {#3}%
  \ifx \reserved@a \reserved@b
    \ifx \reserved@c \@empty
```

In this case there is a `\nocorr` at the start but not at the end, so `\check@icl` should be empty.

```
      \let \check@icl \@empty
    \else
```

Otherwise there is a `\nocorr` both at the start and elsewhere, so no italic corrections should be added.

```
      \let \check@icl \@empty
      \let \check@icr \@empty
    \fi
  \else
    \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.

```
    \else
```

In this case there is no `\nocorr` at the start but there is one elsewhere, so no `\aftergroup` is needed.

```
      \let \check@icr \@empty
    \fi
  \fi
}
```

`\ifmaybe@ic` Switch used soley within `\maybe@ic` not interfering with other switches.

```
\newif\ifmaybe@ic
```

`\maybe@ic` `\maybe@ic@` These macros implement the italic correction.

```
\def \maybe@ic {\futurelet\@let@token\maybe@ic@}
\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?

```
  \ifdim  \fontdimen\@ne\font>\z@
  \else
    \maybe@ictrue
```

It would be possible, but probably not worthwhile, to continue the forward scan beyond any closing braces.

```
    \expandafter\@tfor\expandafter\reserved@a\expandafter:\expandafter=%
        \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.

```
    \do \t@st@ic
```

Frank thinks that the next bit it 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`.

```
    \ifmaybe@ic \sw@slant \fi
  \fi
}
```

`\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.

```
\def \t@st@ic {%
  \expandafter\let\expandafter\reserved@b\expandafter=\reserved@a\relax
  \ifx\reserved@b\@let@token
```

If they are the same we record the fact and jump out of the loop.

```
    \maybe@icfalse
    \@break@tfor
  \fi
}
```

\sw@slant \fix@penalty The definition of the mysterious \sw@slant command is as follows.

```
\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.

```
  \ifdim \lastskip=\z@
    \fix@penalty
  \else
    \skip@ \lastskip
    \unskip
    \fix@penalty
    \hskip \skip@
  \fi
}
```

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 LaTeX code?

```
\def \fix@penalty {%
  \ifnum \lastpenalty=\z@
    \@@italiccorr
  \else
    \count@ \lastpenalty
    \unpenalty
    \@@italiccorr
    \penalty \count@
  \fi
}
```

\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.

```
\def \nocorrlist {,.}
```

`\nfss@text` This command will by default behave like a LaTeX `\mbox` but may be redefined by packages such as `amstext.sty` to be a bit cleverer.

```
\ifx \nfss@text\@undefined
  \def \nfss@text {\leavevmode\hbox}
\fi
```

`\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}
```

```
\def \DeclareOldFontCommand #1#2#3{%
  \DeclareRobustCommand #1{\@fontswitch {#2}{#3}}%
}
```

`\@fontswitch`
`\@@math@egroup`
`\@@math@egroup`

These two commands actually do the necessary tests and declarative font- or alphabet-changing.

```
\def \@fontswitch #1#2{%
  \ifmmode
    \let \math@bgroup \relax
    \def \math@egroup {\let \math@bgroup \@@math@bgroup
                        \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).

```
    #2\relax
  \else
    #1%
  \fi
}
\let \@@math@bgroup \math@bgroup
\let \@@math@egroup \math@egroup
```

These commands are available only in the preamble.

```
\@onlypreamble \DeclareTextFontCommand
\@onlypreamble \DeclareOldFontCommand
```

55 Initialization

`\normalsize` This is defined to produce an error.

```
\def\normalsize{%
  \@latex@error {The font size command \protect\normalsize\space
                 is not defined:\MessageBreak
                 there is probably something wrong with
                 the class file}\@eha
}
⟨/2ekernel⟩
```

File x

lttextcomp.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).

\oldstylenums
\legacyoldstylenums

Preserve the old definition of \oldstylenums under a different name.

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.

```
⟨*2ekernel | latexrelease⟩
⟨latexrelease⟩\IncludeInRelease{2020/02/02}%
⟨latexrelease⟩                 {\oldstylenums}{Old style numerals}%
\DeclareRobustCommand\legacyoldstylenums[1]{%
   \begingroup
```

Provide spacing using the interword space of the current font.

```
    \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 LaTeX fonts even though they only contain substitutions).

```
    \usefont{OML}{\rmdefault}{\f@series}{it}%
    \mathgroup\symletters #1%
   \endgroup
}
```

And here is the improved one that adjusts depending on surroundings.

```
\DeclareRobustCommand\oldstylenums[1]{%
 \begingroup
 \ifmmode
   \mathgroup\symletters #1%
  \else
```

The \CheckEncodingSubset is discused below.

```
   \CheckEncodingSubset\@use@text@encoding{TS1}\tc@oldstylesubst2{{#1}}%
 \fi
 \endgroup
}
```

The helper to select the substitution if needed.

```
\def\tc@oldstylesubst#1{%
  \tc@errorwarn
          {Oldstyle digits unavailable for
           family \f@family.\MessageBreak
           Default oldstyle digits used instead}\@eha
  \bgroup
      \expand@font@defaults
```

The substitution defaults are provided in the file `fonttext.ltx`.

```
      \ifx\f@family\rmdef@ult
         \fontfamily\rmsubstdefault
      \else\ifx\f@family\sfdef@ult
         \fontfamily\sfsubstdefault
      \else\ifx\f@family\ttdef@ult
         \fontfamily\ttsubstdefault
      \else
        \fontfamily\textcompsubstdefault
      \fi\fi\fi
      \fontencoding{TS1}\selectfont#1%
  \egroup
}
```

`\textcompsubstdefault` Here is the default for the "unknown" case:

```
\def\textcompsubstdefault{\rmsubstdefault}
```

```
⟨/2ekernel | latexrelease⟩
⟨latexrelease⟩\EndIncludeInRelease
⟨latexrelease⟩\IncludeInRelease{0000/00/00}%
⟨latexrelease⟩                 {\oldstylenums}{Old style numerals}%
⟨latexrelease⟩
⟨latexrelease⟩\DeclareRobustCommand\oldstylenums[1]{%
⟨latexrelease⟩    \begingroup
⟨latexrelease⟩     \spaceskip\fontdimen\tw@\font
⟨latexrelease⟩     \usefont{OML}{\rmdefault}{\f@series}{it}%
⟨latexrelease⟩     \mathgroup\symletters #1%
⟨latexrelease⟩    \endgroup
⟨latexrelease⟩}
⟨latexrelease⟩\let\legacyoldstylenums\@undefined
⟨latexrelease⟩\def\textcompsubstdefault{cmr}
⟨latexrelease⟩
⟨latexrelease⟩\EndIncludeInRelease
```

Everything else in the this file got introduced 2020/02/02, so we do a single rollback (for now).

```
⟨*2ekernel⟩
⟨/2ekernel⟩
⟨*2ekernel | latexrelease⟩
⟨latexrelease⟩\IncludeInRelease{2020/02/02}%
⟨latexrelease⟩    {\DeclareEncodingSubset}{Text companion symbols}%
```

`\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 choosen 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.

```
\def\DeclareEncodingSubset#1#2{%
  \DeclareEncodingSubset@aux{#1}#2*\DeclareEncodingSubset@aux
}
```

```
\def\DeclareEncodingSubset@aux#1#2*#3\DeclareEncodingSubset@aux#4{%
```

if `#3` is empty then there was no star, otherwise we define all four variants.

```
  \expandafter\ifx\expandafter X\detokenize{#3}X%
    \@DeclareEncodingSubset{#1}{#2}{#4}%
  \else
    \@DeclareEncodingSubset{#1}{#2LF}{#4}%
    \@DeclareEncodingSubset{#1}{#2TLF}{#4}%
    \@DeclareEncodingSubset{#1}{#2OsF}{#4}%
    \@DeclareEncodingSubset{#1}{#2TOsF}{#4}%
  \fi
}
```

The subset info is stored in a command with the name \family:*subset* so if that already exists we change otherwise declare a subset.

```
\def\@DeclareEncodingSubset#1#2#3{%
   \@ifundefined{#1:#2}%
     {\@font@info{Setting #2 sub-encoding to #1/#3}}%
     {\@font@info{Changing #2 sub-encoding to #1/#3}}%
     \@namedef{#1:#2}{#3}}
```

Any reason to allow those in the middle of documents?

```
\@onlypreamble\DeclareEncodingSubset
\@onlypreamble\DeclareEncodingSubset@aux
\@onlypreamble\@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.

```
\def\CheckEncodingSubset#1#2#3#4#5{%
    \ifnum #4>%
      \expandafter\ifx\csname #2:\f@family\endcsname\relax
        0\csname #2:?\endcsname
      \else
        \csname #2:\f@family\endcsname
      \fi
   \relax
   \expandafter\@firstoftwo
  \else
   \expandafter\@secondoftwo
 \fi
  {#1{#2}}{#3}%
  #5%
}
```

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.

```
\def\tc@errorwarn#1#2{\@latex@info{#1}}
```

\tc@subst

```
\def\tc@subst#1{%
   \tc@errorwarn
    {Symbol \string#1 not provided by\MessageBreak
     font family \f@family\space
     in TS1 encoding.\MessageBreak Default family used instead}\@eha
  \bgroup
      \expand@font@defaults
      \ifx\f@family\rmdef@ult
         \fontfamily\rmsubstdefault
      \else\ifx\f@family\sfdef@ult
         \fontfamily\sfsubstdefault
      \else\ifx\f@family\ttdef@ult
         \fontfamily\ttsubstdefault
      \else
         \fontfamily\textcompsubstdefault
      \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.

```
      \@namedef{TS1:\f@family}{0}%
      \selectfont#1%
  \egroup
}
```

\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 a poor man's Euro symbol by combining a "C" with a "=".

```
\def\tc@fake@euro#1{%
   \leavevmode
   \@font@info{Faking \noexpand#1for font family
                          \f@family\MessageBreak in TS1 encoding}%
   \valign{##\cr
      \vfil\hbox to 0.07em{\dimen@\f@size\p@
                           \math@fontsfalse
                           \fontsize{.7\dimen@}\z@\selectfont=\hss}%
      \vfil\cr%
      \hbox{C}\crcr
   }%
}
```

\tc@check@symbol \tc@check@accent These are two abbreviations that we use below to check symbols and accents in TS1. 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 `\textcurreny` is not available.

```
\def\tc@check@symbol{\CheckEncodingSubset\UseTextSymbol{TS1}\tc@subst}
```

Accents and been mad an error in the `textcomp` package when not available. Now that we provide the functionality in the kernel we avoid the eror by swapping in a `T1` accent if the `TS1` accent is not available.

```
%\def\tc@check@accent{\CheckEncodingSubset\UseTextAccent{TS1}\tc@error}
```

```
\def\tc@check@accent#1{\CheckEncodingSubset\UseTextAccent{TS1}{\tc@swap@accent#1}}
\def\tc@swap@accent#1#2{\UseTextAccent{T1}#1}
```

56 Sub-encodings

Here are the default definitions for the `TS1` symbols. First those that we assume are always available if a font implements `TS1`.

```
\DeclareTextSymbolDefault{\textdollar}{TS1}
\UndeclareTextCommand{\textdollar}  {OT1}         % don't use the OT1 def any longer

\DeclareTextSymbolDefault{\textsterling}{TS1}
\UndeclareTextCommand{\textsterling}{OT1}         % don't use the OT1 def any longer

\DeclareTextSymbolDefault{\textperthousand}{TS1}
\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 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`:

```
\DeclareTextSymbolDefault{\textasteriskcentered}{TS1}
\DeclareTextSymbolDefault{\textbullet}{TS1}
\DeclareTextSymbolDefault{\textdaggerdbl}{TS1}
```

```
\DeclareTextSymbolDefault{\textdagger}{TS1}
\DeclareTextSymbolDefault{\textparagraph}{TS1}
\DeclareTextSymbolDefault{\textperiodcentered}{TS1}
\DeclareTextSymbolDefault{\textsection}{TS1}
```

And here are the other TS1 glyphs that are implemented by every font (or nearly everyone—a few are commented out and moved to sub-encoding 9, because they aren't around in one or two fonts.

```
%%\DeclareTextSymbolDefault{\textbardbl}{TS1} % subst in sub-enc 9 above
\DeclareTextSymbolDefault{\textbrokenbar}{TS1}
%%\DeclareTextSymbolDefault{\textcelsius}{TS1} % subst in sub-enc 9 above
\DeclareTextSymbolDefault{\textcent}{TS1}
\DeclareTextSymbolDefault{\textcopyright}{TS1}
\DeclareTextSymbolDefault{\textdegree}{TS1}
\DeclareTextSymbolDefault{\textdiv}{TS1}
\DeclareTextSymbolDefault{\textlnot}{TS1}
\DeclareTextSymbolDefault{\textonehalf}{TS1}
\DeclareTextSymbolDefault{\textonequarter}{TS1}
%%\DeclareTextSymbolDefault{\textonesuperior}{TS1} % subst in sub-enc 9 above
\DeclareTextSymbolDefault{\textordfeminine}{TS1}
\DeclareTextSymbolDefault{\textordmasculine}{TS1}
\DeclareTextSymbolDefault{\textpm}{TS1}
\DeclareTextSymbolDefault{\textquotesingle}{TS1}
\DeclareTextSymbolDefault{\textquotestraightbase}{TS1}
\DeclareTextSymbolDefault{\textquotestraightdblbase}{TS1}
\DeclareTextSymbolDefault{\textregistered}{TS1}
%%\DeclareTextSymbolDefault{\textthreequartersemdash}{TS1} % subst in sub-enc 9 above
\DeclareTextSymbolDefault{\textthreequarters}{TS1}
%%\DeclareTextSymbolDefault{\textthreesuperior}{TS1} % subst in sub-enc 9 above
\DeclareTextSymbolDefault{\texttimes}{TS1}
\DeclareTextSymbolDefault{\texttrademark}{TS1}
%%\DeclareTextSymbolDefault{\texttwelveudash}{TS1} % subst in sub-enc 9 above
%%\DeclareTextSymbolDefault{\texttwosuperior}{TS1} % subst in sub-enc 9 above
\DeclareTextSymbolDefault{\textyen}{TS1}
```

```
\DeclareTextSymbolDefault{\textcapitalcompwordmark}{TS1}
\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.

56.1 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.

```
\DeclareTextCommandDefault{\textcircled}
  {\CheckEncodingSubset\UseTextAccent{TS1}{\UseTextAccent{OMS}}1\textcircled}
```

56.2 Sub-encoding 2 (majority of new OTF fonts via autoinst)

```
\DeclareTextCommandDefault{\t}
  {\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.

```
\ifx\Umathcode\@undefined
  \DeclareTextCommandDefault{\capitalacute}       {\tc@check@accent{\'}2\capitalacute}
  \DeclareTextCommandDefault{\capitalbreve}       {\tc@check@accent{\u}2\capitalbreve}
  \DeclareTextCommandDefault{\capitalcaron}       {\tc@check@accent{\v}2\capitalcaron}
  \DeclareTextCommandDefault{\capitalcedilla}     {\tc@check@accent{\c}2\capitalcedilla}
  \DeclareTextCommandDefault{\capitalcircumflex}  {\tc@check@accent{\^}2\capitalcircumflex}
  \DeclareTextCommandDefault{\capitaldieresis}    {\tc@check@accent{\"}2\capitaldieresis}
  \DeclareTextCommandDefault{\capitaldotaccent}   {\tc@check@accent{\.}2\capitaldotaccent}
  \DeclareTextCommandDefault{\capitalgrave}       {\tc@check@accent{\`}2\capitalgrave}
  \DeclareTextCommandDefault{\capitalhungarumlaut}{\tc@check@accent{\H}2\capitalhungarumlaut}
  \DeclareTextCommandDefault{\capitalmacron}      {\tc@check@accent{\=}2\capitalmacron}
  \DeclareTextCommandDefault{\capitalogonek}      {\tc@check@accent{\k}2\capitalogonek}
  \DeclareTextCommandDefault{\capitalring}        {\tc@check@accent{\r}2\capitalring}
  \DeclareTextCommandDefault{\capitaltie}         {\tc@check@accent{\t}2\capitaltie}
  \DeclareTextCommandDefault{\capitaltilde}       {\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).

```
  \DeclareTextCommandDefault{\newtie}             {\tc@check@accent{\t}2\newtie}
  \DeclareTextCommandDefault{\capitalnewtie}      {\tc@check@accent{\t}2\capitalnewtie}
```

In Unicode engines we just execute the simple accents:

```
\else
  \def\capitalacute{\@tabacckludge'}
  \def\capitalbreve{\u}
  \def\capitalcaron{\v}
  \def\capitalcedilla{\c}
  \def\capitalcircumflex{\^}
  \def\capitaldieresis{\"}
  \def\capitaldotaccent{\.}
  \def\capitalgrave{\@tabacckludge`}
  \def\capitalhungarumlaut{\H}
  \def\capitalmacron{\@tabacckludge=}
  \def\capitalnewtie{\t}
  \def\capitalogonek{\k}
  \def\capitalring{\r}
  \def\capitaltie{\t}
  \def\capitaltilde{\~}
  \def\newtie{\t}
\fi
```

The next two symbols exist in some fonts (faked?), but we ignore that to keep the subsets reasonable compact and most important linear.

```
\DeclareTextCommandDefault{\textlbrackdbl}      {\tc@check@symbol2\textlbrackdbl}
\DeclareTextCommandDefault{\textrbrackdbl}      {\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.

```
\DeclareTextCommandDefault{\texteightoldstyle}  {\tc@check@symbol2\texteightoldstyle}
\DeclareTextCommandDefault{\textfiveoldstyle}   {\tc@check@symbol2\textfiveoldstyle}
\DeclareTextCommandDefault{\textfouroldstyle}   {\tc@check@symbol2\textfouroldstyle}
\DeclareTextCommandDefault{\textnineoldstyle}   {\tc@check@symbol2\textnineoldstyle}
\DeclareTextCommandDefault{\textoneoldstyle}    {\tc@check@symbol2\textoneoldstyle}
\DeclareTextCommandDefault{\textsevenoldstyle}  {\tc@check@symbol2\textsevenoldstyle}
\DeclareTextCommandDefault{\textsixoldstyle}    {\tc@check@symbol2\textsixoldstyle}
\DeclareTextCommandDefault{\textthreeoldstyle}  {\tc@check@symbol2\textthreeoldstyle}
\DeclareTextCommandDefault{\texttwooldstyle}    {\tc@check@symbol2\texttwooldstyle}
\DeclareTextCommandDefault{\textzerooldstyle}   {\tc@check@symbol2\textzerooldstyle}
```

The next set of glyphs is special to TeX fonts (and available with a few older PS fonts supported in the virtual fonts), but not any longer in the majority of fonts provided through autoinst, so we pretend there aren't available in sub-encoding 2 and below.

```
\DeclareTextCommandDefault{\textacutedbl}       {\tc@check@symbol2\textacutedbl}
\DeclareTextCommandDefault{\textasciiacute}     {\tc@check@symbol2\textasciiacute}
\DeclareTextCommandDefault{\textasciibreve}     {\tc@check@symbol2\textasciibreve}
\DeclareTextCommandDefault{\textasciicaron}     {\tc@check@symbol2\textasciicaron}
\DeclareTextCommandDefault{\textasciidieresis}  {\tc@check@symbol2\textasciidieresis}
\DeclareTextCommandDefault{\textasciigrave}     {\tc@check@symbol2\textasciigrave}
\DeclareTextCommandDefault{\textasciimacron}    {\tc@check@symbol2\textasciimacron}
\DeclareTextCommandDefault{\textgravedbl}       {\tc@check@symbol2\textgravedbl}
\DeclareTextCommandDefault{\texttildelow}       {\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 TeX world.

```
\DeclareTextCommandDefault{\textbaht}           {\tc@check@symbol2\textbaht}
\DeclareTextCommandDefault{\textbigcircle}      {\tc@check@symbol2\textbigcircle}
\DeclareTextCommandDefault{\textborn}           {\tc@check@symbol2\textborn}
\DeclareTextCommandDefault{\textcentoldstyle}   {\tc@check@symbol2\textcentoldstyle}
\DeclareTextCommandDefault{\textcircledP}       {\tc@check@symbol2\textcircledP}
\DeclareTextCommandDefault{\textcopyleft}       {\tc@check@symbol2\textcopyleft}
\DeclareTextCommandDefault{\textdblhyphenchar}  {\tc@check@symbol2\textdblhyphenchar}
\DeclareTextCommandDefault{\textdblhyphen}      {\tc@check@symbol2\textdblhyphen}
\DeclareTextCommandDefault{\textdied}           {\tc@check@symbol2\textdied}
\DeclareTextCommandDefault{\textdiscount}       {\tc@check@symbol2\textdiscount}
\DeclareTextCommandDefault{\textdivorced}       {\tc@check@symbol2\textdivorced}
\DeclareTextCommandDefault{\textdollaroldstyle} {\tc@check@symbol2\textdollaroldstyle}
\DeclareTextCommandDefault{\textguarani}        {\tc@check@symbol2\textguarani}
\DeclareTextCommandDefault{\textleaf}           {\tc@check@symbol2\textleaf}
\DeclareTextCommandDefault{\textlquill}         {\tc@check@symbol2\textlquill}
\DeclareTextCommandDefault{\textmarried}        {\tc@check@symbol2\textmarried}
\DeclareTextCommandDefault{\textmho}            {\tc@check@symbol2\textmho}
\DeclareTextCommandDefault{\textmusicalnote}    {\tc@check@symbol2\textmusicalnote}
\DeclareTextCommandDefault{\textnaira}          {\tc@check@symbol2\textnaira}
\DeclareTextCommandDefault{\textopenbullet}     {\tc@check@symbol2\textopenbullet}
\DeclareTextCommandDefault{\textpeso}           {\tc@check@symbol2\textpeso}
\DeclareTextCommandDefault{\textpilcrow}        {\tc@check@symbol2\textpilcrow}
\DeclareTextCommandDefault{\textrecipe}         {\tc@check@symbol2\textrecipe}
\DeclareTextCommandDefault{\textreferencemark}  {\tc@check@symbol2\textreferencemark}
\DeclareTextCommandDefault{\textrquill}         {\tc@check@symbol2\textrquill}
```

```
\DeclareTextCommandDefault{\textservicemark}    {\tc@check@symbol2\textservicemark}
\DeclareTextCommandDefault{\textsurd}           {\tc@check@symbol2\textsurd}
```

The `\textpertenthousand` also belongs in this group but here we have a choice: in T1 there is 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.

```
\DeclareTextCommandDefault{\textpertenthousand} {\tc@check@symbol2\textpertenthousand}
\UndeclareTextCommand{\textpertenthousand}{T1}
```

56.3 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.

```
\DeclareTextCommandDefault{\textlangle}         {\tc@check@symbol3\textlangle}
\DeclareTextCommandDefault{\textrangle}         {\tc@check@symbol3\textrangle}
```

56.4 Sub-encoding 4

```
\DeclareTextCommandDefault{\textcolonmonetary}  {\tc@check@symbol4\textcolonmonetary}
\DeclareTextCommandDefault{\textdong}           {\tc@check@symbol4\textdong}
\DeclareTextCommandDefault{\textdownarrow}      {\tc@check@symbol4\textdownarrow}
\DeclareTextCommandDefault{\textleftarrow}      {\tc@check@symbol4\textleftarrow}
\DeclareTextCommandDefault{\textlira}           {\tc@check@symbol4\textlira}
\DeclareTextCommandDefault{\textrightarrow}     {\tc@check@symbol4\textrightarrow}
\DeclareTextCommandDefault{\textuparrow}        {\tc@check@symbol4\textuparrow}
\DeclareTextCommandDefault{\textwon}            {\tc@check@symbol4\textwon}
```

56.5 Sub-encoding 5 (most older PS fonts)

Most older PS fonts (supported in TEX 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 glpyhs 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 wouput put them all into sub-encoding 9 so that's a compromise really.

Modern fonts that don't typeset a tofu character if a glyph is missing are only cataloged as sub-encoding 5 if they really support of its glyph set.

```
\DeclareTextCommandDefault{\textestimated}      {\tc@check@symbol5\textestimated}
\DeclareTextCommandDefault{\textnumero}         {\tc@check@symbol5\textnumero}
```

56.6 Sub-encoding 6

```
\DeclareTextCommandDefault{\textflorin}         {\tc@check@symbol6\textflorin}
\DeclareTextCommandDefault{\textcurrency}       {\tc@check@symbol6\textcurrency}
```

56.7 Sub-encoding 7

```
\DeclareTextCommandDefault{\textfractionsolidus}{\tc@check@symbol7\textfractionsolidus}
\DeclareTextCommandDefault{\textohm}            {\tc@check@symbol7\textohm}
```

```
\DeclareTextCommandDefault{\textmu}             {\tc@check@symbol7\textmu}
\DeclareTextCommandDefault{\textminus}          {\tc@check@symbol7\textminus}
```

56.8 Sub-encoding 8

```
\DeclareTextCommandDefault{\textblank}          {\tc@check@symbol{8}\textblank}
\DeclareTextCommandDefault{\textinterrobangdown}{\tc@check@symbol{8}\textinterrobangdown}
\DeclareTextCommandDefault{\textinterrobang}    {\tc@check@symbol{8}\textinterrobang}
```

Fonts with this sub-encoding don't have a Euro symbol, but instead of substituting we fake it.

```
\DeclareTextCommandDefault{\texteuro}
            {\CheckEncodingSubset\UseTextSymbol{TS1}\tc@fake@euro{8}\texteuro}
```

56.9 Sub-encoding 9 (most missing)

```
\DeclareTextCommandDefault{\textcelsius}{\tc@check@symbol{9}\textcelsius}
\DeclareTextCommandDefault{\textonesuperior}{\tc@check@symbol{9}\textonesuperior}
\DeclareTextCommandDefault{\textthreequartersemdash}{\tc@check@symbol{9}\textthreequartersemd
\DeclareTextCommandDefault{\textthreesuperior}{\tc@check@symbol{9}\textthreesuperior}
\DeclareTextCommandDefault{\texttwelveudash}{\tc@check@symbol{9}\texttwelveudash}
\DeclareTextCommandDefault{\texttwosuperior}{\tc@check@symbol{9}\texttwosuperior}
\DeclareTextCommandDefault{\textbardbl}{\tc@check@symbol{9}\textbardbl}
```

57 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).

```
\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.

```
%\DeclareTextSymbol{\textcopyleft}{TS1}{171}
%\DeclareTextSymbol{\textdblhyphen}{TS1}{45}
%\DeclareTextSymbol{\textdblhyphenchar}{TS1}{127}
%\DeclareTextSymbol{\textquotestraightbase}{TS1}{13}
%\DeclareTextSymbol{\textquotestraightdblbase}{TS1}{18}
%\DeclareTextSymbol{\textleaf}{TS1}{108}
%\DeclareTextSymbol{\texttwelveudash}{TS1}{21}
%\DeclareTextSymbol{\textthreequartersemdash}{TS1}{22}
```

If oldstyle numerals are asked for we just use `\oldstylenums`.

```
\DeclareTextCommand{\textzerooldstyle} \UnicodeEncodingName{\oldstylenums{0}}
\DeclareTextCommand{\textoneoldstyle}  \UnicodeEncodingName{\oldstylenums{1}}
\DeclareTextCommand{\texttwooldstyle}  \UnicodeEncodingName{\oldstylenums{2}}
\DeclareTextCommand{\textthreeoldstyle}\UnicodeEncodingName{\oldstylenums{3}}
\DeclareTextCommand{\textfouroldstyle} \UnicodeEncodingName{\oldstylenums{4}}
\DeclareTextCommand{\textfiveoldstyle} \UnicodeEncodingName{\oldstylenums{5}}
\DeclareTextCommand{\textsixoldstyle}  \UnicodeEncodingName{\oldstylenums{6}}
\DeclareTextCommand{\textsevenoldstyle}\UnicodeEncodingName{\oldstylenums{7}}
\DeclareTextCommand{\texteightoldstyle}\UnicodeEncodingName{\oldstylenums{8}}
\DeclareTextCommand{\textnineoldstyle} \UnicodeEncodingName{\oldstylenums{9}}
```

These have Unicode slots so this should be integrated into TU explictly

```
\DeclareTextSymbol{\textpilcrow}       \UnicodeEncodingName{"00B6}
\DeclareTextSymbol{\textborn}          \UnicodeEncodingName{"002A}
```

```
\DeclareTextSymbol{\textdied}          \UnicodeEncodingName{"2020}
\DeclareTextSymbol{\textlbrackdbl}     \UnicodeEncodingName{"27E6}
\DeclareTextSymbol{\textrbrackdbl}     \UnicodeEncodingName{"27E7}
\DeclareTextSymbol{\textguarani}       \UnicodeEncodingName{"20B2}
```

We could make `\textcentoldstyle` and `\textdollaroldstyle` point to dollar and cent in the Unicode encoding

```
%\DeclareTextSymbol{\textcentoldstyle}            \UnicodeEncodingName{"00A2}
%\DeclareTextSymbol{\textdollaroldstyle}          \UnicodeEncodingName{"0024}
```

but I think it is better to pick them up from TS1 even if that usually means LMR fonts

```
\DeclareTextSymbol{\textdollaroldstyle}{TS1}{138}
\DeclareTextSymbol{\textcentoldstyle}  {TS1}{139}
```

```
\fi               % --- END of Unicode engines specials
```

58 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 TeX 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.

```
\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.

```
\DeclareEncodingSubset{TS1}{ccr}     {0}
\DeclareEncodingSubset{TS1}{cmbr}    {0}
\DeclareEncodingSubset{TS1}{cmr}     {0}
\DeclareEncodingSubset{TS1}{cmss}    {0}
\DeclareEncodingSubset{TS1}{cmtl}    {0}
\DeclareEncodingSubset{TS1}{cmtt}    {0}
\DeclareEncodingSubset{TS1}{cmvtt}   {0}
\DeclareEncodingSubset{TS1}{pxr}     {0}
\DeclareEncodingSubset{TS1}{pxss}    {0}
\DeclareEncodingSubset{TS1}{pxtt}    {0}
\DeclareEncodingSubset{TS1}{qag}     {0}
\DeclareEncodingSubset{TS1}{qbk}     {0}
\DeclareEncodingSubset{TS1}{qcr}     {0}
\DeclareEncodingSubset{TS1}{qcs}     {0}
\DeclareEncodingSubset{TS1}{qhvc}    {0}
\DeclareEncodingSubset{TS1}{qhv}     {0}
\DeclareEncodingSubset{TS1}{qpl}     {0}
\DeclareEncodingSubset{TS1}{qtm}     {0}
\DeclareEncodingSubset{TS1}{qzc}     {0}
\DeclareEncodingSubset{TS1}{txr}     {0}
\DeclareEncodingSubset{TS1}{txss}    {0}
\DeclareEncodingSubset{TS1}{txtt}    {0}
```

```
\DeclareEncodingSubset{TS1}{lmr}     {1}
\DeclareEncodingSubset{TS1}{lmdh}    {1}
\DeclareEncodingSubset{TS1}{lmss}    {1}
\DeclareEncodingSubset{TS1}{lmssq}   {1}
\DeclareEncodingSubset{TS1}{lmvtt}   {1}
\DeclareEncodingSubset{TS1}{lmtt}    {1} % missing TM, SM, pertenthousand for some reason

\DeclareEncodingSubset{TS1}{ptmx}    {2}
\DeclareEncodingSubset{TS1}{ptmj}    {2}
\DeclareEncodingSubset{TS1}{ul8}     {2}

\DeclareEncodingSubset{TS1}{bch}     {5}  % tofu for blank, ohm
\DeclareEncodingSubset{TS1}{futj}    {5}  % tofu for blank, interrobang/down, ohm
\DeclareEncodingSubset{TS1}{futs}    {5}  % tofu for blank, ohm
\DeclareEncodingSubset{TS1}{futx}    {5}  % probably (currently broken distrib)
\DeclareEncodingSubset{TS1}{pag}     {5}  % tofu for blank, interrobang/down, ohm
\DeclareEncodingSubset{TS1}{pbk}     {5}  % tofu for blank, interrobang/down, ohm
\DeclareEncodingSubset{TS1}{pcr}     {5}  % tofu for blank, interrobang/down, ohm
\DeclareEncodingSubset{TS1}{phv}     {5}  % tofu for blank, interrobang/down, ohm
\DeclareEncodingSubset{TS1}{pnc}     {5}  % tofu for blank, interrobang/down, ohm
\DeclareEncodingSubset{TS1}{pplj}    {5}  % tofu for blank
\DeclareEncodingSubset{TS1}{pplx}    {5}  % tofu for blank
\DeclareEncodingSubset{TS1}{ppl}     {5}  % tofu for blank interrobang/down
\DeclareEncodingSubset{TS1}{ptm}     {5}  % tofu for blank, interrobang/down, ohm
\DeclareEncodingSubset{TS1}{pzc}     {5}  % tofu for blank, interrobang/down, ohm
\DeclareEncodingSubset{TS1}{ul9}     {5}  % tofu for blank, interrobang/down, ohm

\DeclareEncodingSubset{TS1}{dayroms} {6}  % tofu for blank, interrobang/down, ohm
\DeclareEncodingSubset{TS1}{dayrom}  {6}  % tofu for blank, interrobang/down, ohm

\DeclareEncodingSubset{TS1}{augie}   {8}  % really only missing euro
\DeclareEncodingSubset{TS1}{put}     {8}
\DeclareEncodingSubset{TS1}{uag}     {8}  % probably (currently broken distrib)
\DeclareEncodingSubset{TS1}{ugq}     {8}

\DeclareEncodingSubset{TS1}{zi4}     {9}
```

LucidaBright (sold through TUG) probably not quite correct, I guess as I have the older fonts . . .

```
\DeclareEncodingSubset{TS1}{hls}     {5}
\DeclareEncodingSubset{TS1}{hlst}    {5}
\DeclareEncodingSubset{TS1}{hlct}    {5}
\DeclareEncodingSubset{TS1}{hlh}     {5}
\DeclareEncodingSubset{TS1}{hlx}     {8}
\DeclareEncodingSubset{TS1}{hlce}    {8}
\DeclareEncodingSubset{TS1}{hlcn}    {8}
\DeclareEncodingSubset{TS1}{hlcw}    {8}
\DeclareEncodingSubset{TS1}{hlcf}    {8}
```

Below are the newer fonts that have support files for LaTeX. 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.

```
\DeclareEncodingSubset{TS1}{lato-*}                 {0}  % with a bunch of tofu inside
```

```
\DeclareEncodingSubset{TS1}{opensans-*}               {0}  % with a bunch of tofu inside
\DeclareEncodingSubset{TS1}{cantarell-*}              {0}  % with a bunch of tofu inside
\DeclareEncodingSubset{TS1}{fbb-*}                    {0}  % missing centoldstyle
\DeclareEncodingSubset{TS1}{tli}                      {1}  % with lots of tofu inside
\DeclareEncodingSubset{TS1}{Alegreya-*}               {2}
\DeclareEncodingSubset{TS1}{AlegreyaSans-*}           {2}
\DeclareEncodingSubset{TS1}{DejaVuSans-TLF}           {2}
\DeclareEncodingSubset{TS1}{DejaVuSansCondensed-TLF}  {2}
\DeclareEncodingSubset{TS1}{DejaVuSansMono-TLF}       {2}
\DeclareEncodingSubset{TS1}{EBGaramond-*}             {2}
\DeclareEncodingSubset{TS1}{Tempora-TLF}              {2}
\DeclareEncodingSubset{TS1}{Tempora-TOsF}             {2}
\DeclareEncodingSubset{TS1}{Arimo-TLF}                {3}
\DeclareEncodingSubset{TS1}{Carlito-*}                {3}
\DeclareEncodingSubset{TS1}{FiraSans-*}               {3}
\DeclareEncodingSubset{TS1}{IBMPlexSans-TLF}          {3}
\DeclareEncodingSubset{TS1}{Merriweather-OsF}         {3}
\DeclareEncodingSubset{TS1}{Montserrat-*}             {3}
\DeclareEncodingSubset{TS1}{MontserratAlternates-*}   {3}
\DeclareEncodingSubset{TS1}{SourceCodePro-TLF}        {3}
\DeclareEncodingSubset{TS1}{SourceCodePro-TOsF}       {3}
\DeclareEncodingSubset{TS1}{SourceSansPro-*}          {3}
\DeclareEncodingSubset{TS1}{SourceSerifPro-*}         {3}
\DeclareEncodingSubset{TS1}{Tinos-TLF}                {3}
\DeclareEncodingSubset{TS1}{AccanthisADFStdNoThree-LF}{4}
\DeclareEncodingSubset{TS1}{Cabin-TLF}                {4}
\DeclareEncodingSubset{TS1}{Caladea-TLF}              {4}
\DeclareEncodingSubset{TS1}{Chivo-*}                  {4}
\DeclareEncodingSubset{TS1}{ClearSans-TLF}            {4}
\DeclareEncodingSubset{TS1}{Coelacanth-LF}            {4}
\DeclareEncodingSubset{TS1}{CrimsonPro-*}             {4}
\DeclareEncodingSubset{TS1}{FiraMono-TLF}             {4}
\DeclareEncodingSubset{TS1}{FiraMono-TOsF}            {4}
\DeclareEncodingSubset{TS1}{Go-TLF}                   {4}
\DeclareEncodingSubset{TS1}{GoMono-TLF}               {4}
\DeclareEncodingSubset{TS1}{InriaSans-*}              {4}
\DeclareEncodingSubset{TS1}{InriaSerif-*}             {4}
\DeclareEncodingSubset{TS1}{LibertinusSans-*}         {4}
\DeclareEncodingSubset{TS1}{LibertinusSerif-*}        {4}
\DeclareEncodingSubset{TS1}{LibreBodoni-TLF}          {4}
\DeclareEncodingSubset{TS1}{LibreFranklin-TLF}        {4}
\DeclareEncodingSubset{TS1}{LinguisticsPro-LF}        {4}
\DeclareEncodingSubset{TS1}{LinguisticsPro-OsF}       {4}
\DeclareEncodingSubset{TS1}{LinuxBiolinumT-*}         {4}
\DeclareEncodingSubset{TS1}{LinuxLibertineT-*}        {4}
\DeclareEncodingSubset{TS1}{MerriweatherSans-OsF}     {4}
\DeclareEncodingSubset{TS1}{MintSpirit-*}             {4}
\DeclareEncodingSubset{TS1}{MintSpiritNoTwo-*}        {4}
\DeclareEncodingSubset{TS1}{PTMono-TLF}               {4}
\DeclareEncodingSubset{TS1}{PTSans-TLF}               {4}
\DeclareEncodingSubset{TS1}{PTSansCaption-TLF}        {4}
\DeclareEncodingSubset{TS1}{PTSansNarrow-TLF}         {4}
\DeclareEncodingSubset{TS1}{PTSerif-TLF}              {4}
```

```
\DeclareEncodingSubset{TS1}{PTSerifCaption-TLF}       {4}
\DeclareEncodingSubset{TS1}{Raleway-TLF}              {4}
\DeclareEncodingSubset{TS1}{Raleway-TOsF}             {4}
\DeclareEncodingSubset{TS1}{Roboto-*}                 {4}
\DeclareEncodingSubset{TS1}{RobotoMono-TLF}           {4}
\DeclareEncodingSubset{TS1}{RobotoSlab-TLF}           {4}
\DeclareEncodingSubset{TS1}{Rosario-*}                {4}
\DeclareEncodingSubset{TS1}{SticksTooText-*}          {4}
\DeclareEncodingSubset{TS1}{UniversalisADFStd-LF}     {4}

\DeclareEncodingSubset{TS1}{Almendra-OsF}             {5}
\DeclareEncodingSubset{TS1}{Baskervaldx-*}            {5}
\DeclareEncodingSubset{TS1}{BaskervilleF-*}           {5}
\DeclareEncodingSubset{TS1}{Bitter-TLF}               {5}
\DeclareEncodingSubset{TS1}{Cinzel-LF}                {5}
\DeclareEncodingSubset{TS1}{CinzelDecorative-LF}      {5}
\DeclareEncodingSubset{TS1}{DejaVuSerif-TLF}          {5}
\DeclareEncodingSubset{TS1}{DejaVuSerifCondensed-TLF} {5}
\DeclareEncodingSubset{TS1}{GilliusADF-LF}            {5}
\DeclareEncodingSubset{TS1}{GilliusADFCond-LF}        {5}
\DeclareEncodingSubset{TS1}{GilliusADFNoTwo-LF}       {5}
\DeclareEncodingSubset{TS1}{GilliusADFNoTwoCond-LF}   {5}
\DeclareEncodingSubset{TS1}{LobsterTwo-LF}            {5}
\DeclareEncodingSubset{TS1}{OldStandard-TLF}          {5}
\DeclareEncodingSubset{TS1}{PlayfairDisplay-TLF}      {5}
\DeclareEncodingSubset{TS1}{PlayfairDisplay-TOsF}     {5}
\DeclareEncodingSubset{TS1}{TheanoDidot-TLF}          {5}
\DeclareEncodingSubset{TS1}{TheanoDidot-TOsF}         {5}
\DeclareEncodingSubset{TS1}{TheanoModern-TLF}         {5}
\DeclareEncodingSubset{TS1}{TheanoModern-TOsF}        {5}
\DeclareEncodingSubset{TS1}{TheanoOldStyle-TLF}       {5}
\DeclareEncodingSubset{TS1}{TheanoOldStyle-TOsF}      {5}

\DeclareEncodingSubset{TS1}{Crimson-TLF}              {6}
\DeclareEncodingSubset{TS1}{IBMPlexMono-TLF}          {6}
\DeclareEncodingSubset{TS1}{IBMPlexSerif-TLF}         {6}
\DeclareEncodingSubset{TS1}{LibertinusMono-TLF}       {6}
\DeclareEncodingSubset{TS1}{LibertinusSerifDisplay-LF}{6}
\DeclareEncodingSubset{TS1}{LinuxLibertineDisplayT-*} {6}
\DeclareEncodingSubset{TS1}{LinuxLibertineMonoT-LF}   {6}
\DeclareEncodingSubset{TS1}{LinuxLibertineMonoT-TLF}  {6}
\DeclareEncodingSubset{TS1}{Overlock-LF}              {6}

\DeclareEncodingSubset{TS1}{CormorantGaramond-*}      {7}
\DeclareEncodingSubset{TS1}{Heuristica-TLF}           {7}
\DeclareEncodingSubset{TS1}{Heuristica-TOsF}          {7}
\DeclareEncodingSubset{TS1}{IMFELLEnglish-TLF}        {7}
\DeclareEncodingSubset{TS1}{LibreBaskerville-TLF}     {7}
\DeclareEncodingSubset{TS1}{LibreCaslon-*}            {7}
\DeclareEncodingSubset{TS1}{Marcellus-LF}             {7}
\DeclareEncodingSubset{TS1}{NotoSans-*}               {7}
\DeclareEncodingSubset{TS1}{NotoSansMono-TLF}         {7}
\DeclareEncodingSubset{TS1}{NotoSansMono-TOsF}        {7}
\DeclareEncodingSubset{TS1}{NotoSerif-*}              {7}
\DeclareEncodingSubset{TS1}{Quattrocento-TLF}         {7}
\DeclareEncodingSubset{TS1}{QuattrocentoSans-TLF}     {7}
```

```
\DeclareEncodingSubset{TS1}{XCharter-TLF}            {7}
\DeclareEncodingSubset{TS1}{XCharter-TOsF}           {7}
\DeclareEncodingSubset{TS1}{erewhon-*}               {7}
\DeclareEncodingSubset{TS1}{ComicNeue-TLF}           {7}
\DeclareEncodingSubset{TS1}{ComicNeueAngular-TLF}    {7}
\DeclareEncodingSubset{TS1}{Forum-LF}                {7}  % the superiors are missing
```

```
\DeclareEncodingSubset{TS1}{Cochineal-*}             {8}
```

```
\DeclareEncodingSubset{TS1}{AlgolRevived-TLF}        {9}
```

59 Legacy symbol support for lists and footnote symbols

\UseLegacyTextSymbols

```
\def\UseLegacyTextSymbols{%
  \DeclareTextSymbolDefault{\textasteriskcentered}{OMS}%
  \DeclareTextSymbolDefault{\textbardbl}{OMS}%
  \DeclareTextSymbolDefault{\textbullet}{OMS}%
  \DeclareTextSymbolDefault{\textdaggerdbl}{OMS}%
  \DeclareTextSymbolDefault{\textdagger}{OMS}%
  \DeclareTextSymbolDefault{\textparagraph}{OMS}%
  \DeclareTextSymbolDefault{\textperiodcentered}{OMS}%
  \DeclareTextSymbolDefault{\textsection}{OMS}%
  \UndeclareTextCommand{\textsection}{T1}%
  \expandafter\let\csname oldstylenums \expandafter\endcsname
                  \csname legacyoldstylenums \endcsname
}
```

\textlegacyasteriskcentered
\textlegacybardbl
\textlegacybullet
\textlegacydaggerdbl
\textlegacydagger
\textlegacyparagraph
\textlegacyperiodcentered
\textlegacysection

Here are new names for the legacy symbols that LaTeX used to pick up from the OMS encoded fonts (and used dor itemize lists or footnote symbols.

We go the roundabout way via separate OMS declarations so that

```
\renewcommand\textbullet{\textlegacybullet}
```

doesn't produce an endless loop.

```
\DeclareTextSymbol{\textlegacyasteriskcentered}{OMS}{3}   % "03
\DeclareTextSymbol{\textlegacybardbl}{OMS}{107}           % "6B
\DeclareTextSymbol{\textlegacybullet}{OMS}{15}            % "0F
\DeclareTextSymbol{\textlegacydaggerdbl}{OMS}{122}        % "7A
\DeclareTextSymbol{\textlegacydagger}{OMS}{121}           % "79
\DeclareTextSymbol{\textlegacyparagraph}{OMS}{123}        % "7B
\DeclareTextSymbol{\textlegacyperiodcentered}{OMS}{1}     % "01
\DeclareTextSymbol{\textlegacysection}{OMS}{120}          % "78
```

```
\DeclareTextSymbolDefault{\textlegacyasteriskcentered}{OMS}
\DeclareTextSymbolDefault{\textlegacybardbl}{OMS}
\DeclareTextSymbolDefault{\textlegacybullet}{OMS}
\DeclareTextSymbolDefault{\textlegacydaggerdbl}{OMS}
\DeclareTextSymbolDefault{\textlegacydagger}{OMS}
\DeclareTextSymbolDefault{\textlegacyparagraph}{OMS}
\DeclareTextSymbolDefault{\textlegacyperiodcentered}{OMS}
\DeclareTextSymbolDefault{\textlegacysection}{OMS}
```

Supporting rollback …

```
⟨/2ekernel | latexrelease⟩
⟨latexrelease⟩\EndIncludeInRelease
⟨latexrelease⟩\IncludeInRelease{0000/00/00}%
⟨latexrelease⟩   {\DeclareEncodingSubset}{Text companion symbols}%
⟨latexrelease⟩
⟨latexrelease⟩\let\DeclareEncodingSubset\@undefined
⟨latexrelease⟩\let\CheckEncodingSubset\@undefined
⟨latexrelease⟩
⟨latexrelease⟩\DeclareTextSymbolDefault{\textdollar}{OT1}
⟨latexrelease⟩\DeclareTextSymbolDefault{\textsterling}{OT1}
⟨latexrelease⟩\DeclareTextCommand{\textdollar}{OT1}{\hmode@bgroup
⟨latexrelease⟩   \ifdim \fontdimen\@ne\font >\z@
⟨latexrelease⟩     \slshape
⟨latexrelease⟩   \else
⟨latexrelease⟩     \upshape
⟨latexrelease⟩   \fi
⟨latexrelease⟩   \char`\$\egroup}
⟨latexrelease⟩\DeclareTextCommand{\textsterling}{OT1}{\hmode@bgroup
⟨latexrelease⟩   \ifdim \fontdimen\@ne\font >\z@
⟨latexrelease⟩      \itshape
⟨latexrelease⟩   \else
⟨latexrelease⟩      \fontshape{ui}\selectfont
⟨latexrelease⟩   \fi
⟨latexrelease⟩   \char`\$\egroup}
⟨latexrelease⟩\DeclareTextCommand{\textperthousand}{T1}
⟨latexrelease⟩   {\%\char 24 }
⟨latexrelease⟩
⟨latexrelease⟩\DeclareTextSymbolDefault{\textasteriskcentered}{OMS}
⟨latexrelease⟩\DeclareTextSymbolDefault{\textbullet}{OMS}
⟨latexrelease⟩\DeclareTextSymbolDefault{\textdaggerdbl}{OMS}
⟨latexrelease⟩\DeclareTextSymbolDefault{\textdagger}{OMS}
⟨latexrelease⟩\DeclareTextSymbolDefault{\textparagraph}{OMS}
⟨latexrelease⟩\DeclareTextSymbolDefault{\textperiodcentered}{OMS}
⟨latexrelease⟩\DeclareTextSymbolDefault{\textsection}{OMS}
⟨latexrelease⟩
⟨latexrelease⟩\DeclareTextSymbolDefault{\textbardbl}{OMS}
⟨latexrelease⟩\let\textbrokenbar\@undefined
⟨latexrelease⟩\let\textcelsius\@undefined
⟨latexrelease⟩\let\textcent\@undefined
⟨latexrelease⟩\DeclareTextCommandDefault{\textcopyright}{\textcircled{c}}
⟨latexrelease⟩\let\textdegree\@undefined
⟨latexrelease⟩\let\textdiv\@undefined
⟨latexrelease⟩\let\textlnot\@undefined
⟨latexrelease⟩\let\textonehalf\@undefined
⟨latexrelease⟩\let\textonequarter\@undefined
⟨latexrelease⟩\let\textonesuperior\@undefined
⟨latexrelease⟩\DeclareTextCommandDefault{\textordfeminine}{\textsuperscript{a}}
⟨latexrelease⟩\DeclareTextCommandDefault{\textordmasculine}{\textsuperscript{o}}
⟨latexrelease⟩\let\textpm\@undefined
⟨latexrelease⟩\let\textquotesingle\@undefined
⟨latexrelease⟩\let\textquotestraightbase\@undefined
⟨latexrelease⟩\let\textquotestraightdblbase\@undefined
⟨latexrelease⟩\DeclareTextCommandDefault{\textregistered}{\textcircled{%
⟨latexrelease⟩      \check@mathfonts\fontsize\sf@size\z@\math@fontsfalse\selectfont R}}
⟨latexrelease⟩\let\textthreequartersemdash\@undefined
⟨latexrelease⟩\let\textthreequarters\@undefined
⟨latexrelease⟩\let\textthreesuperior\@undefined
⟨latexrelease⟩\let\texttimes\@undefined
⟨latexrelease⟩\DeclareTextCommandDefault{\texttrademark}{\textsuperscript{TM}}
⟨latexrelease⟩\let\texttwelveudash\@undefined
⟨latexrelease⟩\let\texttwosuperior\@undefined
⟨latexrelease⟩\let\textyen\@undefined
⟨latexrelease⟩
⟨latexrelease⟩\let\textcapitalcompwordmark\@undefined
⟨latexrelease⟩\let\textascendercompwordmark\@undefined
⟨latexrelease⟩
⟨latexrelease⟩\DeclareTextAccentDefault{\textcircled}{OMS}
⟨latexrelease⟩\DeclareTextAccentDefault{\t}{OML}
⟨latexrelease⟩
⟨latexrelease⟩\let\capitalacute\@undefined
⟨latexrelease⟩\let\capitalbreve\@undefined
⟨latexrelease⟩\let\capitalcaron\@undefined
⟨latexrelease⟩\let\capitalcedilla\@undefined
⟨latexrelease⟩\let\capitalcircumflex\@undefined
⟨latexrelease⟩\let\capitaldieresis\@undefined
⟨latexrelease⟩\let\capitaldotaccent\@undefined
⟨latexrelease⟩\let\capitalgrave\@undefined
⟨latexrelease⟩\let\capitalhungarumlaut\@undefined
⟨latexrelease⟩\let\capitalmacron\@undefined
⟨latexrelease⟩\let\capitalnewtie\@undefined
⟨latexrelease⟩\let\capitalogonek\@undefined
⟨latexrelease⟩\let\capitalring\@undefined
⟨latexrelease⟩\let\capitaltie\@undefined
⟨latexrelease⟩\let\capitaltilde\@undefined
⟨latexrelease⟩\let\newtie\@undefined
⟨latexrelease⟩
⟨latexrelease⟩\let\textlbrackdbl\@undefined
⟨latexrelease⟩\let\textrbrackdbl\@undefined
⟨latexrelease⟩
⟨latexrelease⟩\let\texteightoldstyle\@undefined
⟨latexrelease⟩\let\textfiveoldstyle\@undefined
⟨latexrelease⟩\let\textfouroldstyle\@undefined
⟨latexrelease⟩\let\textnineoldstyle\@undefined
⟨latexrelease⟩\let\textoneoldstyle\@undefined
⟨latexrelease⟩\let\textsevenoldstyle\@undefined
⟨latexrelease⟩\let\textsixoldstyle\@undefined
⟨latexrelease⟩\let\textthreeoldstyle\@undefined
⟨latexrelease⟩\let\texttwooldstyle\@undefined
⟨latexrelease⟩\let\textzerooldstyle\@undefined
⟨latexrelease⟩
⟨latexrelease⟩\let\textacutedbl\@undefined
⟨latexrelease⟩\let\textasciiacute\@undefined
⟨latexrelease⟩\let\textasciibreve\@undefined
⟨latexrelease⟩\let\textasciicaron\@undefined
⟨latexrelease⟩\let\textasciidieresis\@undefined
⟨latexrelease⟩\let\textasciigrave\@undefined
⟨latexrelease⟩\let\textasciimacron\@undefined
```

```
⟨latexrelease⟩\let\textgravedbl\@undefined
⟨latexrelease⟩\let\texttildelow\@undefined
⟨latexrelease⟩
⟨latexrelease⟩\let\textbaht\@undefined
⟨latexrelease⟩\let\textbigcircle\@undefined
⟨latexrelease⟩\let\textborn\@undefined
⟨latexrelease⟩\let\textcentoldstyle\@undefined
⟨latexrelease⟩\let\textcircledP\@undefined
⟨latexrelease⟩\let\textcopyleft\@undefined
⟨latexrelease⟩\let\textdblhyphenchar\@undefined
⟨latexrelease⟩\let\textdblhyphen\@undefined
⟨latexrelease⟩\let\textdied\@undefined
⟨latexrelease⟩\let\textdiscount\@undefined
⟨latexrelease⟩\let\textdivorced\@undefined
⟨latexrelease⟩\let\textdollaroldstyle\@undefined
⟨latexrelease⟩\let\textguarani\@undefined
⟨latexrelease⟩\let\textleaf\@undefined
⟨latexrelease⟩\let\textlquill\@undefined
⟨latexrelease⟩\let\textmarried\@undefined
⟨latexrelease⟩\let\textmho\@undefined
⟨latexrelease⟩\let\textmusicalnote\@undefined
⟨latexrelease⟩\let\textnaira\@undefined
⟨latexrelease⟩\let\textopenbullet\@undefined
⟨latexrelease⟩\let\textpeso\@undefined
⟨latexrelease⟩\let\textpilcrow\@undefined
⟨latexrelease⟩\let\textrecipe\@undefined
⟨latexrelease⟩\let\textreferencemark\@undefined
⟨latexrelease⟩\let\textrquill\@undefined
⟨latexrelease⟩\let\textservicemark\@undefined
⟨latexrelease⟩\let\textsurd\@undefined
⟨latexrelease⟩
⟨latexrelease⟩\DeclareTextCommand{\textpertenthousand}{T1}
⟨latexrelease⟩                   {\%\char 24\char 24 }
⟨latexrelease⟩
⟨latexrelease⟩\let\textlangle\@undefined
⟨latexrelease⟩\let\textrangle\@undefined
⟨latexrelease⟩
⟨latexrelease⟩\let\textcolonmonetary\@undefined
⟨latexrelease⟩\let\textdong\@undefined
⟨latexrelease⟩\let\textdownarrow\@undefined
⟨latexrelease⟩\let\textleftarrow\@undefined
⟨latexrelease⟩\let\textlira\@undefined
⟨latexrelease⟩\let\textrightarrow\@undefined
⟨latexrelease⟩\let\textuparrow\@undefined
⟨latexrelease⟩\let\textwon\@undefined
⟨latexrelease⟩
⟨latexrelease⟩\let\textestimated\@undefined
⟨latexrelease⟩\let\textnumero\@undefined
⟨latexrelease⟩
⟨latexrelease⟩\let\textflorin\@undefined
⟨latexrelease⟩\let\textcurrency\@undefined
⟨latexrelease⟩
⟨latexrelease⟩\let\textfractionsolidus\@undefined
⟨latexrelease⟩\let\textohm\@undefined
```

```
⟨latexrelease⟩\let\textmu\@undefined
⟨latexrelease⟩\let\textminus\@undefined
⟨latexrelease⟩
⟨latexrelease⟩\let\textblank\@undefined
⟨latexrelease⟩\let\textinterrobangdown\@undefined
⟨latexrelease⟩\let\textinterrobang\@undefined
⟨latexrelease⟩
⟨latexrelease⟩\let\texteuro\@undefined
⟨latexrelease⟩
⟨latexrelease⟩\let\textcelsius\@undefined
⟨latexrelease⟩\let\textonesuperior\@undefined
⟨latexrelease⟩\let\textthreequartersemdash\@undefined
⟨latexrelease⟩\let\textthreesuperior\@undefined
⟨latexrelease⟩\let\texttwelveudash\@undefined
⟨latexrelease⟩\let\texttwosuperior\@undefined
⟨latexrelease⟩\let\textbardbl\@undefined
⟨latexrelease⟩
⟨latexrelease⟩\let\UseLegacyTextSymbols\@undefined
⟨latexrelease⟩\let\textlegacyasteriskcentered\@undefined
⟨latexrelease⟩\let\textlegacybardbl\@undefined
⟨latexrelease⟩\let\textlegacybullet\@undefined
⟨latexrelease⟩\let\textlegacydaggerdbl\@undefined
⟨latexrelease⟩\let\textlegacydagger\@undefined
⟨latexrelease⟩\let\textlegacyparagraph\@undefined
⟨latexrelease⟩\let\textlegacyperiodcentered\@undefined
⟨latexrelease⟩\let\textlegacysection\@undefined
⟨latexrelease⟩
⟨latexrelease⟩\EndIncludeInRelease
⟨*2ekernel⟩
⟨/2ekernel⟩
```

60 The `textcomp` package

```
⟨*TS1sty⟩
\providecommand\DeclareRelease[3]{}
\providecommand\DeclareCurrentRelease[2]{}

\DeclareRelease{}{2018-08-11}{textcomp-2018-08-11.sty}
\DeclareCurrentRelease{}{2020-02-02}

\ProvidesPackage{textcomp}
 [2020/02/02 v2.0n Standard LaTeX package]
```

A precaution in case this is used without rebuilding the format.

```
\NeedsTeXFormat{LaTeX2e}[2020/02/02]
```

This is implemented by defining the default subset:

```
\DeclareOption{full}{\DeclareEncodingSubset{TS1}{?}{0}}
\DeclareOption{almostfull}{\DeclareEncodingSubset{TS1}{?}{1}}
\DeclareOption{euro}{\DeclareEncodingSubset{TS1}{?}{8}}
\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.

```
\DeclareOption{error}{\gdef\tc@errorwarn{\PackageError{textcomp}}}
\DeclareOption{warn}{\gdef\tc@errorwarn#1#2{\PackageWarning{textcomp}{#1}}}
\DeclareOption{info}{\gdef\tc@errorwarn#1#2{\PackageInfo{textcomp}{#1}}}
\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 somehting that is recommended.

```
\DeclareOption{force}{%
    \def\CheckEncodingSubset#1#2#3#4#5{%
      \ifnum #4>%
           0\csname #2:?\endcsname
           \relax
       \expandafter\@firstoftwo
     \else
       \expandafter\@secondoftwo
    \fi
     {#1{#2}}{#3}%
     #5}%
}
```

```
\ExecuteOptions{info}
\ProcessOptions\relax
```

There is not much else to do nowadays, because everything is already set up in the LaTeX kernel.

```
\InputIfFileExists{textcomp.cfg}
  {\PackageInfo{textcomp}{Local configuration file used}}{}
```

```
⟨/TS1sty⟩
```

60.1 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 categorised in the sub-encodings only into 6 classes not 10.

```
⟨*TS1oldsty⟩
\ProvidesPackage{textcomp}
  [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-TeX 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:

```
\PackageInfo{textcomp}{Sub-encoding information:\MessageBreak
    \space\space 5 = only ISO-Adobe without
                              \string\textcurrency\MessageBreak
    \space\space 4 = 5 + \string\texteuro\MessageBreak
    \space\space 3 = 4 + \string\textohm\MessageBreak
    \space\space 2 = 3 + \noexpand\textestimated+
                                \string\textcurrency\MessageBreak
    \space\space 1 = TS1 - \noexpand\textcircled-
                                            \string\t\MessageBreak
    \space\space 0 = TS1 (full)\MessageBreak
    Font families with sub-encoding setting implement\MessageBreak
    only a restricted character set as indicated.\MessageBreak
    Family '?' is the default used for unknown fonts.\MessageBreak
    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 ?.

```
\def\DeclareEncodingSubset#1#2#3{%
  \@ifundefined{#1:#2}%
    {\PackageInfo{textcomp}{Setting #2 sub-encoding to #1/#3}}%
    {\PackageInfo{textcomp}{Changing #2 sub-encoding to #1/#3}}%
  \@namedef{#1:#2}{#3}}
\@onlypreamble\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

```
\newif\iftc@forced   \tc@forcedfalse
```

This is implemented by defining the default subset:

```
\DeclareOption{full}{\DeclareEncodingSubset{TS1}{?}{0}}
\DeclareOption{almostfull}{\DeclareEncodingSubset{TS1}{?}{1}}
\DeclareOption{euro}{\DeclareEncodingSubset{TS1}{?}{4}}
\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.

```
\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.

```
\def\tc@errorwarn{\PackageError}
\DeclareOption{warn}{\gdef\tc@errorwarn#1#2#3{\PackageWarning{#1}{#2}}}
\DeclareOption{quiet}{\gdef\tc@errorwarn#1#2#3{}}
```

```
\ExecuteOptions{almostfull}
\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.

```
\iftc@forced
```

If the “force” option was given we always use the default for testing against.

```
\def\CheckEncodingSubset#1#2#3#4#5{%
    \ifnum #4>%
        0\csname #2:?\endcsname
        \relax
   \expandafter\@firstoftwo
  \else
   \expandafter\@secondoftwo
 \fi
  {#1{#2}}{#3}%
  #5%
}
```

In normal circumstances the test is a bit more complicated: first check if there exists a macro \⟨*arg2*⟩:⟨*current-family*⟩ and if so use that value to test against, otherwise use the default to test against.

```
\else
\def\CheckEncodingSubset#1#2#3#4#5{%
    \ifnum #4>%
      \expandafter\ifx\csname #2:\f@family\endcsname\relax
        0\csname #2:?\endcsname
      \else
        \csname #2:\f@family\endcsname
      \fi
   \relax
   \expandafter\@firstoftwo
  \else
   \expandafter\@secondoftwo
 \fi
  {#1{#2}}{#3}%
  #5%
}
\fi
```

`\tc@subst`

```
\def\tc@subst#1{%
  \tc@errorwarn{textcomp}%
    {Symbol \string#1 not provided by\MessageBreak
     font family \f@family\space
     in TS1 encoding.\MessageBreak Default family used instead}\@eha
  \bgroup\fontfamily\textcompsubstdefault\selectfont#1\egroup
}
```

`\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.

```
% error commands take argument:
% #1 symbol to be used
\def\tc@error#1{%
   \PackageError{textcomp}%  % should be latex error if general
    {Accent \string#1 not provided by\MessageBreak
     font family \f@family\space
     in TS1 encoding}\@eha
}
```

\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 “=”.

```
\def\tc@fake@euro#1{%
    \leavevmode
    \PackageInfo{textcomp}{Faking \noexpand#1for font family
                                    \f@family\MessageBreak in TS1 encoding}%
    \valign{##\cr
        \vfil\hbox to 0.07em{\dimen@\f@size\p@
                                     \math@fontsfalse
                                     \fontsize{.7\dimen@}\z@\selectfont=\hss}%
        \vfil\cr%
        \hbox{C}\crcr
    }%
}
```

\tc@check@symbol
\tc@check@accent

These are two abbreviations that we use below to check symbols and accents in TS1. 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 \textcurreny is not available.

```
\def\tc@check@symbol{\CheckEncodingSubset\UseTextSymbol{TS1}\tc@subst}
\def\tc@check@accent{\CheckEncodingSubset\UseTextAccent{TS1}\tc@error}
```

We start with the commands that are “safe” and which can be unconditionally set up, first the accents…

```
\DeclareTextAccentDefault{\capitalcedilla}{TS1}
\DeclareTextAccentDefault{\capitalogonek}{TS1}
\DeclareTextAccentDefault{\capitalgrave}{TS1}
\DeclareTextAccentDefault{\capitalacute}{TS1}
\DeclareTextAccentDefault{\capitalcircumflex}{TS1}
\DeclareTextAccentDefault{\capitaltilde}{TS1}
\DeclareTextAccentDefault{\capitaldieresis}{TS1}
\DeclareTextAccentDefault{\capitalhungarumlaut}{TS1}
\DeclareTextAccentDefault{\capitalring}{TS1}
\DeclareTextAccentDefault{\capitalcaron}{TS1}
\DeclareTextAccentDefault{\capitalbreve}{TS1}
\DeclareTextAccentDefault{\capitalmacron}{TS1}
\DeclareTextAccentDefault{\capitaldotaccent}{TS1}
```

…and then the other glyphs.

```
\DeclareTextSymbolDefault{\textcapitalcompwordmark}{TS1}
\DeclareTextSymbolDefault{\textascendercompwordmark}{TS1}
\DeclareTextSymbolDefault{\textquotestraightbase}{TS1}
\DeclareTextSymbolDefault{\textquotestraightdblbase}{TS1}
\DeclareTextSymbolDefault{\texttwelveudash}{TS1}
\DeclareTextSymbolDefault{\textthreequartersemdash}{TS1}
\DeclareTextSymbolDefault{\textdollar}{TS1}
\DeclareTextSymbolDefault{\textquotesingle}{TS1}
\DeclareTextSymbolDefault{\textasteriskcentered}{TS1}
\DeclareTextSymbolDefault{\textfractionsolidus}{TS1}
```

```
\DeclareTextSymbolDefault{\textminus}{TS1}
\DeclareTextSymbolDefault{\textlbrackdbl}{TS1}
\DeclareTextSymbolDefault{\textrbrackdbl}{TS1}
\DeclareTextSymbolDefault{\textasciigrave}{TS1}
\DeclareTextSymbolDefault{\texttildelow}{TS1}
\DeclareTextSymbolDefault{\textasciibreve}{TS1}
\DeclareTextSymbolDefault{\textasciicaron}{TS1}
\DeclareTextSymbolDefault{\textgravedbl}{TS1}
\DeclareTextSymbolDefault{\textacutedbl}{TS1}
\DeclareTextSymbolDefault{\textdagger}{TS1}
\DeclareTextSymbolDefault{\textdaggerdbl}{TS1}
\DeclareTextSymbolDefault{\textbardbl}{TS1}
\DeclareTextSymbolDefault{\textperthousand}{TS1}
\DeclareTextSymbolDefault{\textbullet}{TS1}
\DeclareTextSymbolDefault{\textcelsius}{TS1}
\DeclareTextSymbolDefault{\textflorin}{TS1}
\DeclareTextSymbolDefault{\texttrademark}{TS1}
\DeclareTextSymbolDefault{\textcent}{TS1}
\DeclareTextSymbolDefault{\textsterling}{TS1}
\DeclareTextSymbolDefault{\textyen}{TS1}
\DeclareTextSymbolDefault{\textbrokenbar}{TS1}
\DeclareTextSymbolDefault{\textsection}{TS1}
\DeclareTextSymbolDefault{\textasciidieresis}{TS1}
\DeclareTextSymbolDefault{\textcopyright}{TS1}
\DeclareTextSymbolDefault{\textordfeminine}{TS1}
\DeclareTextSymbolDefault{\textlnot}{TS1}
\DeclareTextSymbolDefault{\textregistered}{TS1}
\DeclareTextSymbolDefault{\textasciimacron}{TS1}
\DeclareTextSymbolDefault{\textdegree}{TS1}
\DeclareTextSymbolDefault{\textpm}{TS1}
\DeclareTextSymbolDefault{\texttwosuperior}{TS1}
\DeclareTextSymbolDefault{\textthreesuperior}{TS1}
\DeclareTextSymbolDefault{\textasciiacute}{TS1}
\DeclareTextSymbolDefault{\textmu}{TS1}
\DeclareTextSymbolDefault{\textparagraph}{TS1}
\DeclareTextSymbolDefault{\textperiodcentered}{TS1}
\DeclareTextSymbolDefault{\textonesuperior}{TS1}
\DeclareTextSymbolDefault{\textordmasculine}{TS1}
\DeclareTextSymbolDefault{\textonequarter}{TS1}
\DeclareTextSymbolDefault{\textonehalf}{TS1}
\DeclareTextSymbolDefault{\textthreequarters}{TS1}
\DeclareTextSymbolDefault{\texttimes}{TS1}
\DeclareTextSymbolDefault{\textdiv}{TS1}
```

The `\texteuro` is only available for subsets with id 4 or less. Otherwise we fake the glyph using `\tc@fake@euro`

```
\DeclareTextCommandDefault{\texteuro}
   {\CheckEncodingSubset\UseTextSymbol{TS1}\tc@fake@euro5\texteuro}
```

The `\textohm` is only available for subsets with id 3 or less. Otherwise we produce an error.

```
\DeclareTextCommandDefault{\textohm}{\tc@check@symbol4\textohm}
```

The `\textestimated` and `\textcurrency` are only provided for fonts with subset encoding with id 2 or less.

```
\DeclareTextCommandDefault{\textestimated}%
    {\tc@check@symbol3\textestimated}
\DeclareTextCommandDefault{\textcurrency}%
    {\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.

```
\DeclareTextCommandDefault{\capitaltie}%
    {\tc@check@accent2\capitaltie}
\DeclareTextCommandDefault{\newtie}%
    {\tc@check@accent2\newtie}
\DeclareTextCommandDefault{\capitalnewtie}%
    {\tc@check@accent2\capitalnewtie}
\DeclareTextCommandDefault{\textleftarrow}%
    {\tc@check@symbol2\textleftarrow}
\DeclareTextCommandDefault{\textrightarrow}%
    {\tc@check@symbol2\textrightarrow}
\DeclareTextCommandDefault{\textblank}%
    {\tc@check@symbol2\textblank}
\DeclareTextCommandDefault{\textdblhyphen}%
    {\tc@check@symbol2\textdblhyphen}
\DeclareTextCommandDefault{\textzerooldstyle}%
    {\tc@check@symbol2\textzerooldstyle}
\DeclareTextCommandDefault{\textoneoldstyle}%
    {\tc@check@symbol2\textoneoldstyle}
\DeclareTextCommandDefault{\texttwooldstyle}%
    {\tc@check@symbol2\texttwooldstyle}
\DeclareTextCommandDefault{\textthreeoldstyle}%
    {\tc@check@symbol2\textthreeoldstyle}
\DeclareTextCommandDefault{\textfouroldstyle}%
    {\tc@check@symbol2\textfouroldstyle}
\DeclareTextCommandDefault{\textfiveoldstyle}%
    {\tc@check@symbol2\textfiveoldstyle}
\DeclareTextCommandDefault{\textsixoldstyle}%
    {\tc@check@symbol2\textsixoldstyle}
\DeclareTextCommandDefault{\textsevenoldstyle}%
    {\tc@check@symbol2\textsevenoldstyle}
\DeclareTextCommandDefault{\texteightoldstyle}%
    {\tc@check@symbol2\texteightoldstyle}
\DeclareTextCommandDefault{\textnineoldstyle}%
    {\tc@check@symbol2\textnineoldstyle}
\DeclareTextCommandDefault{\textlangle}%
    {\tc@check@symbol2\textlangle}
\DeclareTextCommandDefault{\textrangle}%
    {\tc@check@symbol2\textrangle}
\DeclareTextCommandDefault{\textmho}%
    {\tc@check@symbol2\textmho}
\DeclareTextCommandDefault{\textbigcircle}%
    {\tc@check@symbol2\textbigcircle}
\DeclareTextCommandDefault{\textuparrow}%
    {\tc@check@symbol2\textuparrow}
\DeclareTextCommandDefault{\textdownarrow}%
    {\tc@check@symbol2\textdownarrow}
\DeclareTextCommandDefault{\textborn}%
```

```
    {\tc@check@symbol2\textborn}
\DeclareTextCommandDefault{\textdivorced}%
    {\tc@check@symbol2\textdivorced}
\DeclareTextCommandDefault{\textdied}%
    {\tc@check@symbol2\textdied}
\DeclareTextCommandDefault{\textleaf}%
    {\tc@check@symbol2\textleaf}
\DeclareTextCommandDefault{\textmarried}%
    {\tc@check@symbol2\textmarried}
\DeclareTextCommandDefault{\textmusicalnote}%
    {\tc@check@symbol2\textmusicalnote}
\DeclareTextCommandDefault{\textdblhyphenchar}%
    {\tc@check@symbol2\textdblhyphenchar}
\DeclareTextCommandDefault{\textdollaroldstyle}%
    {\tc@check@symbol2\textdollaroldstyle}
\DeclareTextCommandDefault{\textcentoldstyle}%
    {\tc@check@symbol2\textcentoldstyle}
\DeclareTextCommandDefault{\textcolonmonetary}%
    {\tc@check@symbol2\textcolonmonetary}
\DeclareTextCommandDefault{\textwon}%
    {\tc@check@symbol2\textwon}
\DeclareTextCommandDefault{\textnaira}%
    {\tc@check@symbol2\textnaira}
\DeclareTextCommandDefault{\textguarani}%
    {\tc@check@symbol2\textguarani}
\DeclareTextCommandDefault{\textpeso}%
    {\tc@check@symbol2\textpeso}
\DeclareTextCommandDefault{\textlira}%
    {\tc@check@symbol2\textlira}
\DeclareTextCommandDefault{\textrecipe}%
    {\tc@check@symbol2\textrecipe}
\DeclareTextCommandDefault{\textinterrobang}%
    {\tc@check@symbol2\textinterrobang}
\DeclareTextCommandDefault{\textinterrobangdown}%
    {\tc@check@symbol2\textinterrobangdown}
\DeclareTextCommandDefault{\textdong}%
    {\tc@check@symbol2\textdong}
\DeclareTextCommandDefault{\textpertenthousand}%
    {\tc@check@symbol2\textpertenthousand}
\DeclareTextCommandDefault{\textpilcrow}%
    {\tc@check@symbol2\textpilcrow}
\DeclareTextCommandDefault{\textbaht}%
    {\tc@check@symbol2\textbaht}
\DeclareTextCommandDefault{\textnumero}%
    {\tc@check@symbol2\textnumero}
\DeclareTextCommandDefault{\textdiscount}%
    {\tc@check@symbol2\textdiscount}
\DeclareTextCommandDefault{\textopenbullet}%
    {\tc@check@symbol2\textopenbullet}
\DeclareTextCommandDefault{\textservicemark}%
    {\tc@check@symbol2\textservicemark}
\DeclareTextCommandDefault{\textlquill}%
    {\tc@check@symbol2\textlquill}
\DeclareTextCommandDefault{\textrquill}%
```

```
    {\tc@check@symbol2\textrquill}
\DeclareTextCommandDefault{\textcopyleft}%
    {\tc@check@symbol2\textcopyleft}
\DeclareTextCommandDefault{\textcircledP}%
    {\tc@check@symbol2\textcircledP}
\DeclareTextCommandDefault{\textreferencemark}%
    {\tc@check@symbol2\textreferencemark}
\DeclareTextCommandDefault{\textsurd}%
    {\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.

```
\DeclareTextCommandDefault{\textcircled}
  {\CheckEncodingSubset\UseTextAccent{TS1}%
    {\UseTextAccent{OMS}}1\textcircled}
\DeclareTextCommandDefault{\t}
  {\CheckEncodingSubset\UseTextAccent{TS1}%
    {\UseTextAccent{OML}}1\t}
```

Finally input the encoding-specific definitions for `TS1` thus making the top-level definitions optimised for this encoding (and not for the default encoding).

```
\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 LaTeX which will prevent the usage of the `TS1` versions. So we better get rid of them:

```
\UndeclareTextCommand{\textsterling}{OT1}
\UndeclareTextCommand{\textdollar}  {OT1}
```

Similar declarations should probably be made for other encodings like `OT4` if they are in use.

```
%\UndeclareTextCommand{\textsterling}{OT4}
%\UndeclareTextCommand{\textdollar}  {OT4}
```

From the `T1` encoding there are two candidates for removal: ‰ and ‱ since these are both constructed from % followed by a tiny ‘₀’ 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 ‰ 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 ‰ and ‱ are not taken from the same physical font) and with PostScript fonts ‰ will come out correctly while ‱ 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`.

```
\UndeclareTextCommand{\textperthousand}{T1}
%\UndeclareTextCommand{\textpertenthousand}{T1}
```

60.1.1 Supporting oldstyle digits

```
\DeclareRobustCommand\oldstylenums[1]{%
 \begingroup
  \ifmmode
   \mathgroup\symletters #1%
  \else
   \CheckEncodingSubset\@use@text@encoding{TS1}%
       {\PackageWarning{textcomp}%
          {Oldstyle digits unavailable for
           family \f@family.\MessageBreak
           Lining digits used instead}}%
       \tw@{#1}%
   \fi
 \endgroup
}
```

60.1.2 Subset encoding defaults

For many font families commonly used in the TEX 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:

```
\iftc@forced \else
```

Computer modern based fonts (e.g., CM, CM-Bright, Concrete):

```
\DeclareEncodingSubset{TS1}{cmr}     {0}
\DeclareEncodingSubset{TS1}{cmss}    {0}
\DeclareEncodingSubset{TS1}{cmtt}    {0}
\DeclareEncodingSubset{TS1}{cmvtt}   {0}
\DeclareEncodingSubset{TS1}{cmbr}    {0}
\DeclareEncodingSubset{TS1}{cmtl}    {0}
\DeclareEncodingSubset{TS1}{ccr}     {0}
```

PSNFSS fonts:

```
\DeclareEncodingSubset{TS1}{ptm}     {4}
\DeclareEncodingSubset{TS1}{pcr}     {4}
\DeclareEncodingSubset{TS1}{phv}     {4}
\DeclareEncodingSubset{TS1}{ppl}     {3}
\DeclareEncodingSubset{TS1}{pag}     {4}
\DeclareEncodingSubset{TS1}{pbk}     {4}
\DeclareEncodingSubset{TS1}{pnc}     {4}
\DeclareEncodingSubset{TS1}{pzc}     {4}
\DeclareEncodingSubset{TS1}{bch}     {4}
\DeclareEncodingSubset{TS1}{put}     {5}
```

Other CTAN fonts (probably not complete):

```
\DeclareEncodingSubset{TS1}{uag}     {5}
\DeclareEncodingSubset{TS1}{ugq}     {5}
\DeclareEncodingSubset{TS1}{ul8}     {4}
\DeclareEncodingSubset{TS1}{ul9}     {4}  % (LuxiSans, one day)
\DeclareEncodingSubset{TS1}{augie}   {5}
\DeclareEncodingSubset{TS1}{dayrom}  {3}
\DeclareEncodingSubset{TS1}{dayroms} {3}
\DeclareEncodingSubset{TS1}{pxr}     {0}
```

```
\DeclareEncodingSubset{TS1}{pxss}    {0}
\DeclareEncodingSubset{TS1}{pxtt}    {0}
\DeclareEncodingSubset{TS1}{txr}     {0}
\DeclareEncodingSubset{TS1}{txss}    {0}
\DeclareEncodingSubset{TS1}{txtt}    {0}
```

Latin Modern and TeX Gyre:

```
\DeclareEncodingSubset{TS1}{lmr}     {0}
\DeclareEncodingSubset{TS1}{lmdh}    {0}
\DeclareEncodingSubset{TS1}{lmss}    {0}
\DeclareEncodingSubset{TS1}{lmssq}   {0}
\DeclareEncodingSubset{TS1}{lmvtt}   {0}
\DeclareEncodingSubset{TS1}{lmtt}    {0}
```

```
\DeclareEncodingSubset{TS1}{qhv}     {0}
\DeclareEncodingSubset{TS1}{qag}     {0}
\DeclareEncodingSubset{TS1}{qbk}     {0}
\DeclareEncodingSubset{TS1}{qcr}     {0}
\DeclareEncodingSubset{TS1}{qcs}     {0}
\DeclareEncodingSubset{TS1}{qpl}     {0}
\DeclareEncodingSubset{TS1}{qtm}     {0}
\DeclareEncodingSubset{TS1}{qzc}     {0}
\DeclareEncodingSubset{TS1}{qhvc}    {0}
```

Fourier-GUTenberg:

```
\DeclareEncodingSubset{TS1}{futs}    {4}
\DeclareEncodingSubset{TS1}{futx}    {4}
\DeclareEncodingSubset{TS1}{futj}    {4}
```

Y&Y's Lucida Bright

```
\DeclareEncodingSubset{TS1}{hlh}     {3}
\DeclareEncodingSubset{TS1}{hls}     {3}
\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`.

```
\DeclareEncodingSubset{TS1}{hlct}    {5}
\DeclareEncodingSubset{TS1}{hlx}     {5}
\DeclareEncodingSubset{TS1}{hlce}    {5}
\DeclareEncodingSubset{TS1}{hlcn}    {5}
\DeclareEncodingSubset{TS1}{hlcw}    {5}
\DeclareEncodingSubset{TS1}{hlcf}    {5}
```

Other commercial families...

```
\DeclareEncodingSubset{TS1}{pplx}    {3}
\DeclareEncodingSubset{TS1}{pplj}    {3}
\DeclareEncodingSubset{TS1}{ptmx}    {4}
\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.

```
\InputIfFileExists{textcomp.cfg}
  {\PackageInfo{textcomp}{Local configuration file used}}{}
```

```
\fi
⟨/TS1oldsty⟩
```

File y
ltpageno.dtx

61 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 pagenumber 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.

```
⟨*2ekernel⟩
\message{page nos.,}

\countdef\c@page=0 \c@page=1
\def\cl@page{}
\def\pagenumbering#1{%
  \global\c@page \@ne \gdef\thepage{\csname @#1\endcsname
   \c@page}}

⟨/2ekernel⟩
```

File z
ltxref.dtx

62 Cross Referencing

The user writes `\label{⟨foo⟩}` to define the following cross-references:

`\ref{⟨foo⟩}`: value of most recently incremented referencable counter. in the current environment. (Chapter, section, theorem and enumeration counters are referencable, 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.

62.1 Cross Referencing

Historical LaTeX 2.09 comments (not necessarily accurate any more):

```
⟨*2ekernel⟩
\message{x-ref,}
```

This is implemented as follows. A referencable 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 LaTeX 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 TeX 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.

```
% \newif\ifG@refundefined
% \def\G@refundefinedtrue{\global\let\ifG@refundefined\iftrue}
% \def\G@refundefinedfalse{\global\let\ifG@refundefined\iffalse}
\def\G@refundefinedtrue{%
  \gdef\@refundefined{%
    \@latex@warning@no@line{There were undefined references}}}
\let\@refundefined\relax
```

\ref
\pageref
\@setref

Referencing a \label. RmS 91/10/25: added a few extra \reset@font, as suggested by Bernd Raichle

RmS 92/08/14: made \ref and \pageref robust

RmS 93/09/08: Added setting of refundefined switch.

```
\def\@setref#1#2#3{%
  \ifx#1\relax
   \protect\G@refundefinedtrue
   \nfss@text{\reset@font\bfseries ??}%
   \@latex@warning{Reference ‘#3’ on page \thepage \space
             undefined}%
  \else
   \expandafter#2#1\null
  \fi}
\def\ref#1{\expandafter\@setref\csname r@#1\endcsname\@firstoftwo{#1}}
\def\pageref#1{\expandafter\@setref\csname r@#1\endcsname
                                    \@secondoftwo{#1}}
```

\newlabel

This command will be written to the .aux file to pass label information from one run to another.

\@newl@bel

The internal form of \newlabel and \bibcite. Note that this macro does it's 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.

```
\def\@newl@bel#1#2#3{{%
  \@ifundefined{#1@#2}%
    \relax
    {\gdef \@multiplelabels {%
       \@latex@warning@no@line{There were multiply-defined labels}}%
     \@latex@warning@no@line{Label ‘#2’ multiply defined}}%
  \global\@namedef{#1@#2}{#3}}}
\def\newlabel{\@newl@bel r}
\@onlypreamble\@newl@bel
```

\if@multiplelabels
\@multiplelabels

This is redefined to produce a warning if at least one label is defined more than once. It is executed by the \enddocument command.

```
\let \@multiplelabels \relax
```

\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.

```
\def\label#1{\@bsphack
  \protected@write\@auxout{}%
         {\string\newlabel{#1}{{\@currentlabel}{\thepage}}}%
  \@esphack}
⟨/2ekernel⟩
⟨*2ekernel | latexrelease⟩
⟨latexrelease⟩\IncludeInRelease{2019/10/01}%
⟨latexrelease⟩                  {\refstepcounter}{Add \labelformat and \Ref}%
```

\refstepcounter Step the counter and allow for labels to point to its current value.

```
\def\refstepcounter#1{\stepcounter{#1}%
    \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.

```
       {\csname p@#1\expandafter\endcsname\csname the#1\endcsname}%
}
```

\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.

```
\def\labelformat#1{\expandafter\def\csname p@#1\endcsname##1}
```

\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.[10]

```
\DeclareRobustCommand\Ref[1]{\protected@edef\@tempa{\ref{#1}}%
   \expandafter\MakeUppercase\@tempa}
```

```
⟨/2ekernel | latexrelease⟩
⟨latexrelease⟩\EndIncludeInRelease
⟨latexrelease⟩\IncludeInRelease{0000/00/00}%
⟨latexrelease⟩                 {\refstepcounter}{Add \labelformat and \Ref}%
⟨latexrelease⟩
⟨latexrelease⟩\def\refstepcounter#1{\stepcounter{#1}%
⟨latexrelease⟩    \protected@edef\@currentlabel
⟨latexrelease⟩       {\csname p@#1\endcsname\csname the#1\endcsname}%
⟨latexrelease⟩}
⟨latexrelease⟩\let\labelformat\@undefined
⟨latexrelease⟩\let\Ref\@undefined
⟨latexrelease⟩
⟨latexrelease⟩\EndIncludeInRelease
⟨*2ekernel⟩
```

\@currentlabel Default for \label commands that come before any environment.

```
\def\@currentlabel{}
```

```
⟨/2ekernel⟩
```

[10] There is one problem with this approach: the braces are kept in a normal \ref which might spoil kerning. Perhaps one day this needs redoing.

File A
ltmiscen.dtx

63 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`.

```
⟨*2ekernel⟩
\message{environments,}
```

63.1 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 TEX82.

`\enddocument` is the user’s command for ending the manuscript file.

`\stop` is a panic button — to end TEX in the middle.

Historical LATEX 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.

\@currenvir

The name of the current environment. Initialized to `document` to so that `\end{document}` works correctly.

```
\def\@currenvir{document}
```

\if@ignore
\@ignoretrue
\@ignorefalse

```
\def\@ignorefalse{\global\let\if@ignore\iffalse}
\def\@ignoretrue {\global\let\if@ignore\iftrue}
\@ignorefalse
```

\ignorespacesafterend

```
\let\ignorespacesafterend\@ignoretrue
```

\enddocument

```
\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 knowledge 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…

```
   \let\AtEndDocument\@firstofone
   \@enddocumenthook
   \@checkend{document}%
   \clearpage
   \begingroup
     \if@filesw
       \immediate\closeout\@mainaux
       \let\@setckpt\@gobbletwo
       \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`.

```
       \@tempswafalse
       \makeatletter \@@input\jobname.aux
     \fi
```

```
     \@dofilelist
```

First we check for font size substitution bigger than `\fontsubfuzz`. The `\relax` is necessary because this is a macro not a register.

```
     \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.

```
       \@font@warning{Size substitutions with differences\MessageBreak
                  up to \font@submax\space have occurred.\@gobbletwo}%
     \fi
```

The macro `\@defaultsubs` is initially `\relax` but gets redefined to produce a warning if there have been some default font substitutions.

```
     \@defaultsubs
```

The macro `\@refundefined` is initially `\relax` but gets redefined to produce a warning if there are undefined refs.

```
     \@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.

```
     \if@filesw
       \ifx \@multiplelabels \relax
         \if@tempswa
           \@latex@warning@no@line{Label(s) may have changed.
               Rerun to get cross-references right}%
         \fi
       \else
         \@multiplelabels
       \fi
     \fi
   \endgroup
   \deadcycles\z@\@@end}
```

`\@testdef`

```
\def\@testdef #1#2#3{%
  \def\reserved@a{#3}\expandafter \ifx \csname #1@#2\endcsname
 \reserved@a  \else \@tempswatrue \fi}
```

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.

```
⟨/2ekernel⟩
⟨*2ekernel | latexrelease⟩
⟨latexrelease⟩\IncludeInRelease{2018/12/01}%
⟨latexrelease⟩                 {\protected@file@percent}{Mask line endings}%
\protected\def\protected@file@percent{}
```

`\add@percent@to@temptokena` Helper function which is used to inspect a sequence of tokens (the second argument of `\@writefile` and it the last token is `\protected@file@percent` it will replace it by a harmless percent. The result is saved in `\@temptokena` for later use.

```
\catcode`\^^A=9
\long\gdef\add@percent@to@temptokena
    #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.

```
    {\expandafter\ifx\expandafter X\detokenize{#2}X\expandafter\dont@add@percent@to@temptoken
            \expandafter\do@add@percent@to@temptokena\fi{#1}}
```

```
\long\def\dont@add@percent@to@temptokena#1{%
  \@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 :-)

```
\begingroup
\catcode`\%=12
\catcode`\^^A=14
\long\gdef\do@add@percent@to@temptokena#1{\@temptokena\expandafter{#1%^^A
```

Can't be on the same line as the `%` — see above.

```
  }}
\endgroup
```

`\@writefile`

```
\long\def\@writefile#1#2{%
  \@ifundefined{tf@#1}\relax
    {%
```

If we write to the file we first prepare `#2` using `\add@percent@to@temptokena` and then write the token register out.

```
      \add@percent@to@temptokena
        \@empty#2\protected@file@percent
        \add@percent@to@temptokena
     \immediate\write\csname tf@#1\endcsname{\the\@temptokena}%
    }%
}
⟨/2ekernel | latexrelease⟩
⟨latexrelease⟩\EndIncludeInRelease
⟨latexrelease⟩\IncludeInRelease{0000/00/00}%
⟨latexrelease⟩                 {\protected@file@percent}{Mask line endings}%
⟨latexrelease⟩\let\protected@file@percent\@undefined
⟨latexrelease⟩\let\add@percent@to@temptokena\@undefined
⟨latexrelease⟩\let\do@add@percent@to@temptokena\@undefined
⟨latexrelease⟩\let\dont@add@percent@to@temptokena\@undefined
⟨latexrelease⟩\long\def\@writefile#1#2{%
⟨latexrelease⟩  \@ifundefined{tf@#1}\relax
⟨latexrelease⟩    {\@temptokena{#2}%
⟨latexrelease⟩     \immediate\write\csname tf@#1\endcsname{\the\@temptokena}%
⟨latexrelease⟩    }%
⟨latexrelease⟩}
⟨latexrelease⟩\EndIncludeInRelease
⟨*2ekernel⟩
```

`\stop`

```
\def\stop{\clearpage\deadcycles\z@\let\par\@@par\@@end}
```

Historical LaTeX 2.09 comments (not necessarily accurate any more):

```
\everypar{\@nodocument} %% To get an error if text appears before the
```

```
\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 ==
                                  (\@currenvir :=L NAME) \NAME
   FI
   @ignore :=G F       %% Added 30 Nov 88
   \begingroup
   \@endpe := F
   \@currenvir :=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
  FI                            %% immediately following text
  IF @ignore = T
    THEN @ignore :=G F
         \ignorespaces
  FI
 END

\@checkend{NAME} ==
 BEGIN
  IF \@currenvir = NAME
    ELSE \@badend{NAME}
  FI
 END
```

End of historical LaTeX 2.09 comments.

\begin

```
⟨/2ekernel⟩
⟨*2ekernel | latexrelease⟩
⟨latexrelease⟩\IncludeInRelease{2019/10/01}%
⟨latexrelease⟩                 {\begin}{Making \begin/\end robust}%
\DeclareRobustCommand\begin[1]{%
\@ifundefined{#1}%
  {\def\reserved@a{\@latex@error{Environment #1 undefined}\@eha}}%
  {\def\reserved@a{\def\@currenvir{#1}%
   \edef\@currenvline{\on@line}%
   \csname #1\endcsname}}%
\@ignorefalse
\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.

```
%\edef\begin
%  {\unexpanded{%
%       \ifx\protect\@typeset@protect
%         \expandafter\@gobble
%       \fi
%       \protect
%     }%
%   \expandafter\noexpand\csname begin \endcsname
%  }
%\@namedef{begin }#1{%
%  \@ifundefined{#1}%
%    {\def\reserved@a{\@latex@error{Environment #1 undefined}\@eha}}%
%    {\def\reserved@a{\def\@currenvir{#1}%
%     \edef\@currenvline{\on@line}%
%     \csname #1\endcsname}}%
%  \@ignorefalse
%  \begingroup\@endpefalse\reserved@a}
```

`\end` 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 `\expandafter`s, the first to get

```
\csname endfoo\endcsname       \@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`s inside that are all going to be executed when `\romannumeral` is hit by a single `\expandafter`.

```
\edef\end
  {\unexpanded{%
     \romannumeral
       \ifx\protect\@typeset@protect
       \expandafter       %1
         \expandafter        %2
       \expandafter       %1
           \expandafter         %3 expands the \csname inside \end<space>
       \expandafter       %1
         \expandafter        %2  expands \end<space>
       \expandafter       %1     expands the \else
           \z@
       \else
         \expandafter\z@\expandafter\protect
       \fi
   }%
   \expandafter\noexpand\csname end \endcsname
  }
```

And here is the original definition of `\end` the way it was in LaTeX for several decades now hidden in `\end␣`.

```
\@namedef{end }#1{%
  \csname end#1\endcsname\@checkend{#1}%
  \expandafter\endgroup\if@endpe\@doendpe\fi
  \if@ignore\@ignorefalse\ignorespaces\fi}
⟨/2ekernel | latexrelease⟩
⟨latexrelease⟩\EndIncludeInRelease
```

An here the rollback in case that is ever needed.

```
⟨latexrelease⟩\IncludeInRelease{0000/00/00}%
⟨latexrelease⟩                 {\begin}{Making \begin/\end robust}%
⟨latexrelease⟩\kernel@make@fragile\begin
⟨latexrelease⟩\kernel@make@fragile\end
⟨latexrelease⟩
⟨latexrelease⟩\EndIncludeInRelease
⟨*2ekernel⟩
```

`\@checkend`

```
\def\@checkend#1{\def\reserved@a{#1}\ifx
      \reserved@a\@currenvir \else\@badend{#1}\fi}
```

`\@currenvline` We do need a default value for `\@currenvline` 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.

```
\let\@currenvline\@empty
```

63.2 Center, Flushright, Flushleft

```
\message{center,}
```

Historical LaTeX 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 LaTeX 2.09 comments.

\@centercr

```
⟨/2ekernel⟩
⟨*2ekernel | latexrelease⟩
⟨latexrelease⟩\IncludeInRelease{2020/02/02}%
⟨latexrelease⟩                  {\@centercr}{Make robust}%
\protected\def\@centercr{\ifhmode \unskip\else \@nolnerr\fi
       \par\@ifstar{\nobreak\@xcentercr}\@xcentercr}
⟨/2ekernel | latexrelease⟩
⟨latexrelease⟩\EndIncludeInRelease
⟨latexrelease⟩\IncludeInRelease{0000/00/00}%
⟨latexrelease⟩                  {\@centercr}{Make robust}%
⟨latexrelease⟩
⟨latexrelease⟩\def\@centercr{\ifhmode \unskip\else \@nolnerr\fi
⟨latexrelease⟩       \par\@ifstar{\nobreak\@xcentercr}\@xcentercr}
⟨latexrelease⟩
⟨latexrelease⟩\EndIncludeInRelease
⟨*2ekernel⟩
```

\@xcentercr

```
\def\@xcentercr{\addvspace{-\parskip}\@ifnextchar
    [\@icentercr\ignorespaces}
```

\@icentercr

```
\def\@icentercr[#1]{\vskip #1\ignorespaces}
```

center We use `\relax` to prevent `\item` scanning too far.

```
\def\center{\trivlist \centering\item\relax}
```

```
\def\endcenter{\endtrivlist}
```

```
⟨/2ekernel⟩
⟨*2ekernel | latexrelease⟩
⟨latexrelease⟩\IncludeInRelease{2019/10/01}%
⟨latexrelease⟩                  {\centering}{Make commands robust}%
```

\centering

```
\DeclareRobustCommand\centering{%
  \let\\\@centercr
  \rightskip\@flushglue\leftskip\@flushglue
  \parindent\z@\parfillskip\z@skip}
```

\raggedright

```
\DeclareRobustCommand\raggedright{%
  \let\\\@centercr\@rightskip\@flushglue \rightskip\@rightskip
  \leftskip\z@skip
  \parindent\z@}
```

\raggedleft

```
\DeclareRobustCommand\raggedleft{%
  \let\\\@centercr
  \rightskip\z@skip\leftskip\@flushglue
  \parindent\z@\parfillskip\z@skip}
```

```
⟨/2ekernel | latexrelease⟩
⟨latexrelease⟩\EndIncludeInRelease
⟨latexrelease⟩\IncludeInRelease{0000/00/00}%
⟨latexrelease⟩                  {\centering}{Make commands robust}%
⟨latexrelease⟩
⟨latexrelease⟩\kernel@make@fragile\centering
⟨latexrelease⟩\kernel@make@fragile\raggedright
⟨latexrelease⟩\kernel@make@fragile\raggedleft
⟨latexrelease⟩
⟨latexrelease⟩\EndIncludeInRelease
⟨*2ekernel⟩
```

\@rightskip

```
\newskip\@rightskip \@rightskip \z@skip
```

flushleft We use `\relax` to prevent `\item` scanning too far.

```
\def\flushleft{\trivlist \raggedright\item\relax}
```

```
\def\endflushleft{\endtrivlist}
```

flushright We use `\relax` to prevent `\item` scanning too far.

```
\def\flushright{\trivlist \raggedleft\item\relax}
```

```
\def\endflushright{\endtrivlist}
```

63.3 Verbatim

```
\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`

```
{\catcode`\ =\active%
\gdef\@vobeyspaces{\catcode`\ \active\let \@xobeysp}}
```

`\@xobeysp`

`\@xverbatim`
`\@sxverbatim`

```
\begingroup \catcode `|=0 \catcode `[= 1
\catcode`]=2 \catcode `\{=12 \catcode `\}=12
\catcode`\\=12 |gdef|@xverbatim#1\end{verbatim}[#1|end[verbatim]]
|gdef|@sxverbatim#1\end{verbatim*}[#1|end[verbatim*]]
|endgroup
```

`\@verbatim` Real start of verbatim environment We use `\relax` to prevent `\item` scanning too far.

```
⟨/2ekernel⟩
⟨*2ekernel | latexrelease⟩
⟨latexrelease⟩\IncludeInRelease{2017-04-15}{\@verbatim}%
⟨latexrelease⟩                    {Disable hyphenation in verbatim}%
\def\@verbatim{\trivlist \item\relax
  \if@minipage\else\vskip\parskip\fi
  \leftskip\@totalleftmargin\rightskip\z@skip
  \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.

```
  \@@par
```

Set `\language` here to suppress hyphenation. Done this way rather than setting `\hyphenchar` as that is a global setting.

```
  \language\l@nohyphenation
  \@tempswafalse
  \def\par{%
    \if@tempswa
```

A `\leavevmode` added: needed if, for example, a blank verbatim line is the first thing in a list item (wow!).

```
      \leavevmode \null \@@par\penalty\interlinepenalty
    \else
      \@tempswatrue
      \ifhmode\@@par\penalty\interlinepenalty\fi
    \fi}%
```

To allow customization we hide the font used in a separate macro.

```
  \let\do\@makeother \dospecials
  \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`!

```
  \everypar \expandafter{\the\everypar \unpenalty}%
}
⟨/2ekernel | latexrelease⟩
⟨latexrelease⟩\EndIncludeInRelease
⟨latexrelease⟩\IncludeInRelease{0000-00-00}{\@verbatim}%
⟨latexrelease⟩                  {Disable hyphenation in verbatim}%
⟨latexrelease⟩\def\@verbatim{\trivlist \item\relax
⟨latexrelease⟩  \if@minipage\else\vskip\parskip\fi
⟨latexrelease⟩  \leftskip\@totalleftmargin\rightskip\z@skip
⟨latexrelease⟩  \parindent\z@\parfillskip\@flushglue\parskip\z@skip
⟨latexrelease⟩  \@@par
⟨latexrelease⟩  \@tempswafalse
⟨latexrelease⟩  \def\par{%
⟨latexrelease⟩    \if@tempswa
⟨latexrelease⟩      \leavevmode \null \@@par\penalty\interlinepenalty
⟨latexrelease⟩    \else
⟨latexrelease⟩      \@tempswatrue
⟨latexrelease⟩      \ifhmode\@@par\penalty\interlinepenalty\fi
⟨latexrelease⟩    \fi}%
⟨latexrelease⟩  \let\do\@makeother \dospecials
⟨latexrelease⟩  \obeylines \verbatim@font \@noligs
⟨latexrelease⟩  \hyphenchar\font\m@ne
⟨latexrelease⟩  \everypar \expandafter{\the\everypar \unpenalty}%
⟨latexrelease⟩}
⟨latexrelease⟩\EndIncludeInRelease
⟨*2ekernel⟩
```

`\verbatim` `\endverbatim` (RmS 93/09/19) Protected against 'missing item' error message triggered by empty verbatim environment.

```
\def\verbatim{\@verbatim \frenchspacing\@vobeyspaces \@xverbatim}
\def\endverbatim{\if@newlist \leavevmode\fi\endtrivlist}
```

`\verbatim@font` Macro to select the font used for verbatim typesetting. It also does other work if necessary for the font used.

```
\def\verbatim@font{\normalfont\ttfamily}
```

```
⟨/2ekernel⟩
⟨*2ekernel | latexrelease⟩
⟨latexrelease⟩\IncludeInRelease{2018/12/01}%
⟨latexrelease⟩                  {\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

```
\DeclareRobustCommand\asciispace{\char 32 }
```

`\verbvisiblespace` This defines how to get a visible space in `\verb*` and friends. In classic TEX this is just the slot 32, but in TU encoded fonts we switch fonts and take the character from cmtt.

```
\ifx\Umathcode\@undefined
  \let\verbvisiblespace\asciispace                                % Pdftex version
\else
  \DeclareRobustCommand\verbvisiblespace
           {\leavevmode{\usefont{OT1}{cmtt}{m}{n}\asciispace}}    % xetex/luatex version
\fi
```

\@setupverbvisiblespace In pdfTEX 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 XeTEX or LuaTEX a font in TU 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 pdfTEX). We then redefine \@xobeysp so that after running \@vobeyspaces we get characters from slot 32 for each active space.

```
\def\@setupverbvisiblespace{%
  \ifx\verbvisiblespace\asciispace
    \let\@xobeysp\asciispace
  \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.

```
    \setbox\z@\hbox{x}%
    \setbox\@verbvisiblespacebox\hbox to\wd\z@{\hss\verbvisiblespace\hss}%
    \def\@xobeysp{\leavevmode\copy\@verbvisiblespacebox}%
  \fi
}
```

\@verbvisiblespacebox The box to hold the visible space character if it isn't in slot 32 in the current typewriter font.

```
\newbox\@verbvisiblespacebox
```

\@sverb Definitions of \@sverb and \@verb changed so \verb+ foo+ does not lose leading blanks when it comes at the beginning of a line. Change made 24 May 89. Suggested by Frank Mittelbach and Rainer Schöpf.

```
\def\@sverb#1{%
  \catcode`#1\active
  \lccode`\~`#1%
  \gdef\verb@balance@group{\verb@egroup
     \@latex@error{\noexpand\verb illegal in command argument}\@ehc}%
  \aftergroup\verb@balance@group
  \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.

```
  \ifnum\catcode`\ =\active
  \else  \@setupverbvisiblespace \@vobeyspaces \fi
}
```

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.

```
\@namedef{verbatim*}{\@verbatim
  \@setupverbvisiblespace
  \frenchspacing\@vobeyspaces\@sxverbatim}
\expandafter\let\csname endverbatim*\endcsname =\endverbatim
```

```
⟨/2ekernel | latexrelease⟩
⟨latexrelease⟩\EndIncludeInRelease
⟨latexrelease⟩\IncludeInRelease{0000/00/00}%
⟨latexrelease⟩                 {\verbvisiblespace}{Setup visible space for verb}%
⟨latexrelease⟩
⟨latexrelease⟩\@namedef{verbatim*}{\@verbatim\@sxverbatim}
⟨latexrelease⟩
⟨latexrelease⟩\let\asciispace            \@undefined
⟨latexrelease⟩\let\verbvisiblespace      \@undefined
⟨latexrelease⟩\let\@setupverbvisiblespace\@undefined
⟨latexrelease⟩\let\@verbvisiblespacebox  \@undefined
⟨latexrelease⟩
⟨latexrelease⟩\def\@sverb#1{%
⟨latexrelease⟩  \catcode`#1\active
⟨latexrelease⟩  \lccode`\~`#1%
⟨latexrelease⟩  \gdef\verb@balance@group{\verb@egroup
⟨latexrelease⟩     \@latex@error{\noexpand\verb illegal in command argument}\@ehc}%
⟨latexrelease⟩  \aftergroup\verb@balance@group
⟨latexrelease⟩  \lowercase{\let~\verb@egroup}}%
⟨latexrelease⟩
⟨latexrelease⟩\EndIncludeInRelease
⟨*2ekernel⟩
```

\@makeother

```
\def\@makeother#1{\catcode`#112\relax}
```

\verb@balance@group

```
\let\verb@balance@group\@empty
```

\verb@egroup

```
\def\verb@egroup{\global\let\verb@balance@group\@empty\egroup}
```

\verb@eol@error

```
\begingroup
  \obeylines%
  \gdef\verb@eol@error{\obeylines%
    \def^^M{\verb@egroup\@latex@error{%
              \noexpand\verb ended by end of line}\@ehc}}%
\endgroup
```

\verb Typesetting a small piece verbatim.

```
⟨/2ekernel⟩
⟨*2ekernel | latexrelease⟩
⟨latexrelease⟩\IncludeInRelease{2017-04-15}{\verb}%
```

```
⟨latexrelease⟩                  {Disable hyphenation in verb}%
\def\verb{\relax\ifmmode\hbox\else\leavevmode\null\fi
  \bgroup
    \verb@eol@error \let\do\@makeother \dospecials
    \verbatim@font\@noligs
```

Set \language here to suppress hyphenation. Done this way rather than setting \hyphenchar as that is a global setting.

```
    \language\l@nohyphenation
    \@ifstar\@sverb\@verb}
⟨/2ekernel | latexrelease⟩
⟨latexrelease⟩\EndIncludeInRelease
⟨latexrelease⟩\IncludeInRelease{0000-00-00}{\verb}%
⟨latexrelease⟩                  {Disable hyphenation in verb}%
⟨latexrelease⟩\def\verb{\relax\ifmmode\hbox\else\leavevmode\null\fi
⟨latexrelease⟩  \bgroup
⟨latexrelease⟩    \verb@eol@error \let\do\@makeother \dospecials
⟨latexrelease⟩    \verbatim@font\@noligs
⟨latexrelease⟩    \@ifstar\@sverb\@verb}
⟨latexrelease⟩\EndIncludeInRelease
⟨*2ekernel⟩
```

\@verb

```
\def\@verb{\@vobeyspaces \frenchspacing \@sverb}
```

\verbatim@nolig@list

```
\def\verbatim@nolig@list{\do\`\do\<\do\>\do\,\do\'\do\-}
```

\do@noligs

```
\def\do@noligs#1{%
  \catcode`#1\active
  \begingroup
     \lccode`\~`#1\relax
     \lowercase{\endgroup\def~{\leavevmode\kern\z@\char`#1}}}
```

\@noligs To stay compatible with packages that use \@noligs we keep it.

```
\def\@noligs{\let\do\do@noligs \verbatim@nolig@list}
```

```
⟨/2ekernel⟩
```

File B
ltmath.dtx

64 Math setup

This file contains a lot of the original plain TEX code, as well as the LATEX environments for math. It still needs sorting out.

```
⟨*2ekernel⟩
\message{math definitions,}
```

64.1 Math commands based on plain TEX

64.1.1 The log-like functions

\log The standard operators:

```
\DeclareRobustCommand\log{\mathop{\operator@font log}\nolimits}
\DeclareRobustCommand\lg{\mathop{\operator@font lg}\nolimits}
\DeclareRobustCommand\ln{\mathop{\operator@font ln}\nolimits}
\DeclareRobustCommand\lim{\mathop{\operator@font lim}}
\DeclareRobustCommand\limsup{\mathop{\operator@font lim\,sup}}
\DeclareRobustCommand\liminf{\mathop{\operator@font lim\,inf}}
\DeclareRobustCommand\sin{\mathop{\operator@font sin}\nolimits}
\DeclareRobustCommand\arcsin{\mathop{\operator@font arcsin}\nolimits}
\DeclareRobustCommand\sinh{\mathop{\operator@font sinh}\nolimits}
\DeclareRobustCommand\cos{\mathop{\operator@font cos}\nolimits}
\DeclareRobustCommand\arccos{\mathop{\operator@font arccos}\nolimits}
\DeclareRobustCommand\cosh{\mathop{\operator@font cosh}\nolimits}
\DeclareRobustCommand\tan{\mathop{\operator@font tan}\nolimits}
\DeclareRobustCommand\arctan{\mathop{\operator@font arctan}\nolimits}
\DeclareRobustCommand\tanh{\mathop{\operator@font tanh}\nolimits}
\DeclareRobustCommand\cot{\mathop{\operator@font cot}\nolimits}
\DeclareRobustCommand\coth{\mathop{\operator@font coth}\nolimits}
\DeclareRobustCommand\sec{\mathop{\operator@font sec}\nolimits}
\DeclareRobustCommand\csc{\mathop{\operator@font csc}\nolimits}
\DeclareRobustCommand\max{\mathop{\operator@font max}}
\DeclareRobustCommand\min{\mathop{\operator@font min}}
\DeclareRobustCommand\sup{\mathop{\operator@font sup}}
\DeclareRobustCommand\inf{\mathop{\operator@font inf}}
\DeclareRobustCommand\arg{\mathop{\operator@font arg}\nolimits}
\DeclareRobustCommand\ker{\mathop{\operator@font ker}\nolimits}
\DeclareRobustCommand\dim{\mathop{\operator@font dim}\nolimits}
\DeclareRobustCommand\hom{\mathop{\operator@font hom}\nolimits}
\DeclareRobustCommand\det{\mathop{\operator@font det}}
\DeclareRobustCommand\exp{\mathop{\operator@font exp}\nolimits}
\DeclareRobustCommand\Pr{\mathop{\operator@font Pr}}
\DeclareRobustCommand\gcd{\mathop{\operator@font gcd}}
\DeclareRobustCommand\deg{\mathop{\operator@font deg}\nolimits}
```

\bmod And some operators have to be done by hand:

```
\DeclareRobustCommand\bmod{%
  \nonscript\mskip-\medmuskip\mkern5mu%
  \mathbin{\operator@font mod}\penalty900\mkern5mu%
  \nonscript\mskip-\medmuskip}
```

\pmod

```
\DeclareRobustCommand\pmod[1]{%
  \allowbreak\mkern18mu({\operator@font mod}\,\,#1)}
```

64.1.2 Bigggggg

\big Variants on \big and friends for use with delimiters:

```
\DeclareRobustCommand\bigl{\mathopen\big}
\DeclareRobustCommand\bigm{\mathrel\big}
\DeclareRobustCommand\bigr{\mathclose\big}
\DeclareRobustCommand\Bigl{\mathopen\Big}
\DeclareRobustCommand\Bigm{\mathrel\Big}
\DeclareRobustCommand\Bigr{\mathclose\Big}
\DeclareRobustCommand\biggl{\mathopen\bigg}
\DeclareRobustCommand\biggm{\mathrel\bigg}
\DeclareRobustCommand\biggr{\mathclose\bigg}
\DeclareRobustCommand\Biggl{\mathopen\Bigg}
\DeclareRobustCommand\Biggm{\mathrel\Bigg}
\DeclareRobustCommand\Biggr{\mathclose\Bigg}
```

64.1.3 The UNSORTED Rest

The other math commands are lifted from plain TEX.

\jot

```
\newdimen\jot
\jot=3pt
```

\interdisplaylinepenalty

```
\newcount\interdisplaylinepenalty
\interdisplaylinepenalty=100
```

\choose

```
\def\choose{\atopwithdelims()}
```

\brack

```
\def\brack{\atopwithdelims[]}
```

\brace

```
\def\brace{\atopwithdelims\{\}}
```

\mathpalette

```
\def\mathpalette#1#2{%
  \mathchoice
    {#1\displaystyle{#2}}%
    {#1\textstyle{#2}}%
    {#1\scriptstyle{#2}}%
    {#1\scriptscriptstyle{#2}}}
```

\root
\rootbox
\r@@t

```
\newbox\rootbox
\def\root#1\of{%
  \setbox\rootbox\hbox{$\m@th\scriptscriptstyle{#1}$}%
  \mathpalette\r@@t}
\def\r@@t#1#2{%
  \setbox\z@\hbox{$\m@th#1\sqrtsign{#2}$}%
  \dimen@\ht\z@ \advance\dimen@-\dp\z@
  \mkern5mu\raise.6\dimen@\copy\rootbox
  \mkern-10mu\box\z@}
```

\phantom
\hphantom
\vphantom

```
\newif\ifv@
\newif\ifh@
⟨/2ekernel⟩
⟨*2ekernel | latexrelease⟩
⟨latexrelease⟩\IncludeInRelease{2019/10/01}%
⟨latexrelease⟩                  {\vphantom}{Make commands robust}%
\DeclareRobustCommand\vphantom{\v@true\h@false\ph@nt}
\DeclareRobustCommand\hphantom{\v@false\h@true\ph@nt}
\DeclareRobustCommand\phantom{\v@true\h@true\ph@nt}
```

\mathstrut

```
\DeclareRobustCommand\mathstrut{\vphantom(}
⟨/2ekernel | latexrelease⟩
⟨latexrelease⟩\EndIncludeInRelease
⟨latexrelease⟩\IncludeInRelease{0000/00/00}%
⟨latexrelease⟩                  {\vphantom}{Make commands robust}%
⟨latexrelease⟩
⟨latexrelease⟩\kernel@make@fragile\vphantom
⟨latexrelease⟩\kernel@make@fragile\hphantom
⟨latexrelease⟩\kernel@make@fragile\phantom
⟨latexrelease⟩\kernel@make@fragile\mathstrut
⟨latexrelease⟩
⟨latexrelease⟩\EndIncludeInRelease
⟨*2ekernel⟩
\def\ph@nt{%
  \ifmmode
    \expandafter\mathpalette\expandafter\mathph@nt
  \else
    \expandafter\makeph@nt
  \fi}
\def\makeph@nt#1{%
  \setbox\z@\hbox{\color@begingroup#1\color@endgroup}\finph@nt}
\def\mathph@nt#1#2{%
  \setbox\z@\hbox{$\m@th#1{#2}$}\finph@nt}
```

```
⟨/2ekernel⟩
⟨*2ekernel | latexrelease⟩
⟨latexrelease⟩\IncludeInRelease{2018/12/01}%
⟨latexrelease⟩                 {\finph@nt}{Start LR-mode}%
\def\finph@nt{%
  \setbox\tw@\null
  \ifv@ \ht\tw@\ht\z@ \dp\tw@\dp\z@\fi
  \ifh@ \wd\tw@\wd\z@\fi
```

```
  \leavevmode@ifvmode\box\tw@}
⟨/2ekernel | latexrelease⟩
⟨latexrelease⟩\EndIncludeInRelease
⟨latexrelease⟩\IncludeInRelease{0000/00/00}%
⟨latexrelease⟩                 {\finph@nt}{Start LR-mode}%
⟨latexrelease⟩\def\finph@nt{%
⟨latexrelease⟩  \setbox\tw@\null
⟨latexrelease⟩  \ifv@ \ht\tw@\ht\z@ \dp\tw@\dp\z@\fi
⟨latexrelease⟩  \ifh@ \wd\tw@\wd\z@\fi \box\tw@}
⟨latexrelease⟩\EndIncludeInRelease
⟨*2ekernel⟩
```

`\smash`

```
\DeclareRobustCommand\smash{%
  \relax % \relax, in case this comes first in \halign
  \ifmmode
    \expandafter\mathpalette\expandafter\mathsm@sh
  \else
    \expandafter\makesm@sh
  \fi}
```

```
\def\makesm@sh#1{%
  \setbox\z@\hbox{\color@begingroup#1\color@endgroup}\finsm@sh}
\def\mathsm@sh#1#2{%
  \setbox\z@\hbox{$\m@th#1{#2}$}\finsm@sh}
```

```
⟨/2ekernel⟩
⟨*2ekernel | latexrelease⟩
⟨latexrelease⟩\IncludeInRelease{2018/12/01}%
⟨latexrelease⟩                 {\finsm@sh}{Start LR-mode}%
\def\finsm@sh{\ht\z@\z@ \dp\z@\z@ \leavevmode@ifvmode\box\z@}
⟨/2ekernel | latexrelease⟩
⟨latexrelease⟩\EndIncludeInRelease
⟨latexrelease⟩\IncludeInRelease{0000/00/00}%
⟨latexrelease⟩                 {\finsm@sh}{Start LR-mode}%
⟨latexrelease⟩\def\finsm@sh{\ht\z@\z@ \dp\z@\z@ \box\z@}
⟨latexrelease⟩\EndIncludeInRelease
⟨*2ekernel⟩
```

`\buildrel`

```
\def\buildrel#1\over#2{\mathrel{\mathop{\kern\z@#2}\limits^{#1}}}
```

```
⟨/2ekernel⟩
⟨*2ekernel | latexrelease⟩
⟨latexrelease⟩\IncludeInRelease{2019/10/01}%
⟨latexrelease⟩                 {\cases}{Make commands robust}%
```

\cases

```
\DeclareRobustCommand\cases[1]{\left\{\,\vcenter{\normalbaselines\m@th
    \ialign{$##\hfil$&\quad{##}\hfil\crcr#1\crcr}}\right.}
```

\matrix

```
\DeclareRobustCommand\matrix[1]{\null\,\vcenter{\normalbaselines\m@th
    \ialign{\hfil$##$\hfil&&\quad\hfil$##$\hfil\crcr
      \mathstrut\crcr\noalign{\kern-\baselineskip}
      #1\crcr\mathstrut\crcr\noalign{\kern-\baselineskip}}}\,}
```

\pmatrix

```
\DeclareRobustCommand\pmatrix[1]{\left(\matrix{#1}\right)}
```

```
⟨/2ekernel | latexrelease⟩
⟨latexrelease⟩\EndIncludeInRelease
⟨latexrelease⟩\IncludeInRelease{0000/00/00}%
⟨latexrelease⟩                 {\cases}{Make commands robust}%
⟨latexrelease⟩
⟨latexrelease⟩\kernel@make@fragile\cases
⟨latexrelease⟩\kernel@make@fragile\matrix
⟨latexrelease⟩\kernel@make@fragile\pmatrix
⟨latexrelease⟩
⟨latexrelease⟩\EndIncludeInRelease
⟨*2ekernel⟩
```

\bordermatrix

```
\def\bordermatrix#1{\begingroup \m@th
  \@tempdima 8.75\p@
  \setbox\z@\vbox{%
    \def\cr{\crcr\noalign{\kern2\p@\global\let\cr\endline}}%
    \ialign{$##$\hfil\kern2\p@\kern\@tempdima&\thinspace\hfil$##$\hfil
      &&\quad\hfil$##$\hfil\crcr
      \omit\strut\hfil\crcr\noalign{\kern-\baselineskip}%
      #1\crcr\omit\strut\cr}}%
  \setbox\tw@\vbox{\unvcopy\z@\global\setbox\@ne\lastbox}%
  \setbox\tw@\hbox{\unhbox\@ne\unskip\global\setbox\@ne\lastbox}%
  \setbox\tw@\hbox{$\kern\wd\@ne\kern-\@tempdima\left(\kern-\wd\@ne
    \global\setbox\@ne\vbox{\box\@ne\kern2\p@}%
    \vcenter{\kern-\ht\@ne\unvbox\z@\kern-\baselineskip}\,\right)$}%
  \null\;\vbox{\kern\ht\@ne\box\tw@}\endgroup}
```

\openup

```
\def\openup{\afterassignment\@penup\dimen@}
```

```
\def\@penup{\advance\lineskip\dimen@
  \advance\baselineskip\dimen@
  \advance\lineskiplimit\dimen@}
```

\displaylines

```
\newif\ifdt@p
```

```
\def\displ@y{\global\dt@ptrue\openup\jot\m@th
  \everycr{\noalign{\ifdt@p \global\dt@pfalse \ifdim\prevdepth>-1000\p@
      \vskip-\lineskiplimit \vskip\normallineskiplimit \fi
      \else \penalty\interdisplaylinepenalty \fi}}}
```

```
\def\@lign{\tabskip\z@skip\everycr{}} % restore inside \displ@y
```

```
\def\displaylines#1{\displ@y \tabskip\z@skip
  \halign{\hb@xt@\displaywidth{$\@lign\hfil\displaystyle##\hfil$}\crcr
    #1\crcr}}
```

\sp
\sb

```
\let\sp=^
\let\sb=_
```

\>
\;
\!

```
%\def\,{\mskip\thinmuskip}      % already defined in ltspace
\def\>{\mskip\medmuskip}
\def\;{\mskip\thickmuskip}
\def\!{\mskip-\thinmuskip}
```

*

```
\DeclareRobustCommand\*{\discretionary{\thinspace\the\textfont2\char2}{}{}}
```

\: Nickname for the medium space since \> is not available inside tabbing.

```
\let\:=\>
```

\active@math@prime This is the definition of the active math prime.

```
\def\active@math@prime{^\bgroup\prim@s}
```

\prime@s

```
{\catcode`\'=\active \global\let'\active@math@prime}
```

```
\def\prim@s{%
  \prime\futurelet\@let@token\pr@m@s}
```

```
\def\pr@m@s{%
  \ifx'\@let@token
    \expandafter\pr@@@s
  \else
    \ifx^\@let@token
      \expandafter\expandafter\expandafter\pr@@@t
    \else
      \egroup
    \fi
  \fi}
```

```
\def\pr@@@s#1{\prim@s}
```

```
\def\pr@@@t#1#2{#2\egroup}
```

```
{\catcode`\_=\active \gdef_{\_}} % _ in math is
                                 % either subscript or \_
```

64.2 Math Environments

\(
\)
Produces `$...$` with checks that `\(` isn't used in math mode, and that `\)` is only used in math mode begun with `\(`.

```
⟨/2ekernel⟩
⟨latexrelease⟩\IncludeInRelease{2015/01/01}{\(}{Make \( robust}%
⟨*2ekernel | latexrelease⟩
\DeclareRobustCommand\({%
  \relax\ifmmode\@badmath\else$\fi}%
\DeclareRobustCommand\){%
  \relax\ifmmode\ifinner$\else\@badmath\fi\else \@badmath\fi}%
⟨/2ekernel | latexrelease⟩
⟨latexrelease⟩\EndIncludeInRelease
⟨latexrelease⟩\IncludeInRelease{0000/00/00}{\(}{Make \( robust}%
⟨latexrelease⟩\def\({%
⟨latexrelease⟩  \relax\ifmmode\@badmath\else$\fi}%
⟨latexrelease⟩\expandafter\let\csname\string( \endcsname\@undefined
⟨latexrelease⟩\def\){%
⟨latexrelease⟩  \relax\ifmmode\ifinner$\else\@badmath\fi\else \@badmath\fi}%
⟨latexrelease⟩\expandafter\let\csname\string) \endcsname\@undefined
⟨latexrelease⟩\EndIncludeInRelease
⟨*2ekernel⟩
```

\[
\]
Produces `$$...$$` 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 `$$`).

```
⟨/2ekernel⟩
⟨latexrelease⟩\IncludeInRelease{2015/01/01}{\[}{Make \[ robust}%
⟨*2ekernel | latexrelease⟩
\DeclareRobustCommand\[{%
   \relax\ifmmode
      \@badmath
   \else
      \ifvmode
         \nointerlineskip
         \makebox[.6\linewidth]{}%
      \fi
      $$%%$$ BRACE MATCH HACK
   \fi
}%

\DeclareRobustCommand\]{%
   \relax\ifmmode
      \ifinner
         \@badmath
      \else
         $$%%$$ BRACE MATCH HACK
      \fi
   \else
      \@badmath
   \fi
   \ignorespaces
}%

⟨/2ekernel | latexrelease⟩
```

```
⟨latexrelease⟩\EndIncludeInRelease
⟨latexrelease⟩\IncludeInRelease{0000/00/00}{\[}{Make \[ robust}%
⟨latexrelease⟩\def\[{%
⟨latexrelease⟩   \relax\ifmmode
⟨latexrelease⟩      \@badmath
⟨latexrelease⟩   \else
⟨latexrelease⟩      \ifvmode
⟨latexrelease⟩         \nointerlineskip
⟨latexrelease⟩         \makebox[.6\linewidth]{}%
⟨latexrelease⟩      \fi
⟨latexrelease⟩      $$%%$$ BRACE MATCH HACK
⟨latexrelease⟩   \fi
⟨latexrelease⟩}%
⟨latexrelease⟩\expandafter\let\csname\string[ \endcsname\@undefined
⟨latexrelease⟩\def\]{%
⟨latexrelease⟩   \relax\ifmmode
⟨latexrelease⟩      \ifinner
⟨latexrelease⟩         \@badmath
⟨latexrelease⟩      \else
⟨latexrelease⟩         $$%%$$ BRACE MATCH HACK
⟨latexrelease⟩      \fi
⟨latexrelease⟩   \else
⟨latexrelease⟩      \@badmath
⟨latexrelease⟩   \fi
⟨latexrelease⟩   \ignorespaces
⟨latexrelease⟩}%
⟨latexrelease⟩\expandafter\let\csname\string] \endcsname\@undefined
⟨latexrelease⟩\EndIncludeInRelease
⟨*2ekernel⟩
```

`math` `displaymath` Disguises for `\(...\)` and `\[...\]`.

```
\let\math=\(
\let\endmath=\)
\def\displaymath{\[}
\def\enddisplaymath{\]\@ignoretrue}
```

`equation` `\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`.)

```
\@definecounter{equation}
\def\equation{$$\refstepcounter{equation}}
\def\endequation{\eqno \hbox{\@eqnnum}$$\@ignoretrue}
```

`\@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.

```
\def\@eqnnum{{\normalfont \normalcolor (\theequation)}}
```

`\stackrel` A disguise for plain TEX's buildrel.

```
\DeclareRobustCommand\stackrel[2]{\mathrel{\mathop{#2}\limits^{#1}}}
```

`\frac` A disguise for plain TEX's `\over`.

```
\DeclareRobustCommand\frac[2]{{\begingroup#1\endgroup\over#2}}
```

`\sqrt`
`\@sqrt` Add an optional argument to plain's `\sqrt` to give the nth root of an expression $\sqrt[n]{e}$.

```
\DeclareRobustCommand\sqrt{\@ifnextchar[\@sqrt\sqrtsign}
\def\@sqrt[#1]{\root #1\of}
```

`eqnarray`
`\@eqcnt`
`\@eqpen`
`\if@eqnsw`
`\@eqnsel`

Here's the eqnarray environment: Default is for left-hand side of equations to be flushright. To make them flushleft, `\let\@eqnsel = \hfil`.

```
\newcount\@eqcnt
\newcount\@eqpen
\newif\if@eqnsw\@eqnswtrue
\newskip\@centering
\@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.

```
\def\eqnarray{%
   \stepcounter{equation}%
   \def\@currentlabel{\p@equation\theequation}%
   \global\@eqnswtrue
   \m@th
   \global\@eqcnt\z@
   \tabskip\@centering
   \let\\\@eqncr
   $$\everycr{}\halign to\displaywidth\bgroup
       \hskip\@centering$\displaystyle\tabskip\z@skip{##}$\@eqnsel
      &\global\@eqcnt\@ne\hskip \tw@\arraycolsep \hfil${##}$\hfil
      &\global\@eqcnt\tw@ \hskip \tw@\arraycolsep
         $\displaystyle{##}$\hfil\tabskip\@centering
      &\global\@eqcnt\thr@@ \hb@xt@\z@\bgroup\hss##\egroup
         \tabskip\z@skip
      \cr
}
```

```
\def\endeqnarray{%
      \@@eqncr
      \egroup
      \global\advance\c@equation\m@ne
   $$\@ignoretrue
}
```

```
\let\@eqnsel=\relax
```

`\nonumber` Switches off equation numbering.

```
\def\nonumber{\global\@eqnswfalse}
```

`\@eqncr`
`\@xeqncr`
`\@yeqncr`

```
\def\@eqncr{%
   {\ifnum0=`}\fi
   \@ifstar{%
      \global\@eqpen\@M\@yeqncr
```

```
  }{%
     \global\@eqpen\interdisplaylinepenalty \@yeqncr
  }%
}
```

```
\def\@yeqncr{\@testopt\@xeqncr\z@skip}
```

```
\def\@xeqncr[#1]{%
  \ifnum0=`{\fi}%
  \@@eqncr
  \noalign{\penalty\@eqpen\vskip\jot\vskip #1\relax}%
}
```

`\@@eqncr`

```
\def\@@eqncr{\let\reserved@a\relax
    \ifcase\@eqcnt \def\reserved@a{& & &}\or \def\reserved@a{& &}%
     \or \def\reserved@a{&}\else
       \let\reserved@a\@empty
       \@latex@error{Too many columns in eqnarray environment}\@ehc\fi
     \reserved@a \if@eqnsw\@eqnnum\stepcounter{equation}\fi
     \global\@eqnswtrue\global\@eqcnt\z@\cr}
```

`eqnarray*` `\@seqncr` Here's the eqnarray* environment:

```
\let\@seqncr=\@eqncr
```

```
\@namedef{eqnarray*}{\def\@eqncr{\nonumber\@seqncr}\eqnarray}
```

```
\@namedef{endeqnarray*}{\nonumber\endeqnarray}
```

`\lefteqn` `\lefteqn{FORMULA}` typesets `FORMULA` in display math style flushleft in a box of width zero.

```
\def\lefteqn#1{\rlap{$\displaystyle #1$}}
```

`\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.

```
\DeclareRobustCommand{\ensuremath}{%
  \ifmmode
    \expandafter\@firstofone
  \else
    \expandafter\@ensuredmath
  \fi}
```

`\@ensuredmath` The `\relax` stops `\ensuremath{}` starting display math.

```
\long\def\@ensuredmath#1{$\relax#1$}
```

```
⟨/2ekernel⟩
```

64.3 External options to the standard document classes

64.3.1 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 quation is too wide, the equation number than may overprint the equation itself.

```
⟨*leqno⟩
\renewcommand\@eqnnum{\hb@xt@.01\p@{}%
                      \rlap{\normalfont\normalcolor
                        \hskip -\displaywidth(\theequation)}}
⟨/leqno⟩
```

64.3.2 Flush left equations

To get the displayed math environments to print the contents flush left (with an indentation) we have to redefine all of LaTeX 2$_\varepsilon$'s displayed math environments.

`\mathindent` The amount of indentation of the equations is stored in a register.

```
⟨*fleqn⟩
\newdimen\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 `fleqn.clo` is read in.

```
\AtEndOfClass{\mathindent\leftmargini}
```

`\[` Begin display math;

```
\IncludeInRelease{2015/01/01}{\[}{Make \[ robust}%
\DeclareRobustCommand\[{\relax
                \ifmmode\@badmath
                \else
                  \begin{trivlist}%
                    \@beginparpenalty\predisplaypenalty
                    \@endparpenalty\postdisplaypenalty
                    \item[]\leavevmode
                    \hb@xt@\linewidth\bgroup $\m@th\displaystyle %$
                      \hskip\mathindent\bgroup
                \fi}
\EndIncludeInRelease
```

```
\IncludeInRelease{0000/00/00}{\[}{Make \[ robust}%
\renewcommand\[{\relax
                \ifmmode\@badmath
                \else
                  \begin{trivlist}%
                    \@beginparpenalty\predisplaypenalty
                    \@endparpenalty\postdisplaypenalty
                    \item[]\leavevmode
                    \hb@xt@\linewidth\bgroup $\m@th\displaystyle %$
                      \hskip\mathindent\bgroup
                \fi}
\EndIncludeInRelease
```

`\]` end display math;

```
\IncludeInRelease{2015/01/01}{\]}{Make \] robust}%
\DeclareRobustCommand\]{\relax
                \ifmmode
                      \egroup $\hfil% $
```

```
                    \egroup
                  \end{trivlist}%
                \else \@badmath
                \fi}
\EndIncludeInRelease
```

```
\IncludeInRelease{0000/00/00}{\]}{Make \] robust}%
\renewcommand\]{\relax
                \ifmmode
                      \egroup $\hfil% $
                    \egroup
                  \end{trivlist}%
                \else \@badmath
                \fi}
\EndIncludeInRelease
```

equation The equation environment

```
\renewenvironment{equation}%
    {\@beginparpenalty\predisplaypenalty
     \@endparpenalty\postdisplaypenalty
     \refstepcounter{equation}%
     \trivlist \item[]\leavevmode
       \hb@xt@\linewidth\bgroup $\m@th% $
         \displaystyle
         \hskip\mathindent}%
        {$\hfil % $
         \displaywidth\linewidth\hbox{\@eqnnum}%
       \egroup
     \endtrivlist}
```

eqnarray The eqnarray environment

```
\renewenvironment{eqnarray}{%
    \stepcounter{equation}%
    \def\@currentlabel{\p@equation\theequation}%
    \global\@eqnswtrue\m@th
    \global\@eqcnt\z@
    \tabskip\mathindent
    \let\\=\@eqncr
    \setlength\abovedisplayskip{\topsep}%
    \ifvmode
      \addtolength\abovedisplayskip{\partopsep}%
    \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.

```
    \addtolength\abovedisplayskip{\parskip}%
    \setlength\belowdisplayskip{\abovedisplayskip}%
    \setlength\belowdisplayshortskip{\abovedisplayskip}%
    \setlength\abovedisplayshortskip{\abovedisplayskip}%
    $$\everycr{}\halign to\linewidth% $$
    \bgroup
      \hskip\@centering
      $\displaystyle\tabskip\z@skip{##}$\@eqnsel&%
      \global\@eqcnt\@ne \hskip \tw@\arraycolsep \hfil${##}$\hfil&%
      \global\@eqcnt\tw@ \hskip \tw@\arraycolsep
        $\displaystyle{##}$\hfil \tabskip\@centering&%
      \global\@eqcnt\thr@@
        \hb@xt@\z@\bgroup\hss##\egroup\tabskip\z@skip\cr}%
      {\@@eqncr
    \egroup
    \global\advance\c@equation\m@ne$$% $$
    \@ignoretrue
    }
⟨/fleqn⟩
```

File C
ltlists.dtx

65 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 flexability 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.

65.1 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.

65.2 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.

65.3 Penalties

`\@beginparpenalty`: put at the beginning of a list

`\@endparpenalty`: put at end of list

`\@itempenalty`: put between items.

65.4 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 <= `\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.

65.5 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 LaTeX 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 \@tempskipa := \lastskip
                    \vskip - \lastskip
                    \vskip \@tempskipa -\@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]
          END

 \@item[LAB] ==
    BEGIN
     if @noparitem = true
       then @noparitem := false
                % NOTE: then clause  hardly every taken,
                %  so made a macro \@donoparitem
            \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 LaTeX 2.09 comments.

\topskip
\partopsep
\itemsep
\parsep
\@topsep
\@topsepadd
\outerparskip

```
⟨*2ekernel⟩
\newskip\topsep
\newskip\partopsep
\newskip\itemsep
\newskip\parsep
\newskip\@topsep
\newskip\@topsepadd
```

```
\newskip\@outerparskip
```

\leftmargin
\rightmargin
\listparindent
\itemindent
\labelwidth
\labelsep
\@totalleftmargin

```
\newdimen\leftmargin
\newdimen\rightmargin
\newdimen\listparindent
\newdimen\itemindent
\newdimen\labelwidth
\newdimen\labelsep
\newdimen\linewidth
\newdimen\@totalleftmargin \@totalleftmargin=\z@
```

\leftmargini
\leftmarginii
\leftmarginiii
\leftmarginiv
\leftmarginv
\leftmarginvi

```
\newdimen\leftmargini
\newdimen\leftmarginii
\newdimen\leftmarginiii
\newdimen\leftmarginiv
\newdimen\leftmarginv
\newdimen\leftmarginvi
```

\@listdepth
\@itempenalty
\@beginparpenalty
\@endparpenalty

```
\newcount\@listdepth \@listdepth=0
\newcount\@itempenalty
\newcount\@beginparpenalty
\newcount\@endparpenalty
```

\@labels

```
\newbox\@labels
```

\if@inlabel
\@inlabelfalse
\@inlabeltrue

```
\newif\if@inlabel \@inlabelfalse
```

\if@newlist
\@newlistfalse
\@newlisttrue

```
\newif\if@newlist   \@newlistfalse
```

\if@noparitem
\@noparitemfalse
\@noparitemtrue

```
\newif\if@noparitem \@noparitemfalse
```

\if@noparlist
\@noparlistfalse
\@noparlisttrue

```
\newif\if@noparlist \@noparlistfalse
```

\if@noitemarg
\@noitemargfalse
\@noitemargtrue

```
\newif\if@noitemarg \@noitemargfalse
```

\if@newlist
\@newlistfalse
\@newlisttrue

```
\newif\if@nmbrlist  \@nmbrlistfalse
```

\list

```
\def\list#1#2{%
  \ifnum \@listdepth >5\relax
    \@toodeep
  \else
    \global\advance\@listdepth\@ne
  \fi
  \rightmargin\z@
  \listparindent\z@
  \itemindent\z@
  \csname @list\romannumeral\the\@listdepth\endcsname
  \def\@itemlabel{#1}%
  \let\makelabel\@mklab
  \@nmbrlistfalse
  #2\relax
  \@trivlist
  \parskip\parsep
  \parindent\listparindent
  \advance\linewidth -\rightmargin
  \advance\linewidth -\leftmargin
  \advance\@totalleftmargin \leftmargin
  \parshape \@ne \@totalleftmargin \linewidth
  \ignorespaces}
```

\par@deathcycles

```
\newcount\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 TeX. We thus need to trap this here by seting `\par` to count the number of times a paragraph ii is called with no progress being made started.

```
\def\@trivlist{%
  \if@noskipsec \leavevmode \fi
  \@topsepadd \topsep
  \ifvmode
    \advance\@topsepadd \partopsep
  \else
    \unskip \par
  \fi
  \if@inlabel
    \@noparitemtrue
    \@noparlisttrue
  \else
    \if@newlist \@noitemerr \fi
    \@noparlistfalse
    \@topsep \@topsepadd
  \fi
  \advance\@topsep \parskip
  \leftskip \z@skip
  \rightskip \@rightskip
  \parfillskip \@flushglue
  \par@deathcycles \z@
  \@setpar{\if@newlist
             \advance\par@deathcycles \@ne
             \ifnum \par@deathcycles >\@m
               \@noitemerr
               {\@@par}%
             \fi
           \else
             {\@@par}%
```

```
          \fi}%
  \global \@newlisttrue
  \@outerparskip \parskip}
```

\trivlist

```
\def\trivlist{%
  \parsep\parskip
  \@nmbrlistfalse
  \@trivlist
  \labelwidth\z@
  \leftmargin\z@
  \itemindent\z@
```

We initialise `\@itemlabel` so that a `trivlist` with an `\item` not having an optional argument doesn't produce an error message.

```
  \let\@itemlabel\@empty
  \def\makelabel##1{##1}}
```

\endlist

```
\def\endlist{%
  \global\advance\@listdepth\m@ne
  \endtrivlist}
```

The definition of `\trivlist` used to be in ltspace.dtx so that other commands could be 'let to it'. They now use `\def`.

\endtrivlist

```
\def\endtrivlist{%
  \if@inlabel
    \leavevmode
    \global \@inlabelfalse
  \fi
  \if@newlist
    \@noitemerr
    \global \@newlistfalse
  \fi
  \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.

```
  \else
    \@inmatherr{\end{\@currenvir}}%
  \fi
  \if@noparlist \else
    \ifdim\lastskip >\z@
      \@tempskipa\lastskip \vskip -\lastskip
      \advance\@tempskipa\parskip \advance\@tempskipa -\@outerparskip
      \vskip\@tempskipa
    \fi
    \@endparenv
  \fi
}
```

\@endparenv
\@doendpe

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)

```
\def\@endparenv{%
  \addpenalty\@endparpenalty\addvspace\@topsepadd\@endpetrue}
```

```
⟨latexrelease⟩\IncludeInRelease{2015/01/01}{\@doendpe}{clubpenalty fix}%
\def\@doendpe{\@endpetrue
     \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 explicity set \clubpenalty back to its default.

```
               \clubpenalty\@clubpenalty
               \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).

```
               {{\setbox\z@\lastbox}%
                \everypar{}\@endpefalse}}
⟨latexrelease⟩\EndIncludeInRelease
```

```
⟨latexrelease⟩\IncludeInRelease{0000/00/00}{\@doendpe}{clubpenalty fix}%
⟨latexrelease⟩\def\@doendpe{\@endpetrue
⟨latexrelease⟩     \def\par{\@restorepar\everypar{}\par\@endpefalse}\everypar
⟨latexrelease⟩               {{\setbox\z@\lastbox}\everypar{}\@endpefalse}}
⟨latexrelease⟩\EndIncludeInRelease
```

\if@endpe
\@endpefalse
\@endpeltrue

```
\newif\if@endpe
\@endpefalse
```

\@mklab

```
\def\@mklab#1{\hfil #1}
```

\item

```
\def\item{%
  \@inmatherr\item
  \@ifnextchar [\@item{\@noitemargtrue \@item[\@itemlabel]}}
```

\@donoparitem

```
\def\@donoparitem{%
  \@noparitemfalse
  \global\setbox\@labels\hbox{\hskip -\leftmargin
                              \unhbox\@labels
                               \hskip \leftmargin}%
  \if@minipage\else
    \@tempskipa\lastskip
    \vskip -\lastskip
    \advance\@tempskipa\@outerparskip
    \advance\@tempskipa -\parskip
    \vskip\@tempskipa
  \fi}
```

\@item

```
\def\@item[#1]{%
  \if@noparitem
    \@donoparitem
  \else
    \if@inlabel
      \indent \par
    \fi
    \ifhmode
      \unskip\unskip \par
    \fi
    \if@newlist
      \if@nobreak
        \@nbitem
      \else
        \addpenalty\@beginparpenalty
        \addvspace\@topsep
        \addvspace{-\parskip}%
      \fi
    \else
      \addpenalty\@itempenalty
      \addvspace\itemsep
    \fi
    \global\@inlabeltrue
  \fi
  \everypar{%
    \@minipagefalse
    \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.

```
    \if@inlabel
      \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.

```
      {\setbox\z@\lastbox
       \ifvoid\z@
         \kern-\itemindent
       \fi}%
      \box\@labels
      \penalty\z@
    \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.

```
    \if@nobreak
      \@nobreakfalse
      \clubpenalty \@M
    \else
      \clubpenalty \@clubpenalty
      \everypar{}%
    \fi}%
```

```
  \if@noitemarg
    \@noitemargfalse
    \if@nmbrlist
```

```
      \refstepcounter\@listctr
    \fi
  \fi
```

We use `\sbox` to support colour commands.

```
  \sbox\@tempboxa{\makelabel{#1}}%
  \global\setbox\@labels\hbox{%
    \unhbox\@labels
    \hskip \itemindent
    \hskip -\labelwidth
    \hskip -\labelsep
    \ifdim \wd\@tempboxa >\labelwidth
      \box\@tempboxa
```

```
    \else
      \hbox to\labelwidth {\unhbox\@tempboxa}%
    \fi
    \hskip \labelsep}%
  \ignorespaces}
```

`\makelabel`

```
\def\makelabel#1{%
  \@latex@error{Lonely \string\item--perhaps a missing
        list environment}\@ehc}
```

`\@nbitem`

```
\def\@nbitem{%
  \@tempskipa\@outerparskip
  \advance\@tempskipa -\parskip
  \addvspace\@tempskipa}
```

`\usecounter`

```
\def\usecounter#1{\@nmbrlisttrue\def\@listctr{#1}\setcounter{#1}\z@}
```

65.6 Itemize and Enumerate

Enumeration is done with four counters: `enumi`, `enumii`, `enumiii` and `enumiv`, where `enum`N 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@enum`N`\theenum`N 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 LaTeX 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`

```
\newcount\@enumdepth \@enumdepth = 0
```

`\c@enumi`
`\c@enumii`
`\c@enumii`
`\c@enumiv`

```
\@definecounter{enumi}
\@definecounter{enumii}
\@definecounter{enumiii}
\@definecounter{enumiv}
```

`enumerate`

```
\def\enumerate{%
  \ifnum \@enumdepth >\thr@@\@toodeep\else
    \advance\@enumdepth\@ne
    \edef\@enumctr{enum\romannumeral\the\@enumdepth}%
```

```
      \expandafter
      \list
        \csname label\@enumctr\endcsname
        {\usecounter\@enumctr\def\makelabel##1{\hss\llap{##1}}}%
  \fi}
\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

```
\newcount\@itemdepth \@itemdepth = 0
```

itemize

```
\def\itemize{%
  \ifnum \@itemdepth >\thr@@\@toodeep\else
    \advance\@itemdepth\@ne
    \edef\@itemitem{labelitem\romannumeral\the\@itemdepth}%

    \expandafter
    \list
      \csname\@itemitem\endcsname
      {\def\makelabel##1{\hss\llap{##1}}}%
  \fi}
\let\enditemize =\endlist
⟨/2ekernel⟩
```

File D
ltboxes.dtx

66 LaTeX Box commands

`\makebox` `\makebox[`⟨*wid*⟩`][`⟨*pos*⟩`]{`⟨*obj*⟩`}`
Puts ⟨*obj*⟩ in an `\hbox` of width ⟨*wid*⟩, positioned by ⟨*pos*⟩.

The possible ⟨*pos*⟩ are:
`s` stretched,
`l` flushleft,
`r` flushright,
`c` (default) centred.

If ⟨*wid*⟩ is missing, then ⟨*pos*⟩ is also missing and ⟨*obj*⟩ is put in an `\hbox` of its natural width.

`\makebox(`⟨*x*⟩`,`⟨*y*⟩`)[`⟨*pos*⟩`]{`⟨*obj*⟩`}`
Puts ⟨*obj*⟩ in an `\hbox` of width $x*$`\unitlength` and height $y*$`\unitlength`. ⟨*pos*⟩ 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 `\makebox` a [b] aligns on the *bottom* of the text as documented. If you want to align on the *baseline* use `\makebox( , )[b]{\raisebox{0pt}[\height][0pt]{xyz}}` or `\makebox( , )[b]{\smash{xyz}}`

`\mbox` `\mbox{`⟨*obj*⟩`}` The same as `\makebox{`⟨*obj*⟩`}`, but is more efficient as no checking for optional arguments is done.

`\newsavebox` `\newsavebox{\cmd}` : If `\cmd` is undefined, then defines it to be a TeX box register.

`\savebox` `\savebox{\cmd}` ... : `\cmd` is defined to be a TeX box register, and the '...' are any `\makebox` arguments. It is like `\makebox`, except it doesn't produce text but saves the value in `\box \cmd`.

`\sbox` `\sbox{`⟨*cmd*⟩`}{`⟨*obj*⟩`}` is an efficient abbreviation for
`\savebox{`⟨*cmd*⟩`}{`⟨*obj*⟩`}`.

`lrbox` `\begin{lrbox}{`⟨*cmd*⟩`}`⟨*text*⟩`\end{lrbox}` is equivalent to
`\sbox{`⟨*cmd*⟩`}{`⟨*text*⟩`}`
except that any white space at the beginning and end of ⟨*text*⟩ is ignored.

`\framebox` `\framebox` ... : like `\makebox`, except it puts a 'frame' around the box. The frame is made of lines of thickness `\fboxrule`, separated by space `\fboxsep` from the text – except for `\framebox(X,Y)` ... , where the thickness of the lines is as for the picture environment, and there is no separation added.

`\fbox` `\fbox{`⟨*obj*⟩`}` is an abbreviation for `\framebox{`⟨*obj*⟩`}`.

`\parbox` `\parbox[`⟨*pos*⟩`][`⟨*height*⟩`][`⟨*inner-pos*⟩`]{`⟨*width*⟩`}{`⟨*text*⟩`}` : Makes a box with `\hsize` ⟨*width*⟩, positioned by ⟨*pos*⟩ as follows: c : `\vcenter` (placed in `$...$` if not in math mode) b : `\vbox` t : `\vtop` default value is c. Sets `\hsize` := ⟨*width*⟩ and calls `\@parboxrestore`, which does the following: Restores the original definitions of:

`\par`
`\\`
`\-`
`\'`
`\‘`
`\=`

Resets the following parameters:

`\parindent`	=	0pt	
`\parskip`	=	0pt	added 20 Jan 87
`\linewidth`	=	`\hsize`	
`\@totalleftmargin`	=	0pt	
`\leftskip`	=	0pt	
`\rightskip`	=	0pt	
`\@rightskip`	=	0pt	
`\parfillskip`	=	0pt plus 1fil	
`\lineskip`	=	`\normallineskip`	
`\baselineskip`	=	`\normalbaselineskip`	

Calls `\sloppy`

Note: `\@arrayparboxrestore` same as `\@parboxrestore` but it doesn't restore `\\`.

`minipage` `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.

`\rule` `\rule[`⟨*raised*⟩`]{`⟨*width*⟩`}{`⟨*height*⟩`}` : Makes a ⟨*width*⟩ $*$ ⟨*height*⟩ rule, raised ⟨*raised*⟩.

`\underline` `\underline{`⟨*text*⟩`}` : Makes an underlined hbox with ⟨*text*⟩ in it.

`\raisebox` `\raisebox{`⟨*distance*⟩`}[`⟨*height*⟩`][`⟨*depth*⟩`]{`⟨*box*⟩`}` :
Raises ⟨*box*⟩ up by ⟨*distance*⟩ length (down if ⟨*distance*⟩ negative). Makes TEX 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.

```
⟨*2ekernel⟩
\message{boxes,}
```

`\makebox` `\makebox` User level command just looks for optional `[` or `(`.

```
⟨/2ekernel⟩
⟨latexrelease⟩\IncludeInRelease{2015/01/01}%
⟨latexrelease⟩                    {\makebox}{Make \makebox robust}%
```

```
⟨*2ekernel | latexrelease⟩
\DeclareRobustCommand\makebox{%
  \leavevmode
  \@ifnextchar(%)
    \@makepicbox
    {\@ifnextchar[\@makebox\mbox}}%
⟨/2ekernel | latexrelease⟩
⟨latexrelease⟩\EndIncludeInRelease
⟨latexrelease⟩\IncludeInRelease{0000/00/00}%
⟨latexrelease⟩                 {\makebox}{Make \makebox robust}%
⟨latexrelease⟩\def\makebox{%
⟨latexrelease⟩  \leavevmode
⟨latexrelease⟩  \@ifnextchar(%)
⟨latexrelease⟩    \@makepicbox
⟨latexrelease⟩    {\@ifnextchar[\@makebox\mbox}}%
⟨latexrelease⟩\expandafter\let\csname makebox \endcsname\@undefined
⟨latexrelease⟩\EndIncludeInRelease
⟨*2ekernel⟩
```

`\mbox` The basic horizontal box command for LaTeX.

```
\DeclareRobustCommand\mbox[1]{\leavevmode\hbox{#1}}
```

`\@makebox` Look for a possible second optional argument (defaults to `c`).

```
\def\@makebox[#1]{%
  \@ifnextchar [{\@imakebox[#1]}{\@imakebox[#1][c]}}
```

`\@begin@tempboxa` Helper macro for supporting `\height`, `\width` etc. Grab `#1` into `\@tempboxa` and measure it.

```
\long\def\@begin@tempboxa#1#2{%
   \begingroup
     \setbox\@tempboxa#1{\color@begingroup#2\color@endgroup}%
     \def\width{\wd\@tempboxa}%
     \def\height{\ht\@tempboxa}%
     \def\depth{\dp\@tempboxa}%
     \let\totalheight\@ovri
     \totalheight\height
     \advance\totalheight\depth}
```

`\@end@tempboxa` End the group started by `\@begin@tempboxa`, so that the scope of `\height` only includes the 'length' argument to the user-command.

```
\let\@end@tempboxa\endgroup
```

`\bm@c` `\bm@l` `\bm@r` `\bm@s` `\bm@t` `\bm@b` Set up spacing.

```
\def\bm@c{\hss\unhbox\@tempboxa\hss}
\def\bm@l{\unhbox\@tempboxa\hss}\let\bm@t\bm@l
\def\bm@r{\hss\unhbox\@tempboxa}\let\bm@b\bm@r
\def\bm@s{\unhbox\@tempboxa}
```

`\@imakebox` Internal form of `\makebox`.

```
\long\def\@imakebox[#1][#2]#3{%
  \@begin@tempboxa\hbox{#3}%
    \setlength\@tempdima{#1}%       support calc
    \hb@xt@\@tempdima{\csname bm@#2\endcsname}%
  \@end@tempboxa}
```

\@makepicbox

Picture mode form of \makebox.

```
\def\@makepicbox(#1,#2){%
  \@ifnextchar[{\@imakepicbox(#1,#2)}{\@imakepicbox(#1,#2)[]}}
```

\@imakepicbox

picture mode version

```
\long\def\@imakepicbox(#1,#2)[#3]#4{%
  \vbox to#2\unitlength
   {\let\mb@b\vss \let\mb@l\hss\let\mb@r\hss
    \let\mb@t\vss
    \@tfor\reserved@a :=#3\do{%
      \if s\reserved@a
        \let\mb@l\relax\let\mb@r\relax
      \else
        \expandafter\let\csname mb@\reserved@a\endcsname\relax
      \fi}%
    \mb@t
    \hb@xt@ #1\unitlength{\mb@l #4\mb@r}%
    \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 LaTeXBook. The kern is removed in compatibility mode.

```
    \kern\z@}}
```

\set@color

This macro is initially a no-op, but the colour package will redefine it to insert a \special.

```
\let\set@color\relax
```

\color@begingroup
\color@endgroup
\color@setgroup
\normalcolor
\color@hbox
\color@vbox
\color@endbox

These macros are initially a no-op, but the colour package will redefine them to be \begingroup, \endgroup, \begingroup\set@color, \hbox\bgroup\color@begingroup, \color@endgroup\egroup. and ⟨*set to main document colour*⟩ respectively.

```
\let\color@begingroup\relax
\let\color@endgroup\relax
\let\color@setgroup\relax
\let\normalcolor\relax
\let\color@hbox\relax
\let\color@vbox\relax
\let\color@endbox\relax
```

\newsavebox

Allocate a new 'savebox'.

```
\def\newsavebox#1{\@ifdefinable{#1}{\newbox#1}}
```

\savebox

Save #1 in a box register.

```
⟨/2ekernel⟩
⟨latexrelease⟩\IncludeInRelease{2015/01/01}%
⟨latexrelease⟩                    {\savebox}{Make \savebox robust}%
⟨*2ekernel | latexrelease⟩
\DeclareRobustCommand\savebox[1]{%
  \@ifnextchar(%)
    {\@savepicbox#1}{\@ifnextchar[{\@savebox#1}{\sbox#1}}}%
⟨/2ekernel | latexrelease⟩
⟨latexrelease⟩\EndIncludeInRelease
```

```
⟨latexrelease⟩\IncludeInRelease{0000/00/00}%
⟨latexrelease⟩                 {\savebox}{Make \savebox robust}%
⟨latexrelease⟩\def\savebox#1{%
⟨latexrelease⟩  \@ifnextchar(%)
⟨latexrelease⟩    {\@savepicbox#1}{\@ifnextchar[{\@savebox#1}{\sbox#1}}}%
⟨latexrelease⟩\expandafter\let\csname savebox \endcsname\@undefined
⟨latexrelease⟩\EndIncludeInRelease
⟨*2ekernel⟩
```

`\sbox` Save `#1` in a box register.

```
\DeclareRobustCommand\sbox[2]{\setbox#1\hbox{%
  \color@setgroup#2\color@endgroup}}
```

`\@savebox` Look for second optional argument.

```
\def\@savebox#1[#2]{%
  \@ifnextchar [{\@isavebox#1[#2]}{\@isavebox#1[#2][c]}}
```

`\@isavebox`

```
\long\def\@isavebox#1[#2][#3]#4{%
  \sbox#1{\@imakebox[#2][#3]{#4}}}
```

`\@savepicbox` Picture mode version of `\savebox`.

```
\def\@savepicbox#1(#2,#3){%
  \@ifnextchar[%]
    {\@isavepicbox#1(#2,#3)}{\@isavepicbox#1(#2,#3)[]}}
```

`\@isavepicbox` Picture mode version of `\savebox`.

```
\long\def\@isavepicbox#1(#2,#3)[#4]#5{%
  \sbox#1{\@imakepicbox(#2,#3)[#4]{#5}}}
```

`\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.

```
\def\lrbox#1{%
  \edef\reserved@a{%
    \endgroup
    \setbox#1\hbox{%
      \begingroup\aftergroup}%
        \def\noexpand\@currenvir{\@currenvir}%
        \def\noexpand\@currenvline{\on@line}}%
  \reserved@a
    \@endpefalse
    \color@setgroup
      \ignorespaces}
```

`\endlrbox` End the `lrbox` environment.

```
\def\endlrbox{\unskip\color@endgroup}
```

`\usebox` unchanged

```
\DeclareRobustCommand\usebox[1]{\leavevmode\copy #1\relax}
```

\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

```
\DeclareRobustCommand\frame[1]{%
  \leavevmode
  \hbox{%
    \hskip-\@wholewidth
    \vbox{%
      \vskip-\@wholewidth
      \hrule \@height\@wholewidth
      \hbox{%
        \vrule\@width\@wholewidth
        #1%
        \vrule\@width\@wholewidth}%
      \hrule\@height\@wholewidth
      \vskip-\@wholewidth}%
    \hskip-\@wholewidth}}
```

\fboxrule \fboxsep user level parameters,

```
\newdimen\fboxrule
\newdimen\fboxsep
```

\fbox Abbreviated framed box command.

```
\DeclareRobustCommand\fbox[1]{%
  \leavevmode
  \setbox\@tempboxa\hbox{%
    \color@begingroup
      \kern\fboxsep{#1}\kern\fboxsep
    \color@endgroup}%
  \@frameb@x\relax}
```

\framebox Framed version of \makebox.

```
⟨/2ekernel⟩
⟨latexrelease⟩\IncludeInRelease{2015/01/01}%
⟨latexrelease⟩                  {\framebox}{Make \framebox robust}%
⟨*2ekernel | latexrelease⟩
\DeclareRobustCommand\framebox{%
  \@ifnextchar(%)
    \@framepicbox{\@ifnextchar[\@framebox\fbox}}%
⟨/2ekernel | latexrelease⟩
⟨latexrelease⟩\EndIncludeInRelease
⟨latexrelease⟩\IncludeInRelease{0000/00/00}%
⟨latexrelease⟩                  {\framebox}{Make \framebox robust}%
⟨latexrelease⟩\def\framebox{%
⟨latexrelease⟩  \@ifnextchar(%)
⟨latexrelease⟩    \@framepicbox{\@ifnextchar[\@framebox\fbox}}%
⟨latexrelease⟩\expandafter\let\csname framebox \endcsname\@undefined
⟨latexrelease⟩\EndIncludeInRelease
⟨*2ekernel⟩
```

\@framebox Deal with optional arguments.

```
\def\@framebox[#1]{%
  \@ifnextchar[%]
```

```
    {\@iframebox[#1]}%
    {\@iframebox[#1][c]}}
```

\@iframebox 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.

```
\long\def\@iframebox[#1][#2]#3{%
  \leavevmode
  \@begin@tempboxa\hbox{#3}%
    \setlength\@tempdima{#1}%
    \setbox\@tempboxa\hb@xt@\@tempdima
         {\kern\fboxsep\csname bm@#2\endcsname\kern\fboxsep}%
    \@frameb@x{\kern-\fboxrule}%
  \@end@tempboxa}
```

\@frameb@x Common part of \framebox and \fbox. #1 is a negative kern in the \framebox case so that the vertical rules do not add to the width of the box.

```
\def\@frameb@x#1{%
  \@tempdima\fboxrule
  \advance\@tempdima\fboxsep
  \advance\@tempdima\dp\@tempboxa
  \hbox{%
    \lower\@tempdima\hbox{%
      \vbox{%
        \hrule\@height\fboxrule
        \hbox{%
          \vrule\@width\fboxrule
          #1%
          \vbox{%
            \vskip\fboxsep
            \box\@tempboxa
            \vskip\fboxsep}%
          #1%
          \vrule\@width\fboxrule}%
        \hrule\@height\fboxrule}%
                          }%
        }%
}
```

\@framepicbox Picture mode version.

```
\def\@framepicbox(#1,#2){%
  \@ifnextchar[{\@iframepicbox(#1,#2)}{\@iframepicbox(#1,#2)[]}}
```

\@iframepicbox Picture mode version.

```
\long\def\@iframepicbox(#1,#2)[#3]#4{%
  \frame{\@imakepicbox(#1,#2)[#3]{#4}}}
```

\parbox The main vertical-box command for LaTeX.

```
⟨/2ekernel⟩
⟨latexrelease⟩\IncludeInRelease{2015/01/01}%
⟨latexrelease⟩                 {\parbox}{Make \parbox robust}%
⟨*2ekernel | latexrelease⟩
\DeclareRobustCommand\parbox{%
```

```
  \@ifnextchar[%]
    \@iparbox
    {\@iiiparbox c\relax[s]}}%
⟨/2ekernel | latexrelease⟩
⟨latexrelease⟩\EndIncludeInRelease
⟨latexrelease⟩\IncludeInRelease{0000/00/00}%
⟨latexrelease⟩                 {\parbox}{Make \parbox robust}%
⟨latexrelease⟩\def\parbox{%
⟨latexrelease⟩  \@ifnextchar[%]
⟨latexrelease⟩    \@iparbox
⟨latexrelease⟩    {\@iiiparbox c\relax[s]}}%
⟨latexrelease⟩\expandafter\let\csname parbox \endcsname\@undefined
⟨latexrelease⟩\EndIncludeInRelease
⟨*2ekernel⟩
```

\@iparbox Optional argument handling.

```
\def\@iparbox[#1]{%
  \@ifnextchar[%]
    {\@iiparbox{#1}}%
    {\@iiiparbox{#1}\relax[s]}}
```

\@iiparbox Optional argument handling.

```
\def\@iiparbox#1[#2]{%
  \@ifnextchar[%]
    {\@iiiparbox{#1}{#2}}%
    {\@iiiparbox{#1}{#2}[#1]}}
```

\@iiiparbox The internal version of \parbox.

\@parboxto

```
\let\@parboxto\@empty
\long\def\@iiiparbox#1#2[#3]#4#5{%
  \leavevmode
  \@pboxswfalse
  \setlength\@tempdima{#4}%
  \@begin@tempboxa\vbox{\hsize\@tempdima\@parboxrestore#5\@@par}%
    \ifx\relax#2\else
      \setlength\@tempdimb{#2}%
      \edef\@parboxto{to\the\@tempdimb}%
    \fi
    \if#1b\vbox
    \else\if #1t\vtop
    \else\ifmmode\vcenter
    \else\@pboxswtrue $\vcenter
    \fi\fi\fi
    \@parboxto{\let\hss\vss\let\unhbox\unvbox
       \csname bm@#3\endcsname}%
    \if@pboxsw \m@th$\fi
  \@end@tempboxa}
```

\@arrayparboxrestore Restore various paragraph parameters.

The rational for allowing two normally global flags to be set locally here was stated originally by Donald Arsenau 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@nobreak`; otherwise this command will be redundant.

```
⟨/2ekernel⟩
⟨latexrelease⟩\IncludeInRelease{2017-04-15}%
⟨latexrelease⟩                 {\normallineskiplimit}
⟨latexrelease⟩                 {reset \lineskiplimit}%
⟨*2ekernel | latexrelease⟩
\def\@arrayparboxrestore{%
  \let\if@nobreak\iffalse
  \let\if@noskipsec\iffalse
  \let\par\@@par
  \let\-\@dischyph
```

Redefined accents to allow changes in font encoding

```
  \let\'\@acci\let\`\@accii\let\=\@acciii
  \parindent\z@ \parskip\z@skip
  \everypar{}%
  \linewidth\hsize
  \@totalleftmargin\z@
  \leftskip\z@skip \rightskip\z@skip \@rightskip\z@skip
  \parfillskip\@flushglue
  \lineskip\normallineskip
```

```
  \lineskiplimit\normallineskiplimit
```

```
  \baselineskip\normalbaselineskip
  \sloppy}
⟨/2ekernel | latexrelease⟩
```

```
⟨latexrelease⟩\EndIncludeInRelease
⟨latexrelease⟩\IncludeInRelease{0000-00-00}%
⟨latexrelease⟩                 {\normallineskiplimit}
⟨latexrelease⟩                 {reset \lineskiplimit}%
⟨latexrelease⟩\def\@arrayparboxrestore{%
⟨latexrelease⟩  \let\if@nobreak\iffalse
⟨latexrelease⟩  \let\if@noskipsec\iffalse
⟨latexrelease⟩  \let\par\@@par
⟨latexrelease⟩  \let\-\@dischyph
⟨latexrelease⟩  \let\'\@acci\let\`\@accii\let\=\@acciii
⟨latexrelease⟩  \parindent\z@ \parskip\z@skip
⟨latexrelease⟩  \everypar{}%
⟨latexrelease⟩  \linewidth\hsize
⟨latexrelease⟩  \@totalleftmargin\z@
⟨latexrelease⟩  \leftskip\z@skip \rightskip\z@skip \@rightskip\z@skip
⟨latexrelease⟩  \parfillskip\@flushglue \lineskip\normallineskip
⟨latexrelease⟩  \baselineskip\normalbaselineskip
⟨latexrelease⟩  \sloppy}
⟨latexrelease⟩\EndIncludeInRelease
⟨*2ekernel⟩
```

`\parboxrestore` Restore various paragraph parameters, and also `\\`.

```
\def\@parboxrestore{\@arrayparboxrestore\let\\\@normalcr}
```

\if@minipage Switch that is true at the start of a minipage.

```
\def\@minipagefalse{\global\let\if@minipage\iffalse}
\def\@minipagetrue {\global\let\if@minipage\iftrue}
\@minipagefalse
```

\minipage Essentially an environment form of `\parbox`.

```
\def\minipage{%
  \@ifnextchar[%]
    \@iminipage
    {\@iiiminipage c\relax[s]}}
```

\@iminipage Optional argument handling.

```
\def\@iminipage[#1]{%
  \@ifnextchar[%]
    {\@iiminipage{#1}}%
    {\@iiiminipage{#1}\relax[s]}}
```

\@iiminipage Optional argument handling.

```
\def\@iiminipage#1[#2]{%
  \@ifnextchar[%]
    {\@iiiminipage{#1}{#2}}%
    {\@iiiminipage{#1}{#2}[#1]}}
```

\@iiiminipage Internal form of `minipage`.

```
\def\@iiiminipage#1#2[#3]#4{%
  \leavevmode
  \@pboxswfalse
  \setlength\@tempdima{#4}%
  \def\@mpargs{{#1}{#2}[#3]{#4}}%
  \setbox\@tempboxa\vbox\bgroup
    \color@begingroup
      \hsize\@tempdima
      \textwidth\hsize \columnwidth\hsize
      \@parboxrestore
      \def\@mpfn{mpfootnote}\def\thempfn{\thempfootnote}\c@mpfootnote\z@
      \let\@footnotetext\@mpfootnotetext
      \let\@listdepth\@mplistdepth \@mplistdepth\z@
      \@minipagerestore
      \@setminipage}
```

\@minipagerestore Hook so that other styles can reset other commands in a minipage.

```
 \let\@minipagerestore=\relax
```

\endminipage

```
\def\endminipage{%
    \par
    \unskip
    \ifvoid\@mpfootins\else
      \vskip\skip\@mpfootins
      \normalcolor
      \footnoterule
      \unvbox\@mpfootins
    \fi
    \@minipagefalse   %% added 24 May 89
  \color@endgroup
  \egroup
  \expandafter\@iiiparbox\@mpargs{\unvbox\@tempboxa}}
```

\@mplistdepth
\@mpfootins

Versions of \@listdepth and \footins local to minipage.

```
\newcount\@mplistdepth
\newinsert\@mpfootins
```

\@mpfootnotetext

Minipage version of \@footnotetext.

Final \strut added 27 Mar 89, on suggestion by Don Hosek

```
\long\def\@mpfootnotetext#1{%
  \global\setbox\@mpfootins\vbox{%
    \unvbox\@mpfootins
    \reset@font\footnotesize
    \hsize\columnwidth
    \@parboxrestore
    \protected@edef\@currentlabel
         {\csname p@mpfootnote\endcsname\@thefnmark}%
    \color@begingroup
      \@makefntext{%
        \rule\z@\footnotesep\ignorespaces#1\@finalstrut\strutbox}%
    \color@endgroup}}
```

```
\newif\if@pboxsw
```

\rule

Draw a rule of the specified size.

```
⟨/2ekernel⟩
⟨latexrelease⟩\IncludeInRelease{2015/01/01}%
⟨latexrelease⟩                 {\rule}{Make \rule robust}%
⟨*2ekernel | latexrelease⟩
\DeclareRobustCommand\rule{\@ifnextchar[\@rule{\@rule[\z@]}}%
⟨/2ekernel | latexrelease⟩
⟨latexrelease⟩\EndIncludeInRelease
⟨latexrelease⟩\IncludeInRelease{0000/00/00}%
⟨latexrelease⟩                 {\rule}{Make \rule robust}%
⟨latexrelease⟩\def\rule{\@ifnextchar[\@rule{\@rule[\z@]}}%
⟨latexrelease⟩\expandafter\let\csname rule \endcsname\@undefined
⟨latexrelease⟩\EndIncludeInRelease
⟨*2ekernel⟩
```

\@rule

Internal form of \rule.

```
 \def\@rule[#1]#2#3{%
   \leavevmode
   \hbox{%
     \setlength\@tempdima{#1}%
     \setlength\@tempdimb{#2}%
     \setlength\@tempdimc{#3}%
     \advance\@tempdimc\@tempdima
     \vrule\@width\@tempdimb\@height\@tempdimc\@depth-\@tempdima}}
```

\@@underline

Saved primitive \underline.

```
\let\@@underline\underline
```

`\underline` LaTeX version works outside math.

```
\DeclareRobustCommand\underline[1]{%
  \relax
  \ifmmode\@@underline{#1}%
  \else $\@@underline{\hbox{#1}}\m@th$\relax\fi}
```

`\raisebox` Raise a box, and change its vertical dimensions.

```
⟨/2ekernel⟩
⟨latexrelease⟩\IncludeInRelease{2015/01/01}%
⟨latexrelease⟩                        {\raisebox}{Make \raisebox robust}%
⟨*2ekernel | latexrelease⟩
\DeclareRobustCommand\raisebox[1]{%
  \leavevmode
  \@ifnextchar[{\@rsbox{#1}}{\@irsbox{#1}[]}}
⟨/2ekernel | latexrelease⟩
⟨latexrelease⟩\EndIncludeInRelease
⟨latexrelease⟩\IncludeInRelease{0000/00/00}%
⟨latexrelease⟩                        {\raisebox}{Make \raisebox robust}%
⟨latexrelease⟩\def\raisebox#1{%
⟨latexrelease⟩  \leavevmode
⟨latexrelease⟩  \@ifnextchar[{\@rsbox{#1}}{\@irsbox{#1}[]}}
⟨latexrelease⟩\expandafter\let\csname raisebox \endcsname\@undefined
⟨latexrelease⟩\EndIncludeInRelease
⟨*2ekernel⟩
```

`\@rsbox` Optional argument handling.

```
\def\@rsbox#1[#2]{%
  \@ifnextchar[{\@iirsbox{#1}[#2]}{\@irsbox{#1}[#2]}}
```

`\@argrsbox` ...

`\@irsbox` Internal version of `\raisebox` (less than two optional args).

```
\long\def\@irsbox#1[#2]#3{%
  \@begin@tempboxa\hbox{#3}%
    \setlength\@tempdima{#1}%
    \ifx\\#2\\\else\setlength\@tempdimb{#2}\fi
    \setbox\@tempboxa\hbox{\raise\@tempdima\box\@tempboxa}%
    \ifx\\#2\\\else\ht\@tempboxa\@tempdimb\fi
    \box\@tempboxa
  \@end@tempboxa}
```

`\@iirsbox` Internal version of `\raisebox` (two optional args).

```
\long\def\@iirsbox#1[#2][#3]#4{%
  \@begin@tempboxa\hbox{#4}%
    \setlength\@tempdima{#1}%
    \setlength\@tempdimb{#2}%
    \setlength\dimen@{#3}%
    \setbox\@tempboxa\hbox{\raise\@tempdima\box\@tempboxa}%
    \ht\@tempboxa\@tempdimb
    \dp\@tempboxa\dimen@
    \box\@tempboxa
  \@end@tempboxa}
```

`\@finalstrut` This macro adds a special strut the *depth* of the box given as `#1`, and height and width 0pt. 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.

```
\def\@finalstrut#1{%
  \unskip\ifhmode\nobreak\fi\vrule\@width\z@\@height\z@\@depth\dp#1}
```

66.1 Some low-level constructs

The following commands are basically inherited from plain TEX.

`\leftline` `\rightline` `\centerline` `\@@line` These macros place text on a full line either centred or left or right adjusted.

```
\def\@@line{\hb@xt@\hsize}
\DeclareRobustCommand\leftline[1]{\@@line{#1\hss}}
\DeclareRobustCommand\rightline[1]{\@@line{\hss#1}}
\DeclareRobustCommand\centerline[1]{\@@line{\hss#1\hss}}
```

`\rlap` `\llap` These macros place text to the left or right of the current reference point without taking up space.

```
\DeclareRobustCommand\rlap[1]{\hb@xt@\z@{#1\hss}}
\DeclareRobustCommand\llap[1]{\hb@xt@\z@{\hss#1}}
```

```
⟨/2ekernel⟩
```

File E

lttab.dtx

67 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 LaTeX 2.09 version, not the extended version described in *The LaTeX Companion*. Use the array package to obtain the extended version.

67.1 tabbing

Historical LaTeX 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 :=  \@nxttabmar :=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 untabs”      fi
     else error message “Left tab in middle of line”
   fi
  END

 \@tabplus ==
  BEGIN
        if  \@nxttabmar < \@hightab
            then \@nxttabmar :=G \@nxttabmar+1
            else error message “Undefined tab”
        fi
  END

 \@tabminus ==
  BEGIN
        if \@nxttabmar > \@firsttab
            then \@nxttabmar :=G \@nxttabmar-1
            else error message “Too many untabs”
        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
     \begingroup
     \@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 LaTeX 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.

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.

```
⟨2ekernel⟩\expandafter\let\csname ver@autotabg.sty\endcsname\fmtversion
```

\@firsttab
\@maxtab

```
⟨*2ekernel⟩
\newdimen\@gtempa
\chardef\@firsttab=\the\allocationnumber
\newdimen\@gtempa\newdimen\@gtempa\newdimen\@gtempa\newdimen\@gtempa
\newdimen\@gtempa\newdimen\@gtempa\newdimen\@gtempa\newdimen\@gtempa
\newdimen\@gtempa\newdimen\@gtempa\newdimen\@gtempa\newdimen\@gtempa
```

```
\newdimen\@gtempa
\chardef\@maxtab=\the\allocationnumber
\dimen\@firsttab=0pt
```

\@nxttabmar
\@curtabmar
\@curtab
\@hightab
\@tabpush

```
\newcount\@nxttabmar
\newcount\@curtabmar
\newcount\@curtab
\newcount\@hightab
\newcount\@tabpush
```

\@curline
\@curfield
\@tabfbox

```
\newbox\@curline
\newbox\@curfield
\newbox\@tabfbox
```

\if@rjfield

```
\newif\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.

```
\def\@startline{%
     \ifnum \@nxttabmar >\@hightab
       \@badtab
       \global\@nxttabmar \@hightab
     \fi
     \global\@curtabmar \@nxttabmar
     \global\@curtab \@curtabmar
     \global\setbox\@curline \hbox {}%
     \@startfield
     \strut}
```

\@stopline

```
\def\@stopline{%
  \unskip
  \@stopfield
  \if@rjfield
    \global\@rjfieldfalse
    \@tempdima\@totalleftmargin
    \advance\@tempdima\linewidth
    \hb@xt@\@tempdima{%
      \@itemfudge\hskip\dimen\@curtabmar
      \box\@curline
      \hfil
      \box\@curfield}%
  \else
    \@addfield
    \hbox{\@itemfudge\hskip\dimen\@curtabmar\box\@curline}%
  \fi}
```

\@startfield

```
\def\@startfield{%
  \global\setbox\@curfield\hbox\bgroup\color@begingroup}
```

\@stopfield

```
\def\@stopfield{%
  \color@endgroup\egroup}
```

\@contfield

```
\def\@contfield{%
  \global\setbox\@curfield\hbox\bgroup\color@begingroup
  \unhbox\@curfield}
```

\@addfield

```
\def\@addfield{\global\setbox\@curline\hbox{\unhbox
     \@curline\unhbox\@curfield}}
```

\@ifatmargin

```
\def\@ifatmargin{\ifdim \wd\@curline =\z@}
```

\@tabcr

```
\def\@tabcr{\@stopline \@ifstar{\penalty \@M \@xtabcr}\@xtabcr}
```

\@xtabcr

```
\def\@xtabcr{\@ifnextchar[\@itabcr{\@startline\ignorespaces}}
```

\@itabcr

```
\def\@itabcr[#1]{\vskip #1\@startline\ignorespaces}
```

\tabbing We use \relax to prevent \item from scanning too far.

```
\def\tabbing{\lineskip \z@skip\let\>\@rtab\let\<\@ltab\let\=\@settab
     \let\+\@tabplus\let\-\@tabminus\let\`\@tabrj\let\'\@tablab
     \let\\=\@tabcr
     \@hightab\@firsttab
     \global\@nxttabmar\@firsttab
     \dimen\@firsttab\@totalleftmargin
     \global\@tabpush\z@ \global\@rjfieldfalse
     \trivlist \item\relax
     \if@minipage\else\vskip\parskip\fi
```

```
     \setbox\@tabfbox\hbox{%
       \rlap{\hskip\@totalleftmargin\indent\the\everypar}}%
     \def\@itemfudge{\box\@tabfbox}%
     \@startline\ignorespaces}
```

\endtabbing

```
\def\endtabbing{%
  \@stopline\ifnum\@tabpush >\z@ \@badpoptabs \fi\endtrivlist}
```

\@rtab Omitted \global added to \@rtab 17 Jun 86

```
\def\@rtab{\@stopfield\@addfield\ifnum \@curtab<\@hightab
      \global\advance\@curtab \@ne \else\@badtab\fi
      \@tempdima\dimen\@curtab
      \advance\@tempdima -\dimen\@curtabmar
      \advance\@tempdima -\wd\@curline
      \global\setbox\@curline\hbox{\unhbox\@curline\hskip\@tempdima}%
      \@startfield\ignorespaces}
```

\@settab

```
\def\@settab{\@stopfield\@addfield
  \ifnum \@curtab <\@maxtab
    \ifnum\@curtab =\@hightab
      \advance\@hightab \@ne
    \fi
    \global\advance\@curtab \@ne
  \else
    \@latex@error{Tab overflow}\@ehd
  \fi
  \dimen\@curtab \dimen\@curtabmar
  \advance\dimen\@curtab \wd\@curline
  \@startfield
  \ignorespaces}
```

\@ltab

```
\def\@ltab{\@ifatmargin\ifnum\@curtabmar >\@firsttab
      \global\advance\@curtab \m@ne \global\advance\@curtabmar\m@ne\else
      \@badtab\fi\else
      \@latex@error{\string\<\space in mid line}\@ehd\fi\ignorespaces}
```

\@tabplus

```
\def\@tabplus{%
  \ifnum\@nxttabmar<\@hightab
    \global\advance\@nxttabmar\@ne
  \else
    \@badtab
  \fi
  \ignorespaces}
```

\@tabminus

```
\def\@tabminus{%
  \ifnum\@nxttabmar>\@firsttab
    \global\advance\@nxttabmar\m@ne
  \else
    \@badtab
  \fi
  \ignorespaces}
```

\@tabrj

```
\def\@tabrj{%
  \@stopfield\@addfield\global\@rjfieldtrue\@startfield\ignorespaces}
```

\@tablab \setbox\@curline made \global in \@tablab. 17 Jun 86

```
\def\@tablab{%
  \@stopfield
  \global\setbox\@curline\hbox{%
    \box\@curline
    \hskip-\wd\@curfield \hskip-\tabbingsep
    \box\@curfield
    \hskip\tabbingsep}%
  \@startfield
  \ignorespaces}
```

```
⟨/2ekernel⟩
⟨*2ekernel | latexrelease⟩
⟨latexrelease⟩\IncludeInRelease{2019/10/01}%
⟨latexrelease⟩                 {\pushtabs}{Make commands robust}%
```

\pushtabs

```
\DeclareRobustCommand\pushtabs{%
  \@stopfield\@addfield\global\advance\@tabpush \@ne \begingroup
       \@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.

```
\DeclareRobustCommand\poptabs{\@stopfield\@addfield
  \ifnum \@tabpush >\z@
    \endgroup
    \global\advance\@tabpush \m@ne
    \ifnum \@curtab >\@hightab
      \global \@curtab \@hightab
      \@badtab
    \fi
  \else
    \@badpoptabs
  \fi
  \@contfield}
```

```
\DeclareRobustCommand\kill{\@stopfield\@startline\ignorespaces}
```

```
⟨/2ekernel | latexrelease⟩
⟨latexrelease⟩\EndIncludeInRelease
⟨latexrelease⟩\IncludeInRelease{0000/00/00}%
⟨latexrelease⟩                 {\pushtabs}{Make commands robust}%
⟨latexrelease⟩
⟨latexrelease⟩\kernel@make@fragile\pushtabs
⟨latexrelease⟩\kernel@make@fragile\poptabs
⟨latexrelease⟩\kernel@make@fragile\kill
⟨latexrelease⟩
⟨latexrelease⟩\EndIncludeInRelease
⟨*2ekernel⟩
```

\tabbingsep

```
\newdimen\tabbingsep
```

67.2 array and tabular environments

Historical LaTeX 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  \crcr }}  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 \crcr}} $} 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

  \@tabularcr and \@argtabularcr  same as \@arraycr and
\@argarraycr
  except without the extra $'s.
```

End of historical LaTeX 2.09 comments.

`\extracolsep` This command needs to expand during the tabular preamble construction so can't be robust.

```
\def\extracolsep#1{\tabskip #1\relax}
```

`\array`

```
\def\array{\let\@acol\@arrayacol \let\@classz\@arrayclassz
 \let\@classiv\@arrayclassiv
 \let\\\@arraycr\let\@halignto\@empty\@tabarray}
```

`\endarray`
`\endtabular`
`\endtabular*`

```
\def\endarray{\crcr\egroup\egroup}
\def\endtabular{\crcr\egroup\egroup $\egroup}
\expandafter \let \csname endtabular*\endcsname = \endtabular
```

`\tabular`

```
\def\tabular{\let\@halignto\@empty\@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.

```
\@namedef{tabular*}#1{%
 \setlength\dimen@{#1}%
   \edef\@halignto{to\the\dimen@}\@tabular}
```

\@tabular

```
\def\@tabular{\leavevmode \hbox \bgroup $\let\@acol\@tabacol
   \let\@classz\@tabclassz
   \let\@classiv\@tabclassiv \let\\\@tabularcr\@tabarray}
```

\@tabarray RmS 91/11/04 added `\m@th`.

```
\def\@tabarray{\m@th\@ifnextchar[\@array{\@array[c]}}
```

RmS 1993/11/03 changed `\halign` to `\ialign` and removed superfluous `\tabskip` assignment

\@array

```
\def\@array[#1]#2{%
  \if #1t\vtop \else \if#1b\vbox \else \vcenter \fi\fi
```

```
  \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.

```
  \setbox\@arstrutbox\hbox{%
    \vrule \@height\arraystretch\ht\strutbox
           \@depth\arraystretch \dp\strutbox
           \@width\z@}%
  \@mkpream{#2}%
  \edef\@preamble{%
    \ialign \noexpand\@halignto
      \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.

```
  \let\@startpbox\@@startpbox \let\@endpbox\@@endpbox
  \let\tabularnewline\\%
    \let\par\@empty
    \let\@sharp##%
    \set@typeset@protect
    \lineskip\z@skip\baselineskip\z@skip
```

If the parsing of the preamble goes wrong there my be some characters left which TEX 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).

```
    \ifhmode \@preamerr\z@ \@@par\fi
    \@preamble}
```

\@arraycr Array version of \\.

```
\def\@arraycr{%
  ${\ifnum0=`}\fi\@ifstar\@xarraycr\@xarraycr}
```

\@arraycr

```
\def\@xarraycr{\@ifnextchar[\@argarraycr{\ifnum0=`{\fi}${}\cr}}
```

\@argarraycr

```
\def\@argarraycr[#1]{%
  \ifnum0=`{\fi}${}\ifdim #1>\z@ \@xargarraycr{#1}\else
   \@yargarraycr{#1}\fi}
```

\tabularnewline Tabular version of \\.

```
\let\tabularnewline\relax
```

\@tabularcr

```
\def\@tabularcr{%
  {\ifnum0=`}\fi\@ifstar\@xtabularcr\@xtabularcr}
```

\@xtabularcr

```
\def\@xtabularcr{\@ifnextchar[\@argtabularcr{\ifnum0=`{\fi}\cr}}
```

\@argtabularcr

```
\def\@argtabularcr[#1]{%
  \ifnum0=`{\fi}%
    \ifdim #1>\z@
      \unskip\@xargarraycr{#1}%
    \else
      \@yargarraycr{#1}%
    \fi}
```

\@xargarraycr

```
\def\@xargarraycr#1{\@tempdima #1\advance\@tempdima \dp \@arstrutbox
   \vrule \@height\z@ \@depth\@tempdima \@width\z@ \cr}
```

\@yargarraycr

```
\def\@yargarraycr#1{\cr\noalign{\vskip #1}}
```

\multicolumn *Historical LaTeX 2.09 comments (not necessarily accurate any more):*

```
 \multicolumn{NUMBER}{FORMAT}{ITEM} ==
  BEGIN
  \multispan{NUMBER}
  \begingroup
  \@addamp == null
  \@mkpream{FORMAT}
  \@sharp == ITEM
  \protect == \relax
  \@startpbox == \@@startpbox
  \@endpbox == \@@endpbox
  \@arstrut
  \@preamble
  \endgroup
  END
```

End of historical LaTeX 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.

```
\long\def\multicolumn#1#2#3{\multispan{#1}\begingroup
  \@mkpream{#2}%
  \def\@sharp{#3}\set@typeset@protect
  \let\@startpbox\@@startpbox\let\@endpbox\@@endpbox
  \@arstrut \@preamble\hbox{}\endgroup\ignorespaces}
```

Historical LaTeX 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 ->                                  % |
         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  \@xexpast S *0x\@@  END

\@xexpast 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 == \@xexpast
     else  \reserved@b == \@xexnoop
   fi
   \expandafter \reserved@b \reserved@a S3 \@@
 END
```

End of historical LATEX 2.09 comments.

\@xexnoop

```
\def\@xexnoop #1\@@{}
```

\@expast

```
\def\@expast#1{\@xexpast #1*0x\@@}
```

\@xexpast

```
\def\@xexpast#1*#2#3#4\@@{%
  \edef\reserved@a{#1}%
  \@tempcnta#2\relax
  \ifnum\@tempcnta>\z@
    \@whilenum\@tempcnta>\z@\do
       {\edef\reserved@a{\reserved@a#3}\advance\@tempcnta \m@ne}%
    \let\reserved@b\@xexpast
  \else
    \let\reserved@b\@xexnoop
  \fi
  \expandafter\reserved@b\reserved@a #4\@@}
```

\if@firstamp
\@addamp

```
\newif\if@firstamp
```

```
\def\@addamp{%
  \if@firstamp
    \@firstampfalse
  \else
    \edef\@preamble{\@preamble &}%
  \fi}
```

\@arrayacol
\@tabacol
\@ampacol
\@acolampacol

```
\def\@arrayacol{\edef\@preamble{\@preamble \hskip \arraycolsep}}
\def\@tabacol{\edef\@preamble{\@preamble \hskip \tabcolsep}}
\def\@ampacol{\@addamp \@acol}
\def\@acolampacol{\@acol\@addamp\@acol}
```

\@mkpream

```
\def\@mkpream#1{\@firstamptrue\@lastchclass6
  \let\@preamble\@empty
  \let\protect\@unexpandable@protect
  \let\@sharp\relax
  \let\@startpbox\relax\let\@endpbox\relax
  \@expast{#1}%
  \expandafter\@tfor \expandafter
    \@nextchar \expandafter:\expandafter=\reserved@a\do
       {\@testpach\@nextchar
    \ifcase \@chclass \@classz \or \@classi \or \@classii \or \@classiii
      \or \@classiv \or\@classv \fi\@lastchclass\@chclass}%
  \ifcase \@lastchclass \@acol
      \or \or \@preamerr \@ne\or \@preamerr \tw@\or \or \@acol \fi}
```

\@arrayclassz

```
\def\@arrayclassz{\ifcase \@lastchclass \@acolampacol \or \@ampacol \or
   \or \or \@addamp \or
   \@acolampacol \or \@firstampfalse \@acol \fi
\edef\@preamble{\@preamble
  \ifcase \@chnum
     \hfil$\relax\@sharp$\hfil \or $\relax\@sharp$\hfil
    \or \hfil$\relax\@sharp$\fi}}
```

\@tabclassz RmS 91/08/14 inserted extra braces around entry for NFSS

```
\def\@tabclassz{%
  \ifcase\@lastchclass
    \@acolampacol
  \or
    \@ampacol
  \or
  \or
  \or
    \@addamp
  \or
    \@acolampacol
  \or
    \@firstampfalse\@acol
  \fi
  \edef\@preamble{%
    \@preamble{%
      \ifcase\@chnum
        \hfil

        \hskip1sp%
        \ignorespaces\@sharp\unskip\hfil
      \or
```

```
        \hskip1sp\ignorespaces\@sharp\unskip\hfil
      \or
        \hfil\hskip1sp\ignorespaces\@sharp\unskip
      \fi}}
```

\@classi

```
\def\@classi{%
  \ifcase\@lastchclass
    \@acol\@arrayrule
  \or
    \@addtopreamble{\hskip \doublerulesep}\@arrayrule
  \or
  \or
  \or
    \@arrayrule
  \or
    \@acol\@arrayrule
  \or
    \@arrayrule
  \fi}
```

\@classii

```
\def\@classii{%
  \ifcase\@lastchclass
  \or
    \@addtopreamble{\hskip .5\arrayrulewidth}%
  \fi}
```

\@classiii

```
\def\@classiii{\ifcase \@lastchclass \@acolampacol \or
   \@addamp\@acol \or
   \or \or \@addamp \or
   \@acolampacol \or \@ampacol \fi}
```

\@tabclassiv

```
\def\@tabclassiv{\@addtopreamble\@nextchar}
```

\@arrayclassiv

```
\def\@arrayclassiv{\@addtopreamble{$\@nextchar$}}
```

\@classv

```
\def\@classv{\@addtopreamble{\@startpbox{\@nextchar}\ignorespaces
\@sharp\@endpbox}}
```

\@addtopreamble

```
\def\@addtopreamble#1{\edef\@preamble{\@preamble #1}}
```

\@chclass
\@lastchclass
\@chnum

```
\newcount\@chclass
\newcount\@lastchclass
\newcount\@chnum
```

\arraycolsep
\tabcolsep
\arrayrulewidth
\doublerulesep

```
\newdimen\arraycolsep
\newdimen\tabcolsep
\newdimen\arrayrulewidth
\newdimen\doublerulesep
```

\arraystretch

```
\def\arraystretch{1}    % Default value.
```

\@arstrutbox
\@arstrut

```
\newbox\@arstrutbox
\def\@arstrut{%
  \relax\ifmmode\copy\@arstrutbox\else\unhcopy\@arstrutbox\fi}
```

\@arrayrule

```
\def\@arrayrule{\@addtopreamble{\hskip -.5\arrayrulewidth
   \vrule \@width \arrayrulewidth\hskip -.5\arrayrulewidth}}
```

\@testpatch

```
\def\@testpach#1{\@chclass \ifnum \@lastchclass=\tw@ 4 \else
    \ifnum \@lastchclass=3 5 \else
     \z@ \if #1c\@chnum \z@ \else
                              \if #1l\@chnum \@ne \else
                              \if #1r\@chnum \tw@ \else
          \@chclass \if #1|\@ne \else
                    \if #1@\tw@ \else
                    \if #1p3 \else \z@ \@preamerr 0\fi
  \fi  \fi  \fi  \fi  \fi  \fi
\fi}
```

\hline

```
\def\hline{%
  \noalign{\ifnum0=`}\fi\hrule \@height \arrayrulewidth \futurelet
   \reserved@a\@xhline}
```

\@xhline

```
\def\@xhline{\ifx\reserved@a\hline
               \vskip\doublerulesep
```

Measure from the middle of the rules.

```
               \vskip-\arrayrulewidth
             \fi
      \ifnum0=`{\fi}}
```

\vline

```
\def\vline{\vrule \@width \arrayrulewidth}
```

\cline
\@cline

The old LaTeX2.09 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
```

```
\def\cline#1{\@cline#1\@nil}
```

```
\def\@cline#1-#2\@nil{%
  \omit
```

Use the counter from `\multispan`.

```
  \@multicnt#1%
  \advance\@multispan\m@ne
  \ifnum\@multicnt=\@ne\@firstofone{&\omit}\fi
  \@multicnt#2%
  \advance\@multicnt-#1%
  \advance\@multispan\@ne
```

The original had `\unskip` at this point, but how could a skip get here ???

```
  \leaders\hrule\@height\arrayrulewidth\hfill
  \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.

```
  \noalign{\vskip-\arrayrulewidth}}
```

`\mscount`

The `\mscount` counter is no longer declared, saving a csname and a register. It is declared in compatibility mode.

`\multispan`
`\@multispan`
`\sp@n`

Modify `\multispan` slightly from its plain TeX definition to allow more efficient code sharing with `\multicolumn`. Also share a count register with `\multiput`.

```
\def\multispan{\omit\@multispan}
```

```
\def\@multispan#1{%
  \@multicnt#1\relax
  \loop\ifnum\@multicnt>\@ne \sp@n\repeat}
```

```
\def\sp@n{\span\omit\advance\@multicnt\m@ne}
```

`\@startpbox`
`\@endpbox`

Helper macros for ‘p’ columns.

`\@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)

```
\def\@startpbox#1{\vtop\bgroup \setlength\hsize{#1}\@arrayparboxrestore}
```

```
\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.

`\@@startpbox`
`\@@endpbox`

```
\let\@@startpbox=\@startpbox
\let\@@endpbox=\@endpbox
```

```
⟨/2ekernel⟩
```

File F
ltpictur.dtx

68 Picture Mode

Picture mode commands. In addition to the commands available in LaTeX2.09, 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 LaTeX 2.09 comments (not necessarily accurate any more):

```
 \unitlength      = value of dimension argument
 \@wholewidth     = current line width
 \@halfwidth      = half of current line width
 \@linefnt        = font for drawing lines
 \@circlefnt      = 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 LaTeX 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.

```
⟨2ekernel⟩\expandafter\let\csname ver@autopict.sty\endcsname\fmtversion
```

`\@wholewidth`
`\@halfwidth`

```
⟨*2ekernel⟩
\newdimen\@wholewidth
\newdimen\@halfwidth
```

`\unitlength`

```
\newdimen\unitlength \unitlength =1pt
```

`\@picbox`
`\@picht`

```
\newbox\@picbox
\newdimen\@picht
```

`\picture` `#1` should be white space.

`\pictur@` `#1` should be a `(` (eating any white space before the bracket),

```
\long\def\picture#1{\pictur@#1}
\def\pictur@(#1){%
  \@ifnextchar({\@picture(#1)}{\@picture(#1)(0,0)}}
```

\@picture

```
\def\@picture(#1,#2)(#3,#4){%
  \@picht#2\unitlength
  \setbox\@picbox\hb@xt@#1\unitlength\bgroup
    \hskip -#3\unitlength
    \lower #4\unitlength\hbox\bgroup
      \ignorespaces}
```

\endpicture

```
\def\endpicture{%
  \egroup\hss\egroup
    \ht\@picbox\@picht\dp\@picbox\z@
    \mbox{\box\@picbox}}
```

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).

```
\long\def\put(#1,#2)#3{%
  \@killglue\raise#2\unitlength
  \hb@xt@\z@{\kern#1\unitlength #3\hss}%
  \ignorespaces}
```

\multiput `#3` had better be a `(`.

```
\def\multiput(#1,#2)#3{%
  \@xdim #1\unitlength
  \@ydim #2\unitlength
   \@multiput(}
```

\multiput

```
\long\def\@multiput(#1,#2)#3#4{%
  \@killglue\@multicnt #3\relax
  \@whilenum \@multicnt >\z@\do
    {\raise\@ydim\hb@xt@\z@{\kern\@xdim #4\hss}%
     \advance\@multicnt\m@ne
     \advance\@xdim#1\unitlength\advance\@ydim#2\unitlength}%
  \ignorespaces}
```

\@killglue

```
\def\@killglue{\unskip\@whiledim \lastskip >\z@\do{\unskip}}
```

\thinlines
\thicklines

```
\DeclareRobustCommand\thinlines{\let\@linefnt\tenln \let\@circlefnt\tencirc
  \@wholewidth\fontdimen8\tenln \@halfwidth .5\@wholewidth}
\DeclareRobustCommand\thicklines{\let\@linefnt\tenlnw \let\@circlefnt\tencircw
  \@wholewidth\fontdimen8\tenlnw \@halfwidth .5\@wholewidth}
```

\linethickness

```
\DeclareRobustCommand\linethickness[1]
   {\@wholewidth #1\relax \@halfwidth .5\@wholewidth}
```

\ishortstack

```
\def\shortstack{\@ifnextchar[\@shortstack{\@shortstack[c]}}
```

\@ishortstack

```
\def\@shortstack[#1]{%
  \leavevmode
  \vbox\bgroup
    \baselineskip-\p@\lineskip 3\p@
    \let\mb@l\hss\let\mb@r\hss
    \expandafter\let\csname mb@#1\endcsname\relax
    \let\\\@stackcr
    \@ishortstack}
```

\@ishortstack

```
\def\@ishortstack#1{\ialign{\mb@l {##}\unskip\mb@r\cr #1\crcr}\egroup}
```

\@stackcr
\@ixstackcr

```
\def\@stackcr{\@ifstar\@ixstackcr\@ixstackcr}
\def\@ixstackcr{\@ifnextchar[\@istackcr{\cr\ignorespaces}}
```

\@istackcr

```
\def\@istackcr[#1]{\cr\noalign{\vskip #1}\ignorespaces}
```

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{\@linefnt \@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 LaTeX 2.09 comments.

\if@negarg

```
\newif\if@negarg
```

\line

```
\def\line(#1,#2)#3{\@xarg #1\relax \@yarg #2\relax
  \@linelen #3\unitlength
  \ifdim\@linelen<\z@\@badlinearg\else
    \ifnum\@xarg =\z@ \@vline
      \else \ifnum\@yarg =\z@ \@hline \else \@sline\fi
    \fi
  \fi}
```

\@sline

```
\def\@sline{%
  \ifnum\@xarg<\z@ \@negargtrue \@xarg -\@xarg \@yyarg -\@yarg
  \else \@negargfalse \@yyarg \@yarg \fi
\ifnum \@yyarg >\z@ \@tempcnta\@yyarg \else \@tempcnta -\@yyarg \fi
\ifnum\@tempcnta>6 \@badlinearg\@tempcnta\z@ \fi
\ifnum\@xarg>6 \@badlinearg\@xarg \@ne \fi
\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.

```
\ifdim\wd\@linechar=\z@
   \setbox\@linechar\hbox{.}%
   \@badlinearg
\fi
\ifnum \@yarg >\z@ \let\@upordown\raise \@clnht\z@
   \else\let\@upordown\lower \@clnht \ht\@linechar\fi
\@clnwd \wd\@linechar
\if@negarg
  \hskip -\wd\@linechar \def\reserved@a{\hskip -2\wd\@linechar}%
\else
     \let\reserved@a\relax
\fi
\@whiledim \@clnwd <\@linelen \do
  {\@upordown\@clnht\copy\@linechar
   \reserved@a
   \advance\@clnht \ht\@linechar
   \advance\@clnwd \wd\@linechar}%
\advance\@clnht -\ht\@linechar
\advance\@clnwd -\wd\@linechar
\@tempdima\@linelen\advance\@tempdima -\@clnwd
\@tempdimb\@tempdima\advance\@tempdimb -\wd\@linechar
\if@negarg \hskip -\@tempdimb \else \hskip \@tempdimb \fi
\multiply\@tempdima \@m
\@tempcnta \@tempdima
\@tempdima \wd\@linechar \divide\@tempcnta \@tempdima
\@tempdima \ht\@linechar \multiply\@tempdima \@tempcnta
\divide\@tempdima \@m
\advance\@clnht \@tempdima
\ifdim \@linelen <\wd\@linechar
   \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).

```
   \ifdim \@linelen = \z@
   \else
     \@picture@warn
   \fi
   \else\@upordown\@clnht\copy\@linechar\fi}
```

\@hline

```
\def\@hline{\ifnum \@xarg <\z@ \hskip -\@linelen \fi
\vrule \@height \@halfwidth \@depth \@halfwidth \@width \@linelen
\ifnum \@xarg <\z@ \hskip -\@linelen \fi}
```

\@getlinechar

```
\def\@getlinechar(#1,#2){\@tempcnta#1\relax\multiply\@tempcnta 8%
  \advance\@tempcnta -9\ifnum #2>\z@ \advance\@tempcnta #2\relax\else
  \advance\@tempcnta -#2\relax\advance\@tempcnta 64 \fi
  \char\@tempcnta}
```

\vector

```
\def\vector(#1,#2)#3{\@xarg #1\relax \@yarg #2\relax
  \@tempcnta \ifnum\@xarg<\z@ -\@xarg\else\@xarg\fi
  \ifnum\@tempcnta<5\relax
  \@linelen #3\unitlength
  \ifdim\@linelen<\z@\@badlinearg\else
    \ifnum\@xarg =\z@ \@vvector
      \else \ifnum\@yarg =\z@ \@hvector \else \@svector\fi
    \fi
  \fi
  \else\@badlinearg\fi}
```

\@hvector

```
\def\@hvector{\@hline\hb@xt@\z@{\@linefnt
 \ifnum \@xarg <\z@ \@getlarrow(1,0)\hss\else
    \hss\@getrarrow(1,0)\fi}}
```

\@vvector

```
\def\@vvector{\ifnum \@yarg <\z@ \@downvector \else \@upvector \fi}
```

\@svector

```
\def\@svector{\@sline
  \@tempcnta\@yarg \ifnum\@tempcnta <\z@ \@tempcnta -\@tempcnta\fi
  \ifnum\@tempcnta <5%
    \hskip -\wd\@linechar
    \@upordown\@clnht \hbox{\@linefnt  \if@negarg
    \@getlarrow(\@xarg,\@yyarg)\else \@getrarrow(\@xarg,\@yyarg)\fi}%
  \else\@badlinearg\fi}
```

\@getlarrow

```
\def\@getlarrow(#1,#2){\ifnum #2=\z@ \@tempcnta 27 % '33
  \else
  \@tempcnta #1\relax\multiply\@tempcnta \sixt@@n
  \advance\@tempcnta -9 \@tempcntb #2\relax\multiply\@tempcntb \tw@
  \ifnum \@tempcntb >\z@ \advance\@tempcnta \@tempcntb
  \else\advance\@tempcnta -\@tempcntb\advance\@tempcnta 64
  \fi\fi\char\@tempcnta}
```

\@getrarrow

```
\def\@getrarrow(#1,#2){\@tempcntb #2\relax
\ifnum\@tempcntb <\z@ \@tempcntb -\@tempcntb\relax\fi
\ifcase \@tempcntb\relax \@tempcnta 45  % '55
\or
\ifnum #1<\thr@@ \@tempcnta #1\relax\multiply\@tempcnta
24\advance\@tempcnta -6 \else \ifnum #1=\thr@@ \@tempcnta 49
\else\@tempcnta 58 \fi\fi\or
\ifnum #1<\thr@@ \@tempcnta=#1\relax\multiply\@tempcnta
24\advance\@tempcnta -\thr@@ \else \@tempcnta 51 \fi\or
\@tempcnta #1\relax\multiply\@tempcnta
\sixt@@n \advance\@tempcnta -\tw@ \else
\@tempcnta #1\relax\multiply\@tempcnta
\sixt@@n \advance\@tempcnta 7 \fi\ifnum #2<\z@ \advance\@tempcnta 64 \fi
\char\@tempcnta}
```

\@vline

```
\def\@vline{\ifnum \@yarg <\z@ \@downline \else \@upline\fi}
```

\@upline

```
\def\@upline{%
  \hb@xt@\z@{\hskip -\@halfwidth \vrule \@width \@wholewidth
   \@height \@linelen \@depth \z@\hss}}
```

\@downline

```
\def\@downline{%
  \hb@xt@\z@{\hskip -\@halfwidth \vrule \@width \@wholewidth
   \@height \z@ \@depth \@linelen \hss}}
```

\@upvector

```
\def\@upvector{\@upline\setbox\@tempboxa\hbox{\@linefnt\char 54}% '66
  \raise \@linelen \hb@xt@\z@{\lower \ht\@tempboxa\box\@tempboxa\hss}}
```

\@downvector

```
\def\@downvector{\@downline\lower \@linelen
      \hb@xt@\z@{\@linefnt\char 63  % '77
      \hss}}
```

Historical LaTeX 2.09 comments (not necessarily accurate any more):

```
\dashbox{D}(X,Y) ==
 BEGIN
 leave vertical mode
 \hb@xt@ 0pt {
      \baselineskip := 0pt
      \lineskip     := 0pt
 %% HORIZONTAL DASHES
      \@dashdim := X * \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{\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 < \@dascnt
                 do \copy\@dashbox
                    \@tempcnta := \@tempcnta + 1
                 od
              }
      \@tempcnta := 0
      put(0,Y){\hskip \@dashdim
               while \@tempcnta < \@dascnt
                 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

 \@imakepicbox(X,Y)
END
```

End of historical LaTeX 2.09 comments.

\dashbox

```
\def\dashbox#1(#2,#3){\leavevmode\hb@xt@\z@{\baselineskip \z@skip
\lineskip \z@skip
\@dashdim #2\unitlength
\@dashcnt \@dashdim \advance\@dashcnt 200
\@dashdim #1\unitlength\divide\@dashcnt \@dashdim
\ifodd\@dashcnt\@dashdim \z@
\advance\@dashcnt \@ne \divide\@dashcnt \tw@
\else \divide\@dashdim \tw@ \divide\@dashcnt \tw@
\advance\@dashcnt \m@ne
\setbox\@dashbox \hbox{\vrule \@height \@halfwidth \@depth \@halfwidth
\@width \@dashdim}\put(0,0){\copy\@dashbox}%
\put(0,#3){\copy\@dashbox}%
\put(#2,0){\hskip-\@dashdim\copy\@dashbox}%
\put(#2,#3){\hskip-\@dashdim\box\@dashbox}%
\multiply\@dashdim \thr@@
\fi
\setbox\@dashbox \hbox{\vrule \@height \@halfwidth \@depth \@halfwidth
\@width #1\unitlength\hskip #1\unitlength}\@tempcnta\z@
\put(0,0){\hskip\@dashdim \@whilenum \@tempcnta <\@dashcnt
\do{\copy\@dashbox\advance\@tempcnta \@ne }}\@tempcnta\z@
\put(0,#3){\hskip\@dashdim \@whilenum \@tempcnta <\@dashcnt
\do{\copy\@dashbox\advance\@tempcnta \@ne }}%
\@dashdim #3\unitlength
\@dashcnt \@dashdim \advance\@dashcnt 200
\@dashdim #1\unitlength\divide\@dashcnt \@dashdim
\ifodd\@dashcnt \@dashdim \z@
\advance\@dashcnt \@ne \divide\@dashcnt \tw@
\else
\divide\@dashdim \tw@ \divide\@dashcnt \tw@
\advance\@dashcnt \m@ne
\setbox\@dashbox\hbox{\hskip -\@halfwidth
\vrule \@width \@wholewidth
\@height \@dashdim}\put(0,0){\copy\@dashbox}%
\put(#2,0){\copy\@dashbox}%
\put(0,#3){\lower\@dashdim\copy\@dashbox}%
\put(#2,#3){\lower\@dashdim\copy\@dashbox}%
\multiply\@dashdim \thr@@
\fi
\setbox\@dashbox\hbox{\vrule \@width \@wholewidth
\@height #1\unitlength}\@tempcnta\z@
\put(0,0){\hskip -\@halfwidth \vbox{\@whilenum \@tempcnta <\@dashcnt
\do{\vskip #1\unitlength\copy\@dashbox\advance\@tempcnta \@ne }%
\vskip\@dashdim}}\@tempcnta\z@
\put(#2,0){\hskip -\@halfwidth \vbox{\@whilenum \@tempcnta<\@dashcnt
\do{\vskip #1\unitlength\copy\@dashbox\advance\@tempcnta \@ne }%
\vskip\@dashdim}}\@makepicbox(#2,#3)}
```

Historical LaTeX 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 * `\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 * `\unitlength` and height
Y * `\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{\@circlefnt \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
    \begingroup
      \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      := \@ovyy/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@ \@ovxx{
                  \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 LaTeX 2.09 comments.

\if@ovt \if@ovb \if@ovl \if@ovr

If producing the Top Bottom Left or Right of an oval.

```
\newif\if@ovt
\newif\if@ovb
\newif\if@ovl
\newif\if@ovr
```

\@ovxx \@ovyy \@ovdx \@ovdy \@ovro \@ovri

```
\newdimen\@ovxx
\newdimen\@ovyy
\newdimen\@ovdx
\newdimen\@ovdy
\newdimen\@ovro
\newdimen\@ovri
```

`\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

```
\def\@getcirc#1{\@tempdima #1\relax \advance\@tempdima 2\p@
  \@tempcnta\@tempdima
  \@tempdima 4\p@ \divide\@tempcnta\@tempdima
  \ifnum \@tempcnta >10\relax
      \@picture@warn
      \@tempcnta 10\relax
  \fi
  \ifnum \@tempcnta >\z@ \advance\@tempcnta\m@ne
```

Warn if requirements for oval or circle can't be met.

```
    \else \@picture@warn \fi
  \multiply\@tempcnta 4\relax
  \setbox \@tempboxa \hbox{\@circlefnt
  \char \@tempcnta}\@tempdima \wd \@tempboxa}
```

\@picture@warn

Generic warning for lines, vectors (used in `\@sline`) and oval or circle (used in `\@getcirc`) are not available at right size.

```
\def\@picture@warn{\@latex@warning{%
     \string\oval, \string\circle, or \string\line\space
     size unavailable}}
```

\@put

```
\def\@put#1#2#3{\raise #2\hb@xt@\z@{\hskip #1#3\hss}}
```

\oval

```
\def\oval(#1,#2){\@ifnextchar[{\@oval(#1,#2)}{\@oval(#1,#2)[]}}
```

```
⟨/2ekernel⟩
⟨latexrelease⟩\IncludeInRelease{2016/03/31}%
⟨latexrelease⟩                 {\@ovhlinetrue}%
⟨latexrelease⟩                 {Avoid almost zero length leaders}%
⟨*2ekernel | latexrelease⟩
```

\if@ovvline \if@ovhline

Tests whether horizontal or vertical lines are needed.

```
\newif\if@ovvline \@ovvlinetrue
\newif\if@ovhline \@ovhlinetrue
```

\@oval

```
\def\@oval(#1,#2)[#3]{\begingroup\boxmaxdepth \maxdimen
  \@ovttrue \@ovbtrue \@ovltrue \@ovrtrue

  \@ovvlinefalse \@ovhlinefalse

  \@tfor\reserved@a :=#3\do{\csname @ov\reserved@a false\endcsname}%
  \@ovxx #1\unitlength
  \@ovyy #2\unitlength

  \@tempdimb \ifdim \@ovyy >\@ovxx \@ovxx \@ovvlinetrue
  \else \@ovyy \ifdim \@ovyy =\@ovxx \else \@ovhlinetrue \fi\fi

  \advance \@tempdimb -2\p@
  \@getcirc \@tempdimb
  \@ovro \ht\@tempboxa \@ovri \dp\@tempboxa
  \@ovdx\@ovxx \advance\@ovdx -\@tempdima \divide\@ovdx \tw@
  \@ovdy\@ovyy \advance\@ovdy -\@tempdima \divide\@ovdy \tw@

  \ifdim \@ovdx >\z@ \@ovhlinetrue \fi
  \ifdim \@ovdy >\z@ \@ovvlinetrue \fi

  \@circlefnt \setbox\@tempboxa
  \hbox{\if@ovr \@ovvert32\kern -\@tempdima \fi
  \if@ovl \kern \@ovxx \@ovvert01\kern -\@tempdima \kern -\@ovxx \fi
  \if@ovt \@ovhorz \kern -\@ovxx \fi
  \if@ovb \raise \@ovyy \@ovhorz \fi}\advance\@ovdx\@ovro
  \advance\@ovdy\@ovro \ht\@tempboxa\z@ \dp\@tempboxa\z@
  \@put{-\@ovdx}{-\@ovdy}{\box\@tempboxa}%
  \endgroup}
```

\@ovvert

```
\def\@ovvert#1#2{\vbox to\@ovyy{%
    \if@ovb \@tempcntb \@tempcnta \advance \@tempcntb #1\relax
      \kern -\@ovro \hbox{\char \@tempcntb}\nointerlineskip
    \else \kern \@ovri \kern \@ovdy \fi

    \if@ovvline \leaders\vrule \@width \@wholewidth \fi

    \vfil \nointerlineskip
    \if@ovt \@tempcntb \@tempcnta \advance \@tempcntb #2\relax
      \hbox{\char \@tempcntb}%
    \else \kern \@ovdy \kern \@ovro \fi}}
```

\@ovhorz

```
\def\@ovhorz{\hb@xt@\@ovxx{\kern \@ovro
    \if@ovr \else \kern \@ovdx \fi

    \if@ovhline \leaders \hrule \@height \@wholewidth \fi

    \hfil
    \if@ovl \else \kern \@ovdx \fi
    \kern \@ovri}}

⟨/2ekernel | latexrelease⟩
⟨latexrelease⟩\EndIncludeInRelease
⟨latexrelease⟩\IncludeInRelease{0000/00/00}%
⟨latexrelease⟩                    {\@ovhlinetrue}%
```

```
⟨latexrelease⟩                  {Avoid almost zero length leaders}%
⟨latexrelease⟩\let\if@ovvline\@undefined
⟨latexrelease⟩\let\if@ovhline\@undefined
⟨latexrelease⟩\def\@oval(#1,#2)[#3]{\begingroup\boxmaxdepth \maxdimen
⟨latexrelease⟩  \@ovttrue \@ovbtrue \@ovltrue \@ovrtrue
⟨latexrelease⟩  \@tfor\reserved@a :=#3\do
⟨latexrelease⟩                 {\csname @ov\reserved@a false\endcsname}%
⟨latexrelease⟩  \@ovxx #1\unitlength
⟨latexrelease⟩  \@ovyy #2\unitlength
⟨latexrelease⟩  \@tempdimb \ifdim \@ovyy >\@ovxx \@ovxx\else \@ovyy \fi
⟨latexrelease⟩  \advance \@tempdimb -2\p@
⟨latexrelease⟩  \@getcirc \@tempdimb
⟨latexrelease⟩  \@ovro \ht\@tempboxa \@ovri \dp\@tempboxa
⟨latexrelease⟩  \@ovdx\@ovxx \advance\@ovdx -\@tempdima \divide\@ovdx \tw@
⟨latexrelease⟩  \@ovdy\@ovyy \advance\@ovdy -\@tempdima \divide\@ovdy \tw@
⟨latexrelease⟩  \@circlefnt \setbox\@tempboxa
⟨latexrelease⟩  \hbox{\if@ovr \@ovvert32\kern -\@tempdima \fi
⟨latexrelease⟩  \if@ovl
⟨latexrelease⟩   \kern \@ovxx \@ovvert01\kern -\@tempdima \kern -\@ovxx
⟨latexrelease⟩  \fi
⟨latexrelease⟩  \if@ovt \@ovhorz \kern -\@ovxx \fi
⟨latexrelease⟩  \if@ovb \raise \@ovyy \@ovhorz \fi}\advance\@ovdx\@ovro
⟨latexrelease⟩  \advance\@ovdy\@ovro \ht\@tempboxa\z@ \dp\@tempboxa\z@
⟨latexrelease⟩  \@put{-\@ovdx}{-\@ovdy}{\box\@tempboxa}%
⟨latexrelease⟩  \endgroup}
⟨latexrelease⟩\def\@ovvert#1#2{\vbox to\@ovyy{%
⟨latexrelease⟩    \if@ovb \@tempcntb \@tempcnta \advance \@tempcntb #1\relax
⟨latexrelease⟩      \kern -\@ovro \hbox{\char \@tempcntb}\nointerlineskip
⟨latexrelease⟩    \else \kern \@ovri \kern \@ovdy \fi
⟨latexrelease⟩    \leaders\vrule \@width \@wholewidth\vfil \nointerlineskip
⟨latexrelease⟩    \if@ovt \@tempcntb \@tempcnta \advance \@tempcntb #2\relax
⟨latexrelease⟩      \hbox{\char \@tempcntb}%
⟨latexrelease⟩    \else \kern \@ovdy \kern \@ovro \fi}}
⟨latexrelease⟩\def\@ovhorz{\hb@xt@\@ovxx{\kern \@ovro
⟨latexrelease⟩    \if@ovr \else \kern \@ovdx \fi
⟨latexrelease⟩    \leaders \hrule \@height \@wholewidth \hfil
⟨latexrelease⟩    \if@ovl \else \kern \@ovdx \fi
⟨latexrelease⟩    \kern \@ovri}}
⟨latexrelease⟩\EndIncludeInRelease
⟨*2ekernel⟩
```

\circle

```
\def\circle{\@inmatherr\circle\@ifstar\@dot\@circle}
```

\@circle

```
\def\@circle#1{%
  \begingroup \boxmaxdepth \maxdimen \@tempdimb #1\unitlength
   \ifdim \@tempdimb >15.5\p@ \@getcirc\@tempdimb
      \@ovro\ht\@tempboxa
     \setbox\@tempboxa\hbox{\@circlefnt
      \advance\@tempcnta\tw@ \char \@tempcnta
      \advance\@tempcnta\m@ne \char \@tempcnta \kern -2\@tempdima
      \advance\@tempcnta\tw@
      \raise \@tempdima \hbox{\char\@tempcnta}\raise \@tempdima
```

```
        \box\@tempboxa}\ht\@tempboxa\z@ \dp\@tempboxa\z@
      \@put{-\@ovro}{-\@ovro}{\box\@tempboxa}%
   \else  \@circ\@tempdimb{96}\fi\endgroup}
```

\@dot Internal form of \circle*.

```
\def\@dot#1{\@tempdimb #1\unitlength \@circ\@tempdimb{112}}
```

\@circ

```
\def\@circ#1#2{\@tempdima #1\relax \advance\@tempdima .5\p@
   \@tempcnta\@tempdima \@tempdima \p@
   \divide\@tempcnta\@tempdima
   \ifnum\@tempcnta >15\relax \@tempcnta 15\relax \fi
   \ifnum \@tempcnta >\z@ \advance\@tempcnta\m@ne\fi
   \advance\@tempcnta #2\relax
   \@circlefnt \char\@tempcnta}
```

\@xarg
\@yarg
\@yyarg
Counters used for manipulating the 'slope' arguments.

```
\newcount\@xarg
\newcount\@yarg
\newcount\@yyarg
```

\@multicnt Counter used in \multiput, and also \multicolumn.

```
\newcount\@multicnt
```

\@xdim
\yxdim
Length registers.

```
\newdimen\@xdim
\newdimen\@ydim
```

\@linechar Box for holding a line segment character, for sloping lines.

```
\newbox\@linechar
```

\@linelen Length of the line currently being built.

```
\newdimen\@linelen
```

\@clnwd
\@clnht
Height and width of current line segment.

```
\newdimen\@clnwd
\newdimen\@clnht
```

\@dashdim
\@dashbox
\@dashcnt
\dashbox internal registers.

```
\newdimen\@dashdim
\newbox\@dashbox
\newcount\@dashcnt
```

Initialization: "\thinlines"

```
\let\@linefnt\tenln
\let\@circlefnt\tencirc
\@wholewidth\fontdimen8\tenln
\@halfwidth .5\@wholewidth
```

68.1 Curves

The new `\qbezier` command, based on the old `\bezier` defined in `bezier.sty`. *Historical LaTeX 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 LaTeX 2.09 comments.

`\qbeziermax` The maximum number of points to plot.

```
367 \def\qbeziermax{500}
```

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 (.

```
\newcommand\qbezier[2][0]{\bezier{#1}#2}
```

`\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 (.

```
\def\bezier#1)#2(#3)#4({\@bezier#1)(#3)(}
```

`\@bezier`

```
\def\@bezier#1(#2,#3)(#4,#5)(#6,#7){%
  \ifnum #1=\z@
      \@ovxx #4\unitlength
        \advance\@ovxx -#2\unitlength
        \ifdim \@ovxx<\z@ \@ovxx -\@ovxx \fi
      \@ovdx #6\unitlength
        \advance\@ovdx -#4\unitlength
        \ifdim \@ovdx<\z@ \@ovdx -\@ovdx \fi
        \ifdim \@ovxx<\@ovdx \@ovxx \@ovdx \fi
      \@ovyy #5\unitlength
        \advance\@ovyy -#3\unitlength
        \ifdim \@ovyy<\z@ \@ovyy -\@ovyy \fi
      \@ovdy #7\unitlength
        \advance\@ovdy -#5\unitlength
        \ifdim \@ovdy<\z@  \@ovdy -\@ovdy \fi
        \ifdim \@ovyy<\@ovdy \@ovyy  \@ovdy \fi
      \@multicnt
         \ifdim \@ovxx>\@ovyy \@ovxx \else \@ovyy \fi
      \@ovxx .5\@halfwidth \divide\@multicnt\@ovxx
      \ifnum \qbeziermax<\@multicnt \@multicnt\qbeziermax\relax \fi
  \else \@multicnt#1\relax \fi
  \@tempcnta\@multicnt \advance\@tempcnta\@ne
  \@ovdx #4\unitlength \advance\@ovdx -#2\unitlength
      \multiply\@ovdx \tw@
  \@ovxx #6\unitlength \advance\@ovxx -#2\unitlength
      \advance\@ovxx -\@ovdx \divide\@ovxx\@multicnt
  \@ovdy #5\unitlength \advance\@ovdy -#3\unitlength
       \multiply\@ovdy \tw@
  \@ovyy #7\unitlength \advance\@ovyy -#3\unitlength
      \advance\@ovyy -\@ovdy \divide\@ovyy\@multicnt

  \setbox\@tempboxa\hbox{%
            \hskip -\@halfwidth
            \vrule \@height\@halfwidth
                  \@depth \@halfwidth
                  \@width \@wholewidth}%
   \put(#2,#3){%
     \count@\z@
     \@whilenum{\count@<\@tempcnta}\do
        {\@xdim\count@\@ovxx
           \advance\@xdim\@ovdx
           \divide\@xdim\@multicnt
           \multiply\@xdim\count@
         \@ydim\count@\@ovyy
            \advance\@ydim\@ovdy
            \divide\@ydim\@multicnt
            \multiply\@ydim\count@
         \raise \@ydim
            \hb@xt@\z@{\kern\@xdim
                        \unhcopy\@tempboxa\hss}%
         \advance\count@\@ne}}
```

As the commands above all use “picture” interface we couldn’t define them with `\DeclareRobustCommand` so we do that now.

```
⟨/2ekernel⟩
⟨*2ekernel | latexrelease⟩
⟨latexrelease⟩\IncludeInRelease{2019/10/01}%
⟨latexrelease⟩                 {\bezier}{Make commands robust}%
\MakeRobust\bezier
\MakeRobust\circle
\MakeRobust\dashbox
\MakeRobust\line
\MakeRobust\linethickness
\MakeRobust\multiput
\MakeRobust\oval
\MakeRobust\put
\MakeRobust\qbezier
\MakeRobust\shortstack
\MakeRobust\thinlines
\MakeRobust\vector
⟨/2ekernel | latexrelease⟩
⟨latexrelease⟩\EndIncludeInRelease
⟨latexrelease⟩\IncludeInRelease{0000/00/00}%
⟨latexrelease⟩                 {\bezier}{Make commands robust}%
⟨latexrelease⟩
⟨latexrelease⟩\kernel@make@fragile\bezier
⟨latexrelease⟩\kernel@make@fragile\circle
⟨latexrelease⟩\kernel@make@fragile\dashbox
⟨latexrelease⟩\kernel@make@fragile\line
⟨latexrelease⟩\kernel@make@fragile\linethickness
⟨latexrelease⟩\kernel@make@fragile\multiput
⟨latexrelease⟩\kernel@make@fragile\oval
⟨latexrelease⟩\kernel@make@fragile\put
⟨latexrelease⟩\kernel@make@fragile\qbezier
⟨latexrelease⟩\kernel@make@fragile\shortstack
⟨latexrelease⟩\kernel@make@fragile\thinlines
⟨latexrelease⟩\kernel@make@fragile\vector
⟨latexrelease⟩
⟨latexrelease⟩\EndIncludeInRelease
⟨*2ekernel⟩
```

```
⟨/2ekernel⟩
```

File G
ltthm.dtx

69 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 ⟨*name*⟩ to be just as one would expect a theorem environment to be, except that it prints ⟨*text*⟩ instead of "Theorem".

If ⟨*oldname*⟩ is given, then environments ⟨*name*⟩ and ⟨*oldname*⟩ use the same counter, so using a ⟨*name*⟩ environment advances the number of the next ⟨*name*⟩ environment, and vice-versa.

If ⟨*counter*⟩ is given, then environment ⟨*name*⟩ is numbered within ⟨*counter*⟩.

E.g., if ⟨*counter*⟩ = `subsection`, then the first ⟨*name*⟩ in subsection 7.2 is numbered ⟨*text*⟩ 7.2.1.

The way ⟨*name*⟩ environments are numbered can be changed by redefining `\the`⟨*name*⟩.

Historical LaTeX 2.09 comments (not necessarily accurate any more):

```
 DOCUMENT STYLE PARAMETERS

 \@thmcounter{COUNTER} : A command such that
               \edef\theCOUNTER{\@thmcounter{COUNTER}}
    defines \theCOUNTER to produce a number for a theorem environment.
    The default is:
           BEGIN \noexpand\arabic{COUNTER} END

 \@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, \@thmcountersep should be \def'ed to '-'.  Its
         default is '.'.

 \@begintheorem{NAME}{NUMBER} : A command that begins a
theorem
         environment for a 'theorem' named 'NAME NUMBER' –
         e.g., \@begintheorem{Lemma}{3.7} starts Lemma 3.7.

 \@opargbegintheorem{NAME}{NUMBER}{OPARG} :
         A command that begins a theorem
         environment for a 'theorem' named 'NAME NUMBER' with
optional
         argument OPARG – e.g., \@begintheorem{Lemma}{3.7}{Jones}
         starts 'Lemma 3.7 (Jones):'.

 \@endtheorem : A command that ends a theorem environment.
```

```
\newtheorem{NAME}{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 LaTeX 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 LaTeX book indicates that \newtheorem may be used anywhere in the document...

```
⟨*2ekernel⟩
\def\newtheorem#1{%
  \@ifnextchar[{\@othm{#1}}{\@nthm{#1}}}
```

\@nthm

```
\def\@nthm#1#2{%
  \@ifnextchar[{\@xnthm{#1}{#2}}{\@ynthm{#1}{#2}}}
```

\@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)

```
\def\@xnthm#1#2[#3]{%
  \expandafter\@ifdefinable\csname #1\endcsname
    {\@definecounter{#1}\@newctr{#1}[#3]%
     \expandafter\xdef\csname the#1\endcsname{%
       \expandafter\noexpand\csname the#3\endcsname \@thmcountersep
          \@thmcounter{#1}}%
     \global\@namedef{#1}{\@thm{#1}{#2}}%
     \global\@namedef{end#1}{\@endtheorem}}}
```

\@ynthm

```
\def\@ynthm#1#2{%
  \expandafter\@ifdefinable\csname #1\endcsname
    {\@definecounter{#1}%
     \expandafter\xdef\csname the#1\endcsname{\@thmcounter{#1}}%
     \global\@namedef{#1}{\@thm{#1}{#2}}%
     \global\@namedef{end#1}{\@endtheorem}}}
```

\@othm

```
\def\@othm#1[#2]#3{%
  \@ifundefined{c@#2}{\@nocounterr{#2}}%
    {\expandafter\@ifdefinable\csname #1\endcsname
    {\global\@namedef{the#1}{\@nameuse{the#2}}%
  \global\@namedef{#1}{\@thm{#2}{#3}}%
  \global\@namedef{end#1}{\@endtheorem}}}}
```

\@thm

```
\def\@thm#1#2{%
  \refstepcounter{#1}%
  \@ifnextchar[{\@ythm{#1}{#2}}{\@xthm{#1}{#2}}}
```

\@xthm
\@ythm

```
\def\@xthm#1#2{%
  \@begintheorem{#2}{\csname the#1\endcsname}\ignorespaces}
\def\@ythm#1#2[#3]{%
  \@opargbegintheorem{#2}{\csname the#1\endcsname}{#3}\ignorespaces}
```

Default values

`\@thmcounter`
`\@thmcountersep`

```
\def\@thmcounter#1{\noexpand\arabic{#1}}
\def\@thmcountersep{.}
```

`\@begintheorem`
`\@opargbegintheorem`
`\@endtheorem`

Providing theorem defaults.

```
\def\@begintheorem#1#2{\trivlist
   \item[\hskip \labelsep{\bfseries #1\ #2}]\itshape}
\def\@opargbegintheorem#1#2#3{\trivlist
      \item[\hskip \labelsep{\bfseries #1\ #2\ (#3)}]\itshape}
\def\@endtheorem{\endtrivlist}
⟨/2ekernel⟩
```

File H
ltsect.dtx

70 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 LaTeX 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.

```
⟨*2ekernel⟩
\message{title,}
```

70.1 The Title

`\title` `\author` `\date` `\thanks` `\and`

The user defines the title and author by the declarations `\title{`⟨*name*⟩`}`, `\author{`⟨*name*⟩`}`

Similarly the date is declared with `\date{`⟨*date*⟩`}`.

Inside these, the `\thanks{`⟨*footnote text*⟩`}` command may be used to make acknowledgements, notice of address, etc. in a footnote. If there are multiple authors, they have to be separated with the `\and` command.

`\maketitle`

And finally, the `\maketitle` command produces the actual title, using the information previously saved with the other commands.

```
⟨/2ekernel⟩
⟨*2ekernel | latexrelease⟩
⟨latexrelease⟩\IncludeInRelease{2019/10/01}%
⟨latexrelease⟩                    {\title}{Make commands robust}%
```

`\title`

`\title` for use in `\maketitle`. If not given `\maketitle` will produce an error message.

```
\DeclareRobustCommand\title[1]{\gdef\@title{#1}}
```

`\author`

`\author` for use in `\maketitle`. If not given `\maketitle` will produce a warning message.

```
\DeclareRobustCommand\author[1]{\gdef\@author{#1}}
```

`\date`

`\date` for use in `\maketitle`. If not given `\maketitle` will produce `\today` as the default.

```
\DeclareRobustCommand\date[1]{\gdef\@date{#1}}
```

`\thanks`

```
\DeclareRobustCommand\thanks[1]{\footnotemark
    \protected@xdef\@thanks{\@thanks
        \protect\footnotetext[\the\c@footnote]{#1}}%
}
```

\and

```
\DeclareRobustCommand\and{%   % \begin{tabular}
  \end{tabular}%
  \hskip 1em \@plus.17fil%
  \begin{tabular}[t]{c}}%     % \end{tabular}
```

```
⟨/2ekernel | latexrelease⟩
⟨latexrelease⟩\EndIncludeInRelease
⟨latexrelease⟩\IncludeInRelease{0000/00/00}%
⟨latexrelease⟩                 {\title}{Make commands robust}%
⟨latexrelease⟩
⟨latexrelease⟩\kernel@make@fragile\title
⟨latexrelease⟩\kernel@make@fragile\author
⟨latexrelease⟩\kernel@make@fragile\date
⟨latexrelease⟩\kernel@make@fragile\thanks
⟨latexrelease⟩\kernel@make@fragile\and
⟨latexrelease⟩
⟨latexrelease⟩\EndIncludeInRelease
⟨*2ekernel⟩
```

\@title

```
\def\@title{\@latex@error{No \noexpand\title given}\@ehc}
```

\@author

```
\def\@author{\@latex@warning@no@line{No \noexpand\author given}}
```

\@date

```
\gdef\@date{\today}
```

\@thanks

```
\let\@thanks\@empty
```

```
\message{sectioning,}
```

70.2 Sectioning

\@secpenalty

```
\newcount\@secpenalty
\@secpenalty = -300
```

\if@noskipsec
\@noskipsectrue

Way back in 1991 (08/26) FMi & RmS set the `\@noskipsec` switch to true for the preamble and to false in `\document`. This was done to trap lists and related text in the preamble but it does not catch everything.

```
\newif\if@noskipsec \@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 LaTeX 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.

```
\def\@startsection#1#2#3#4#5#6{%
  \if@noskipsec \leavevmode \fi
  \par
  \@tempskipa #4\relax
  \@afterindenttrue
  \ifdim \@tempskipa <\z@
    \@tempskipa -\@tempskipa \@afterindentfalse
  \fi
  \if@nobreak
    \everypar{}%
  \else
    \addpenalty\@secpenalty\addvspace\@tempskipa
  \fi
  \@ifstar
    {\@ssect{#3}{#4}{#5}{#6}}%
    {\@dblarg{\@sect{#1}{#2}{#3}{#4}{#5}{#6}}}}
```

\@sect Pseudocode for the \@sect command *Historical LaTeX 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.

```
\def\@sect#1#2#3#4#5#6[#7]#8{%
  \ifnum #2>\c@secnumdepth
    \let\@svsec\@empty
  \else
    \refstepcounter{#1}%
```

Since `\@seccntformat` 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.

```
    \protected@edef\@svsec{\@seccntformat{#1}\relax}%
  \fi
  \@tempskipa #5\relax
  \ifdim \@tempskipa>\z@
    \begingroup
```

This `{` used to be after the argument to `\@hangfrom` but was moved here to allow commands such as `\MakeUppercase` to be used at the end of `#6`.

```
      #6{%
        \@hangfrom{\hskip #3\relax\@svsec}%
          \interlinepenalty \@M #8\@@par}%
    \endgroup
    \csname #1mark\endcsname{#7}%
    \addcontentsline{toc}{#1}{%
      \ifnum #2>\c@secnumdepth \else
        \protect\numberline{\csname the#1\endcsname}%
      \fi
      #7}%
  \else
```

`\relax` added 2 May 90

```
    \def\@svsechd{%
      #6{\hskip #3\relax
      \@svsec #8}%
      \csname #1mark\endcsname{#7}%
      \addcontentsline{toc}{#1}{%
        \ifnum #2>\c@secnumdepth \else
          \protect\numberline{\csname the#1\endcsname}%
        \fi
        #7}}%
  \fi
  \@xsect{#5}}
```

`\@xsect` Pseudocode for the `\@xsect` command *Historical LaTeX 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

  END
```

End of historical LaTeX 2.09 comments.

```
\def\@xsect#1{%
  \@tempskipa #1\relax
  \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.

```
    \par \nobreak
    \vskip \@tempskipa
    \@afterheading
  \else
    \@nobreakfalse
    \global\@noskipsectrue
    \everypar{%
      \if@noskipsec
        \global\@noskipsecfalse
       {\setbox\z@\lastbox}%
        \clubpenalty\@M
        \begingroup \@svsechd \endgroup
        \unskip
        \@tempskipa #1\relax
        \hskip -\@tempskipa
      \else
        \clubpenalty \@clubpenalty
        \everypar{}%
      \fi}%
  \fi
  \ignorespaces}
```

`\@seccntformat` This command formats the section number including the space following it.

```
\def\@seccntformat#1{\csname the#1\endcsname\quad}
```

Pseudocode for the `\@ssect` command *Historical LaTeX 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 LaTeX 2.09 comments.

Pseudocode for the `\@afterheading` command *Historical LaTeX 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 LaTeX 2.09 comments.

`\@ssect`

```
\def\@ssect#1#2#3#4#5{%
  \@tempskipa #3\relax
  \ifdim \@tempskipa>\z@
    \begingroup
```

This `{` used to be after the argument to `\@hangfrom` but was moved here to allow commands such as `\MakeUppercase` to be used at the end of `#4`.

```
      #4{%
        \@hangfrom{\hskip #1}%
          \interlinepenalty \@M #5\@@par}%
    \endgroup
  \else
```

```
    \def\@svsechd{#4{\hskip #1\relax #5}}%
  \fi
  \@xsect{#3}}
```

\if@afterindent
\@afterindenttrue

```
\newif\if@afterindent \@afterindenttrue
```

\@afterheading This hook is used in setting up custom-built headings in classes.dtx.

```
\def\@afterheading{%
  \@nobreaktrue
  \everypar{%
    \if@nobreak
      \@nobreakfalse
      \clubpenalty \@M
      \if@afterindent \else
        {\setbox\z@\lastbox}%
      \fi
    \else
      \clubpenalty \@clubpenalty
      \everypar{}%
    \fi}}
```

\@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.

```
\def\@hangfrom#1{\setbox\@tempboxa\hbox{{#1}}%
      \hangindent \wd\@tempboxa\noindent\box\@tempboxa}
```

\c@secnumdepth
\c@tocdepth

```
\newcount\c@secnumdepth
\newcount\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*{`...`}`

```
\def\secdef#1#2{\@ifstar{#2}{\@dblarg{#1}}}
```

70.2.1 Initializations

\sectionmark
\subsectionmark
\subsubsectionmark
\paragraphmark
\subparagraphmark

```
\let\sectionmark\@gobble
\let\subsectionmark\@gobble
\let\subsubsectionmark\@gobble
\let\paragraphmark\@gobble
\let\subparagraphmark\@gobble
```

```
\message{contents,}
```

70.3 Table of Contents etc.

70.3.1 Convention

`\tf@`⟨*foo*⟩ = file number for output for table foo. The file is opened only if `@filesw` = `true`.

70.3.2 Commands

A `\l@`⟨*type*⟩`{`⟨*entry*⟩`}{`⟨*page*⟩`}` Macro needs to 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 LaTeX 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 LaTeX 2.09 comments.

```
\def\@starttoc#1{%
  \begingroup
    \makeatletter
    \@input{\jobname.#1}%
    \if@filesw
      \expandafter\newwrite\csname tf@#1\endcsname
      \immediate\openout \csname tf@#1\endcsname \jobname.#1\relax
    \fi
    \@nobreakfalse
  \endgroup}
```

\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.

```
⟨/2ekernel⟩
⟨*2ekernel | latexrelease⟩
⟨latexrelease⟩\IncludeInRelease{2018/12/01}%
⟨latexrelease⟩                 {\addcontentsline}{Mask line endings}%
\def\addcontentsline#1#2#3{%
  \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.

```
                        \protected@file@percent}}
⟨/2ekernel | latexrelease⟩
⟨latexrelease⟩\EndIncludeInRelease
⟨latexrelease⟩\IncludeInRelease{0000/00/00}%
⟨latexrelease⟩                 {\addcontentsline}{Mask line endings}%
⟨latexrelease⟩\def\addcontentsline#1#2#3{%
⟨latexrelease⟩  \addtocontents{#1}{\protect\contentsline{#2}{#3}{\thepage}}}
⟨latexrelease⟩\EndIncludeInRelease
⟨*2ekernel⟩
```

\addtocontents The \addtocontents{⟨*table*⟩}{⟨*text*⟩} command adds ⟨*text*⟩ to the .⟨*table*⟩ file, with no page number.

```
\long\def\addtocontents#1#2{%
  \protected@write\@auxout
      {\let\label\@gobble \let\index\@gobble \let\glossary\@gobble}%
      {\string\@writefile{#1}{#2}}}
```

\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.

```
\def\contentsline#1{\csname l@#1\endcsname}
```

\@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`

```
⟨/2ekernel⟩
⟨*2ekernel | latexrelease⟩
⟨latexrelease⟩\IncludeInRelease{2018/12/01}%
⟨latexrelease⟩                {\@dottedtocline}{Prevent protrusion}%
\def\@dottedtocline#1#2#3#4#5{%
  \ifnum #1>\c@tocdepth \else
    \vskip \z@ \@plus.2\p@
    {\leftskip #2\relax \rightskip \@tocrmarg \parfillskip -\rightskip
     \parindent #2\relax\@afterindenttrue
     \interlinepenalty\@M
     \leavevmode
     \@tempdima #3\relax
     \advance\leftskip \@tempdima \null\nobreak\hskip -\leftskip
     {#4}\nobreak
     \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.

```
        \mkern \@dotsep mu\hbox{.}\mkern \@dotsep
        mu$}\hfill
     \nobreak
     \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.

```
                        \kern-\p@\kern\p@}%
     \par}%
  \fi}
```

`\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.

```
\DeclareRobustCommand\noprotrusion{\leavevmode\kern-\p@\kern\p@}
```

```
⟨/2ekernel | latexrelease⟩
⟨latexrelease⟩\EndIncludeInRelease
⟨latexrelease⟩\IncludeInRelease{0000/00/00}%
⟨latexrelease⟩                 {\@dottedtocline}{Prevent protrusion}%
⟨latexrelease⟩\def\@dottedtocline#1#2#3#4#5{%
⟨latexrelease⟩  \ifnum #1>\c@tocdepth \else
⟨latexrelease⟩    \vskip \z@ \@plus.2\p@
⟨latexrelease⟩    {\leftskip #2\relax \rightskip \@tocrmarg \parfillskip -\rightskip
⟨latexrelease⟩     \parindent #2\relax\@afterindenttrue
⟨latexrelease⟩     \interlinepenalty\@M
⟨latexrelease⟩     \leavevmode
⟨latexrelease⟩     \@tempdima #3\relax
⟨latexrelease⟩     \advance\leftskip \@tempdima \null\nobreak\hskip -\leftskip
⟨latexrelease⟩     {#4}\nobreak
⟨latexrelease⟩     \leaders\hbox{$\m@th
⟨latexrelease⟩        \mkern \@dotsep mu\hbox{.}\mkern \@dotsep
⟨latexrelease⟩        mu$}\hfill
⟨latexrelease⟩     \nobreak
⟨latexrelease⟩     \hb@xt@\@pnumwidth{\hfil\normalfont \normalcolor #5}%
⟨latexrelease⟩     \par}%
⟨latexrelease⟩  \fi}
⟨latexrelease⟩
⟨latexrelease⟩\let\noprotrusion\@undefined
⟨latexrelease⟩\EndIncludeInRelease
⟨*2ekernel⟩
```

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*⟩ flushleft in a box of width `\@tempdima` (Before 25 Jan 88 change, it also added `\@tempdima` to the hanging indentation.)

```
\def\numberline#1{\hb@xt@\@tempdima{#1\hfil}}
⟨/2ekernel⟩
```

File I
ltfloat.dtx

71 Floats

The different types of floats are identified by a ⟨*type*⟩ 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 ⟨*type*⟩ has associated a positive ⟨*type number*⟩, which is a power of two. E.g.,
figures might be have type number 1, tables type number 2, programs type number 4, etc.

The locations where a float can go are specified by a ⟨*placement specifier*⟩, 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.

71.1 Floating Environments

```
⟨*2ekernel⟩
\message{floats,}
```

Historical LaTeX 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.

```
\c@topnumber       : Number of floats allowed at the top of a column.
\topfraction       : Fraction of column that can be devoted to floats.
\c@dbltopnumber, \dbltopfraction
                   : Same as above, but for double-column floats.
\c@bottomnumber, \bottomfraction
                   : Same as above for bottom of page.
\c@totalnumber     : Number of floats allowed in a single column,
                          including in-text floats.
\textfraction      :Minimum fraction of column that must contain text.
\floatpagefraction: Minimum fraction of page that must be taken
                          up by float page.
\dblfloatpagefraction
                   : 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{\@captype}
      \@dblarg{\@caption{\@captype}}
     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 LaTeX 2.09 comments.

\@floatpenalty

```
\newcount\@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.

```
\def\caption{%
  \ifx\@captype\@undefined
    \@latex@error{\noexpand\caption outside float}\@ehd
    \expandafter\@gobble
  \else
    \refstepcounter\@captype
    \expandafter\@firstofone
  \fi
  {\@dblarg{\@caption\@captype}}%
}
```

\@caption

```
\long\def\@caption#1[#2]#3{%
  \par
  \addcontentsline{\csname ext@#1\endcsname}{#1}%
    {\protect\numberline{\csname the#1\endcsname}{\ignorespaces #2}}%
  \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 LaTeX constructs test for this flag and do not add vertical space when it is true. The intention is that this emulates TeX'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.

```
    \@parboxrestore
    \if@minipage
      \@setminipage
    \fi
    \normalsize
    \@makecaption{\csname fnum@#1\endcsname}{\ignorespaces #3}\par
  \endgroup}
```

\@float

\@dblflset

```
\def\@float#1{%
  \@ifnextchar[%
    {\@xfloat{#1}}%
    {\edef\reserved@a{\noexpand\@xfloat{#1}[\csname fps@#1\endcsname]}%
     \reserved@a}}
```

\@dblfloat

```
\def\@dblfloat{%
  \if@twocolumn\let\reserved@a\@dbflt\else\let\reserved@a\@float\fi
  \reserved@a}
```

\fps@dbl Note that all double floats have default fps 'tp'.

\@setfps This sets the fps, dealing with error conditions by adding the default.

\@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.

```
⟨/2ekernel⟩
⟨latexrelease⟩\IncludeInRelease{2015/01/01}%
⟨latexrelease⟩                   {\@xfloat}{Check float options}%
⟨*2ekernel | latexrelease⟩
\def\@xfloat #1[#2]{%
  \@nodocument
  \def \@captype {#1}%
   \def \@fps {#2}%
   \@onelevel@sanitize \@fps
   \def \reserved@b {!}%
   \ifx \reserved@b \@fps
     \@fpsadddefault
   \else
     \ifx \@fps \@empty
       \@fpsadddefault
     \fi
   \fi
   \ifhmode
     \@bsphack
     \@floatpenalty -\@Mii
   \else
     \@floatpenalty-\@Miii
   \fi
  \ifinner
     \@parmoderr\@floatpenalty\z@
  \else
    \@next\@currbox\@freelist
      {%
       \@tempcnta \sixt@@n
       \expandafter \@tfor \expandafter \reserved@a
         \expandafter :\expandafter =\@fps
         \do
```

Start of changes, use a nested if structure, ending in an error.

```
          {%
           \if \reserved@a h%
             \ifodd \@tempcnta
             \else
               \advance \@tempcnta \@ne
             \fi
           \else\if \reserved@a t%
```

```
      \@setfpsbit \tw@
     \else\if \reserved@a b%
      \@setfpsbit 4%
     \else\if \reserved@a p%
      \@setfpsbit 8%
     \else\if \reserved@a !%
      \ifnum \@tempcnta>15
       \advance\@tempcnta -\sixt@@n\relax
      \fi
     \else
      \@latex@error{Unknown float option `\reserved@a'}%
      {Option `\reserved@a' ignored and `p' used.}%
      \@setfpsbit 8%
     \fi\fi\fi\fi\fi
     }%
```

End of changes

```
   \@tempcntb \csname ftype@\@captype \endcsname
   \multiply \@tempcntb \@xxxii
   \advance \@tempcnta \@tempcntb
   \global \count\@currbox \@tempcnta
   }%
  \@fltovf
 \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.

```
 \global \setbox\@currbox
  \color@vbox
   \normalcolor
   \vbox \bgroup
    \hsize\columnwidth
    \@parboxrestore
    \@floatboxreset
}%
⟨/2ekernel | latexrelease⟩
⟨latexrelease⟩\EndIncludeInRelease
⟨latexrelease⟩\IncludeInRelease{0000/00/00}%
⟨latexrelease⟩         {\@xfloat}{Check float options}%
⟨latexrelease⟩\def\@xfloat #1[#2]{%
⟨latexrelease⟩  \@nodocument
⟨latexrelease⟩  \def \@captype {#1}%
⟨latexrelease⟩   \def \@fps {#2}%
⟨latexrelease⟩   \@onelevel@sanitize \@fps
⟨latexrelease⟩   \def \reserved@b {!}%
⟨latexrelease⟩   \ifx \reserved@b \@fps
⟨latexrelease⟩     \@fpsadddefault
⟨latexrelease⟩   \else
⟨latexrelease⟩     \ifx \@fps \@empty
⟨latexrelease⟩       \@fpsadddefault
⟨latexrelease⟩     \fi
⟨latexrelease⟩   \fi
```

```
⟨latexrelease⟩   \ifhmode
⟨latexrelease⟩     \@bsphack
⟨latexrelease⟩     \@floatpenalty -\@Mii
⟨latexrelease⟩   \else
⟨latexrelease⟩     \@floatpenalty-\@Miii
⟨latexrelease⟩   \fi
⟨latexrelease⟩  \ifinner
⟨latexrelease⟩     \@parmoderr\@floatpenalty\z@
⟨latexrelease⟩  \else
⟨latexrelease⟩    \@next\@currbox\@freelist
⟨latexrelease⟩      {%
⟨latexrelease⟩       \@tempcnta \sixt@@n
⟨latexrelease⟩       \expandafter \@tfor \expandafter \reserved@a
⟨latexrelease⟩         \expandafter :\expandafter =\@fps
⟨latexrelease⟩         \do
⟨latexrelease⟩          {%
⟨latexrelease⟩           \if \reserved@a h%
⟨latexrelease⟩             \ifodd \@tempcnta
⟨latexrelease⟩             \else
⟨latexrelease⟩               \advance \@tempcnta \@ne
⟨latexrelease⟩             \fi
⟨latexrelease⟩           \fi
⟨latexrelease⟩           \if \reserved@a t%
⟨latexrelease⟩             \@setfpsbit \tw@
⟨latexrelease⟩           \fi
⟨latexrelease⟩           \if \reserved@a b%
⟨latexrelease⟩             \@setfpsbit 4%
⟨latexrelease⟩           \fi
⟨latexrelease⟩           \if \reserved@a p%
⟨latexrelease⟩             \@setfpsbit 8%
⟨latexrelease⟩           \fi
⟨latexrelease⟩           \if \reserved@a !%
⟨latexrelease⟩             \ifnum \@tempcnta>15
⟨latexrelease⟩               \advance\@tempcnta -\sixt@@n\relax
⟨latexrelease⟩             \fi
⟨latexrelease⟩           \fi
⟨latexrelease⟩           }%
⟨latexrelease⟩       \@tempcntb \csname ftype@\@captype \endcsname
⟨latexrelease⟩       \multiply \@tempcntb \@xxxii
⟨latexrelease⟩       \advance \@tempcnta \@tempcntb
⟨latexrelease⟩       \global \count\@currbox \@tempcnta
⟨latexrelease⟩       }%
⟨latexrelease⟩    \@fltovf
⟨latexrelease⟩  \fi
⟨latexrelease⟩  \global \setbox\@currbox
⟨latexrelease⟩    \color@vbox
⟨latexrelease⟩      \normalcolor
⟨latexrelease⟩      \vbox \bgroup
⟨latexrelease⟩        \hsize\columnwidth
⟨latexrelease⟩        \@parboxrestore
⟨latexrelease⟩        \@floatboxreset
⟨latexrelease⟩}%
⟨latexrelease⟩\EndIncludeInRelease
⟨*2ekernel⟩
```

`\@floatboxreset` The rational 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.

```
\def \@floatboxreset {%
        \reset@font
        \normalsize
        \@setminipage
}
```

`\@setnobreak`

```
\def \@setnobreak{%
  \if@nobreak
    \let\outer@nobreak\@nobreaktrue
    \@nobreakfalse
  \fi
}
```

`\@setminipage`

```
\def \@setminipage{%
  \@minipagetrue
  \everypar{\@minipagefalse\everypar{}}%
}
```

`\end@float`

```
\def\end@float{%
  \@endfloatbox
  \ifnum\@floatpenalty <\z@
```

We make sure that we never exceed `\textheight`, otherwise float will never get typeset (91/03/15 FMi).

```
    \@largefloatcheck
    \@cons\@currlist\@currbox
    \ifnum\@floatpenalty <-\@Mii
      \penalty -\@Miv
```

Saving and restoring `\prevdepth` added 26 May 87 to prevent extra vertical space when used in vertical mode.

```
      \@tempdima\prevdepth
      \vbox{}%
      \prevdepth\@tempdima
      \penalty\@floatpenalty
    \else
      \vadjust{\penalty -\@Miv \vbox{}\penalty\@floatpenalty}\@Esphack
    \fi
  \fi
}
```

\end@dblfloat

```
⟨/2ekernel⟩
⟨latexrelease⟩\IncludeInRelease{2015/01/01}%
⟨latexrelease⟩                 {\end@dblfloat}{float order in 2-column}%
⟨*2ekernel | latexrelease⟩
\def\end@dblfloat{%
  \if@twocolumn
    \@endfloatbox
    \ifnum\@floatpenalty <\z@
      \@largefloatcheck
```

Force the depth of two column float boxes.

```
      \global\dp\@currbox1sp %
```

What follows is essentially `\end@float` without a starting `\@endfloatbox`.

```
      \@cons\@currlist\@currbox
      \ifnum\@floatpenalty <-\@Mii
        \penalty -\@Miv
        \@tempdima\prevdepth
        \vbox{}%
        \prevdepth\@tempdima
        \penalty\@floatpenalty
      \else
        \vadjust{\penalty -\@Miv \vbox{}\penalty\@floatpenalty}\@Esphack
      \fi

    \fi
  \else
    \end@float
  \fi
}%
⟨/2ekernel | latexrelease⟩
⟨latexrelease⟩\EndIncludeInRelease
⟨latexrelease⟩\IncludeInRelease{0000/00/00}%
⟨latexrelease⟩                 {\end@dblfloat}{float order in 2-column}%
⟨latexrelease⟩\def\end@dblfloat{%
⟨latexrelease⟩\if@twocolumn
⟨latexrelease⟩  \@endfloatbox
⟨latexrelease⟩  \ifnum\@floatpenalty <\z@
```

We make sure that we never exceed `\textheight`, otherwise float will never get typeset (91/03/15 FMi).

```
⟨latexrelease⟩    \@largefloatcheck
⟨latexrelease⟩    \@cons\@dbldeferlist\@currbox
⟨latexrelease⟩  \fi
```

RmS 92/03/18 changed `\@esphack` to `\@Esphack`.

```
⟨latexrelease⟩    \ifnum \@floatpenalty =-\@Mii \@Esphack\fi
⟨latexrelease⟩\else
⟨latexrelease⟩  \end@float
⟨latexrelease⟩\fi
⟨latexrelease⟩}%
⟨latexrelease⟩\EndIncludeInRelease
⟨*2ekernel⟩
```

\@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.

```
\def \@endfloatbox{%
      \par\vskip\z@skip      %% \par\vskip\z@ added 15 Dec 87
      \@minipagefalse
      \outer@nobreak
    \egroup                  %% end of vbox
  \color@endbox
}
```

\outer@nobreak

```
\let\outer@nobreak\@empty
```

\@largefloatcheck This calculates by how much a float is oversize for the page and prints this in a warning message.

```
\def \@largefloatcheck{%
  \ifdim \ht\@currbox>\textheight
    \@tempdima -\textheight
    \advance \@tempdima \ht\@currbox
    \@latex@warning {Float too large for page by \the\@tempdima}%
    \ht\@currbox \textheight
  \fi
}
```

\@dbflt
\@xdblfloat

```
\def\@dbflt#1{\@ifnextchar[{\@xdblfloat{#1}}{\@xdblfloat{#1}[tp]}}
\def\@xdblfloat#1[#2]{%
  \@xfloat{#1}[#2]\hsize\textwidth\linewidth\textwidth}
```

Moved to ltoutput 93/12/16

```
%\newcount\c@topnumber
%\newcount\c@dbltopnumber
%\newcount\c@bottomnumber
%\newcount\c@totalnumber
```

\@floatplacement An analysis of \@floatplacement:

This should be called whenever \@colht has been set.

```
\def\@floatplacement{\global\@topnum\c@topnumber
   % Textpage bit, global:
  \global\@toproom \topfraction\@colht
  \global\@botnum  \c@bottomnumber
  \global\@botroom \bottomfraction\@colht
  \global\@colnum  \c@totalnumber
   % Floatpage bit, local:
  \@fpmin   \floatpagefraction\@colht}
⟨/2ekernel⟩
```

\@dblfloatplacement This should be called only within a group. Now changed to provide extra checks in \@addtodblcol, needed when processing a BANG float.

```
⟨latexrelease⟩\IncludeInRelease{2015/01/01}%
⟨latexrelease⟩            {\@dblfloatplacement}{float order in 2-column}%
⟨*2ekernel | latexrelease⟩
```

When making two column float area, look for floats with 1sp depth.

```
\def\@dblfloatplacement{\global\@dbltopnum\c@dbltopnumber
   \global\@dbltoproom \dbltopfraction\@colht
   \@textmin \@colht
   \advance \@textmin -\@dbltoproom
   \@fpmin \dblfloatpagefraction\textheight
   \@fptop \@dblfptop
   \@fpsep \@dblfpsep
   \@fpbot \@dblfpbot
```

`\f@depth` is used in `\@testwrongwidth` to look for either column or dblcolumn floats. A value of `1sp` signals the latter. Because of this setting here, `\@dblfloatplacment` needs to be called inside a group which is a questionable design.

```
   \def\f@depth{1sp}}%
```

```
⟨/2ekernel | latexrelease⟩
⟨latexrelease⟩\EndIncludeInRelease
⟨latexrelease⟩\IncludeInRelease{0000/00/00}%
⟨latexrelease⟩                 {\@dblfloatplacement}{float order in 2-column}%
⟨latexrelease⟩\def \@dblfloatplacement {%
```

Textpage bit: global, but need not be.

```
⟨latexrelease⟩  \global \@dbltopnum \c@dbltopnumber
⟨latexrelease⟩  \global \@dbltoproom \dbltopfraction\@colht
```

This new bit uses `\@textmin` to locally store the amount of extra room in the column.

```
⟨latexrelease⟩  \@textmin \@colht
⟨latexrelease⟩  \advance \@textmin -\@dbltoproom
```

Floatpage bit: must be local.

```
⟨latexrelease⟩  \@fpmin \dblfloatpagefraction\textheight
⟨latexrelease⟩  \@fptop \@dblfptop
⟨latexrelease⟩  \@fpsep \@dblfpsep
⟨latexrelease⟩  \@fpbot \@dblfpbot
⟨latexrelease⟩}%
⟨latexrelease⟩\EndIncludeInRelease
⟨*2ekernel⟩
```

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., `\@leftmarginskip` ==

```
      \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 LaTeX 2.09 comments.

\marginpar

```
\def\marginpar{%
  \ifhmode
    \@bsphack
    \@floatpenalty -\@Mii
  \else
    \@floatpenalty-\@Miii
  \fi
  \ifinner
    \@parmoderr
    \@floatpenalty\z@
  \else
    \@next\@currbox\@freelist{}{}%
    \@next\@marbox\@freelist{\global\count\@marbox\m@ne}%
       {\@floatpenalty\z@
        \@fltovf\def\@currbox{\@tempboxa}\def\@marbox{\@tempboxa}}%
  \fi
  \@ifnextchar [\@xmpar\@ympar}
```

\@xmpar

```
\long\def\@xmpar[#1]#2{%
  \@savemarbox\@marbox{#1}%
  \@savemarbox\@currbox{#2}%
  \@xympar}
```

\@ympar

```
\long\def\@ympar#1{%
  \@savemarbox\@marbox{#1}%
  \global\setbox\@currbox\copy\@marbox
  \@xympar}
```

\@savemarbox

```
\long\def \@savemarbox #1#2{%
  \global\setbox #1%
    \color@vbox
      \vtop{%
        \hsize\marginparwidth
        \@parboxrestore
        \@marginparreset
        #2%
        \@minipagefalse
        \outer@nobreak
        }%
    \color@endbox
}
```

\@marginparreset The rational for allowing these normally global flags to be set locally here, via `\@parboxrestore` was stated originally by Donald Arsenau 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.

```
\def \@marginparreset {%
        \reset@font
        \normalsize
%        \let\if@nobreak\iffalse
%        \let\if@noskipsec\iffalse
%        \@setnobreak
        \@setminipage
}
```

`\@xympar`

Setting the box here is done only because the code uses `\end@float`; it will be empty and gets discarded.

```
\def \@xympar{%
  \ifnum\@floatpenalty <\z@\@cons\@currlist\@marbox\fi
  \setbox\@tempboxa
    \color@vbox
      \vbox \bgroup
  \end@float
  \@ignorefalse
  \@esphack
}
```

`\reversemarginpar`
`\normalmarginpar`

```
\def\reversemarginpar{\global\@mparbottom\z@ \@reversemargintrue}
\def\normalmarginpar{\global\@mparbottom\z@ \@reversemarginfalse}
```

```
\message{footnotes,}
```

71.2 Footnotes

Historical LaTeX 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 `\textsuperscript{\@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 LaTeX 2.09 comments.

`\footins` LaTeX does use the same insert for footnotes as PLAIN.

```
\newinsert\footins
```

LaTeX leaves these initializations for the `\footins` insert.

```
\skip\footins=\bigskipamount % space added when footnote is present
\count\footins=1000 % footnote magnification factor (1 to 1)
\dimen\footins=8in % maximum footnotes per page
```

`\footnoterule` LaTeX keeps PLAIN TeX's `\footnoterule` as the default.

```
\def\footnoterule{\kern-3\p@
  \hrule \@width 2in \kern 2.6\p@} % the \hrule is .4pt high
```

`\thefootnote`

```
\@definecounter{footnote}
\def\thefootnote{\@arabic\c@footnote}
```

`\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.

```
\@definecounter{mpfootnote}
\def\thempfootnote{{\itshape\@alph\c@mpfootnote}}
```

`\@makefnmark` Default definition.

```
%\def\@makefnmark{\hbox{$^{\@thefnmark}\m@th$}}
\def\@makefnmark{\hbox{\@textsuperscript{\normalfont\@thefnmark}}}
```

`\textsuperscript` This command provides superscript characters in the current text font. It's implementation might change!!!

```
\DeclareRobustCommand*\textsuperscript[1]{%
  \@textsuperscript{\selectfont#1}}
```

`\@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.

```
\def\@textsuperscript#1{%
  {\m@th\ensuremath{^{\mbox{\fontsize\sf@size\z@#1}}}}}
```

`\textsubscript`

```
⟨/2ekernel⟩
⟨latexrelease⟩\IncludeInRelease{2015/01/01}%
⟨latexrelease⟩                   {\textsubscript}{\textsubscript}%
⟨*2ekernel | latexrelease⟩
```

```
\DeclareRobustCommand*\textsubscript[1]{%
  \@textsubscript{\selectfont#1}}%
```

`\@textsubscript`

```
\def\@textsubscript#1{%
  {\m@th\ensuremath{_{\mbox{\fontsize\sf@size\z@#1}}}}}
```

```
⟨/2ekernel | latexrelease⟩
⟨latexrelease⟩\EndIncludeInRelease
⟨latexrelease⟩\IncludeInRelease{0000/00/00}%
⟨latexrelease⟩                   {\textsubscript}{\textsubscript}%
⟨latexrelease⟩\let\textsubscript\@undefined
⟨latexrelease⟩\let\@textsubscript\@undefined
⟨latexrelease⟩\EndIncludeInRelease
⟨*2ekernel⟩
```

`\footnotesep`

```
\newdimen\footnotesep
```

`\footnote`

```
\def\footnote{\@ifnextchar[\@xfootnote{\stepcounter\@mpfn
     \protected@xdef\@thefnmark{\thempfn}%
     \@footnotemark\@footnotetext}}
```

`\@xfootnote`

```
\def\@xfootnote[#1]{%
   \begingroup
     \csname c@\@mpfn\endcsname #1\relax
     \unrestored@protected@xdef\@thefnmark{\thempfn}%
   \endgroup
   \@footnotemark\@footnotetext}
```

`\@footnotetext`

```
\long\def\@footnotetext#1{\insert\footins{%
    \reset@font\footnotesize
    \interlinepenalty\interfootnotelinepenalty
    \splittopskip\footnotesep
    \splitmaxdepth \dp\strutbox \floatingpenalty \@MM
    \hsize\columnwidth \@parboxrestore
    \protected@edef\@currentlabel{%
       \csname p@footnote\endcsname\@thefnmark
    }%
    \color@begingroup
      \@makefntext{%
        \rule\z@\footnotesep\ignorespaces#1\@finalstrut\strutbox}%
    \color@endgroup}}%
```

\footnotemark

```
\def\footnotemark{%
   \@ifnextchar[\@xfootnotemark
     {\stepcounter{footnote}%
      \protected@xdef\@thefnmark{\thefootnote}%
      \@footnotemark}}
```

\@xfootnotemark

```
\def\@xfootnotemark[#1]{%
   \begingroup
      \c@footnote #1\relax
      \unrestored@protected@xdef\@thefnmark{\thefootnote}%
   \endgroup
   \@footnotemark}
```

\@footnotemark

```
\def\@footnotemark{%
  \leavevmode
  \ifhmode\edef\@x@sf{\the\spacefactor}\nobreak\fi
  \@makefnmark
  \ifhmode\spacefactor\@x@sf\fi
  \relax}
```

\footnotetext

```
\def\footnotetext{%
    \@ifnextchar [\@xfootnotenext
       {\protected@xdef\@thefnmark{\thempfn}%
    \@footnotetext}}
```

\@xfootnotenext

```
\def\@xfootnotenext[#1]{%
  \begingroup
     \csname c@\@mpfn\endcsname #1\relax
     \unrestored@protected@xdef\@thefnmark{\thempfn}%
  \endgroup
  \@footnotetext}
```

\thempfn
\@mpfn

```
\def\@mpfn{footnote}
\def\thempfn{\thefootnote}
⟨/2ekernel⟩
```

File J

ltidxglo.dtx

72 Index and Glossary Generation

Index and Glossary commands.

`\makeindex` A preamble command to turn on indexing.

`\makeglossary` A preamble command to turn on making glossary entries.

`\index` Make an index entry for `#1`.

`\glossary` Make a glossary entry for `#1`.

Historical LaTeX 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 LaTeX 2.09 comments.

```
⟨*2ekernel⟩
\message{index,}
```

\makeindex

```
\def\makeindex{%
  \newwrite\@indexfile
  \immediate\openout\@indexfile=\jobname.idx
  \def\index{\@bsphack\begingroup
             \@sanitize
             \@wrindex}\typeout
   {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.

```
  \let\makeindex\@empty
}
\@onlypreamble\makeindex
```

\@wrindex

```
\def\@wrindex#1{%
   \protected@write\@indexfile{}%
      {\string\indexentry{#1}{\thepage}}%
 \endgroup
 \@esphack}
```

\index

```
\def\index{\@bsphack\begingroup \@sanitize\@index}
```

\@index

```
\def\@index#1{\endgroup\@esphack}
```

\makeglossary

```
\def\makeglossary{%
  \newwrite\@glossaryfile
  \immediate\openout\@glossaryfile=\jobname.glo
  \def\glossary{\@bsphack\begingroup
                \@sanitize
                \@wrglossary}\typeout
   {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.

```
  \let\makeglossary\@empty
}
\@onlypreamble\makeglossary
```

\@wrglossary

```
\def\@wrglossary#1{%
   \protected@write\@glossaryfile{}%
      {\string\glossaryentry{#1}{\thepage}}%
 \endgroup
 \@esphack}
```

\glossary

```
\def\glossary{\@bsphack\begingroup\@sanitize\@index}
```

```
⟨/2ekernel⟩
```

File K
ltbibl.dtx

73 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.

`\bibliography` `\bibliography{`⟨*file1,file2, ... ,filen*⟩`}` : specifies the bibdata files. Writes a `\bibdata` entry on the `.aux` file and tries to read in `mainfile.bbl`.

`\bibliographystyle` `\bibliographystyle{`⟨*style*⟩`}` : Writes a `\bibstyle` entry on the `.aux` file.

`thebibliography` The `thebibliography` environment is a list environment. To save the use of an extra counter, it should use `enumiv` as the item counter. Instead of using `\item`, 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.

`\cite` Entries are cited by the command `\cite{`⟨*name*⟩`}`.

`\nocite` `\nocite{`⟨*citations*⟩`}` puts information on the `.aux` file that causes BIBTEX to include the `{`⟨*citations*⟩`}` list in the bibliography, but puts nothing in the text.

`\nocite{*}` is special: it tells BIBTEX to put the whole of a collection of references into the bibiography.

```
⟨*2ekernel⟩
\message{bibliography,}
```

Historical LATEX 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@FOO : The name or number of the reference created by \cite{FOO}
E.g., if \cite{FOO} -> [17] , then \b@FOO -> 17.

End of historical LaTeX 2.09 comments.

\bibitem

```
\def\bibitem{\@ifnextchar[\@lbibitem\@bibitem}
```

\@lbibitem

```
\def\@lbibitem[#1]#2{\item[\@biblabel{#1}\hfill]\if@filesw
      {\let\protect\noexpand
       \immediate
       \write\@auxout{\string\bibcite{#2}{#1}}}\fi\ignorespaces}
```

\@bibitem

```
\def\@bibitem#1{\item\if@filesw \immediate\write\@auxout
       {\string\bibcite{#1}{\the\value{\@listctr}}}\fi\ignorespaces}
```

\bibcite

```
\def\bibcite{\@newl@bel b}
```

\citation

```
\let\citation\@gobble
```

\cite

```
\DeclareRobustCommand\cite{%
  \@ifnextchar [{\@tempswatrue\@citex}{\@tempswafalse\@citex[]}}
```

\@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)

```
\def\@citex[#1]#2{\leavevmode
  \let\@citea\@empty
  \@cite{\@for\@citeb:=#2\do
    {\@citea\def\@citea{,\penalty\@m\ }%
     \edef\@citeb{\expandafter\@firstofone\@citeb\@empty}%
     \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 Arsenau. 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.

```
     \@ifundefined{b@\@citeb}{\hbox{\reset@font\bfseries ?}%
       \G@refundefinedtrue
       \@latex@warning
         {Citation `\@citeb' on page \thepage \space undefined}}%
       {\@cite@ofmt{\csname b@\@citeb\endcsname}}}}{#1}}
```

`\bibdata`
`\bibstyle`

```
\let\bibdata=\@gobble
\let\bibstyle=\@gobble
```

`\bibliography`

```
\def\bibliography#1{%
  \if@filesw
    \immediate\write\@auxout{\string\bibdata{\zap@space#1 \@empty}}%
  \fi
  \@input@{\jobname.bbl}}
```

`\bibliographystyle`

```
\def\bibliographystyle#1{%
  \ifx\@begindocumenthook\@undefined\else
    \expandafter\AtBeginDocument
  \fi
    {\if@filesw
       \immediate\write\@auxout{\string\bibstyle{#1}}%
     \fi}}
```

`\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.

```
\def\nocite#1{\@bsphack
```

With the implementation designed already in LaTeX 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 LaTeX, missing all citations done with `\nocite`. Thus we do only generate an error message and leave the fix for a LaTeX 2_ε successor.

```
  \ifx\@onlypreamble\document
```

Since we are after `\begin{document}` we can do the citations:

```
    \@for\@citeb:=#1\do{%
      \edef\@citeb{\expandafter\@firstofone\@citeb}%
      \if@filesw\immediate\write\@auxout{\string\citation{\@citeb}}\fi
      \@ifundefined{b@\@citeb}{\G@refundefinedtrue
          \@latex@warning{Citation ‘\@citeb’ undefined}}{}}%
  \else
```

But before `\begin{document}` we raise an error message:

```
    \@latex@error{Cannot be used in preamble}\@eha
```

Without the compatibility problems we could fix the problem as follows:

```
    % \AtBeginDocument{\nocite{#1}}
  \fi
  \@esphack}
```

Since `\nocite{*}` should not produce a warning about undefined citation keys (seee 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 BibTeX in the first place).

```
\expandafter\let\csname b@*\endcsname\@empty
```

73.1 Default definitions

This hook determines the ‘relative formatting’ of the two logical parts of a citation with comment.

`\@cite`

```
\def\@cite#1#2{[{#1\if@tempswa , #2\fi}]}
```

`\@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.

```
\let\@cite@ofmt\hbox
```

`\@biblabel`

```
\def\@biblabel#1{[#1]}
⟨/2ekernel⟩
```

File L
ltpage.dtx

74 Page styles and related commands

74.1 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.

74.2 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.)

74.3 marking conventions

LaTeX extends TeX'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 TeX's `\botmark.`
`\rightmark` : Used in the output routine, gets the current 'right' mark. Works like TeX'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.

```
⟨*2ekernel⟩
```

`\pagestyle` User command to set the page style for this and following pages.

```
\def\pagestyle#1{%
  \@ifundefined{ps@#1}%
    \undefinedpagestyle
    {\@nameuse{ps@#1}}}
```

\thispagestyle User command to set the page style for this page only.

```
\def\thispagestyle#1{%
  \@ifundefined{ps@#1}%
    \undefinedpagestyle
    {\global\@specialpagetrue\gdef\@specialstyle{#1}}}
```

\ps@empty The empty page style: No head or foot line.

```
\def\ps@empty{%
  \let\@mkboth\@gobbletwo\let\@oddhead\@empty\let\@oddfoot\@empty
  \let\@evenhead\@empty\let\@evenfoot\@empty}
```

\ps@plain The plain page style: No head, centred page number in foot.

```
\def\ps@plain{\let\@mkboth\@gobbletwo
     \let\@oddhead\@empty\def\@oddfoot{\reset@font\hfil\thepage
     \hfil}\let\@evenhead\@empty\let\@evenfoot\@oddfoot}
```

\@leftmark
\@rightmark We implement `\@leftmark` and `\@rightmark` in terms of already defined commands to save token space. We can't get rid of them since they are sometimes used in applications.

```
\let\@leftmark\@firstoftwo
\let\@rightmark\@secondoftwo
```

```
⟨/2ekernel⟩
⟨*2ekernel | latexrelease⟩
⟨latexrelease⟩\IncludeInRelease{2019/10/01}%
⟨latexrelease⟩                  {\markboth}{Make commands robust}%
```

\markboth
\markright User commands for setting LaTeX marks.

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`

```
\DeclareRobustCommand\markboth[2]{%
  \begingroup
    \let\label\relax \let\index\relax \let\glossary\relax
    \unrestored@protected@xdef\@themark {{#1}{#2}}%
    \@temptokena \expandafter{\@themark}%
    \mark{\the\@temptokena}%
  \endgroup
  \if@nobreak\ifvmode\nobreak\fi\fi}
```

```
\DeclareRobustCommand\markright[1]{%
  \begingroup
    \let\label\relax \let\index\relax \let\glossary\relax
```

Protection is handled inside `\@markright`.

```
    \expandafter\@markright\@themark {#1}%
    \@temptokena \expandafter{\@themark}%
    \mark{\the\@temptokena}%
  \endgroup
  \if@nobreak\ifvmode\nobreak\fi\fi}
```

```
⟨/2ekernel | latexrelease⟩
⟨latexrelease⟩\EndIncludeInRelease
⟨latexrelease⟩\IncludeInRelease{0000/00/00}%
```

```
⟨latexrelease⟩                  {\markboth}{Make commands robust}%
⟨latexrelease⟩
⟨latexrelease⟩\kernel@make@fragile\markboth
⟨latexrelease⟩\kernel@make@fragile\markright
⟨latexrelease⟩
⟨latexrelease⟩\EndIncludeInRelease
⟨*2ekernel⟩
```

`\@markright`
`\leftmark`
`\rightmark`

```
\def\@markright#1#2#3{\@temptokena {#1}%
  \unrestored@protected@xdef\@themark{{\the\@temptokena}{#3}}}
\def\leftmark{\expandafter\@leftmark\botmark\@empty\@empty}
\def\rightmark{\expandafter\@rightmark\firstmark\@empty\@empty}
```

`\@themark` Initialise LaTeX's marks without setting a TeX mark ⟨*whatsit*⟩.

```
\def\@themark{{}{}}
```

`\mark` Test versions of LaTeX 2$_\varepsilon$ initialised TeX's `\mark` system at this point, but this was removed before the first release.

```
\AtBeginDocument{\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`.)

```
\DeclareRobustCommand\raggedbottom{%
  \def\@textbottom{\vskip \z@ \@plus.0001fil}\let\@texttop\relax}
```

`\flushbottom` `\flushbottom`: Inverse of `\raggedbottom` — makes all pages the same height.

```
\DeclareRobustCommand\flushbottom{%
  \let\@textbottom\relax \let\@texttop\relax}
```

`\sloppy` `\sloppy` will never (well, hardly ever) produce overfull boxes, but may produce underfull ones. (14 June 85)

```
\DeclareRobustCommand\sloppy{%
  \tolerance 9999%
  \emergencystretch 3em%
  \hfuzz .5\p@
  \vfuzz\hfuzz}
```

sloppypar A sloppypar environment is equivalent to `{\par \sloppy ... \par}`.

```
\def\sloppypar{\par\sloppy}
\def\endsloppypar{\par}
```

`\fussy` Resets TeX's parameters to their normal finicky values.

```
\DeclareRobustCommand\fussy{%
  \emergencystretch\z@
  \tolerance 200%
  \hfuzz .1\p@
  \vfuzz\hfuzz}
```

`\overfullrule` LaTeX default is no overfull box rule. Changed by document class option.

```
\overfullrule 0pt
```

```
⟨/2ekernel⟩
```

File M
ltoutput.dtx

75 Output Routine

75.1 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.

```
⟨defx⟩\begingroup
⟨defx⟩\makeatletter
⟨defx⟩\nfss@catcodes
⟨2ekernel⟩\expandafter\let\csname ver@autoout1.sty\endcsname\fmtversion
```

Historical LaTeX 2.09 comments (not necessarily accurate any more):

```
⟨*2ekernel⟩
\message{output,}
```

```
    ****************************************
    *                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

\@dblfptop, \@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	`\@dblfptop`
`\@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	\pagebreak \newpage
-10001	\clearpage (\penalty -10000 \vbox{} \penalty -10001)
-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
                              \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 LaTeX 2.09 comments.

```
\maxdeadcycles = 100
\let\@elt\relax
\def\@next#1#2#3#4{\ifx#2\@empty #4\else
  \expandafter\@xnext #2\@@#1#2#3\fi}
\def\@xnext \@elt #1#2\@@#3#4{\def#3{#1}\gdef#4{#2}}

\def\@testfalse{\global\let\if@test\iffalse}
\def\@testtrue {\global\let\if@test\iftrue}
\@testfalse

\def\@bitor#1#2{\@testfalse {\let\@elt\@xbitor
  \@tempcnta #1\relax #2}}
```

RmS 91/11/22: Added test for `\count#1 = 0`. Suggested by Chris Rowley.

```
\def\@xbitor #1{\@tempcntb \count#1
   \ifnum \@tempcnta =\z@
   \else
     \divide\@tempcntb\@tempcnta
     \ifodd\@tempcntb \@testtrue\fi
   \fi}
```

DEFINITION OF FLOAT BOXES:

```
⟨/2ekernel⟩
⟨latexrelease⟩\IncludeInRelease{2015/10/01}%
⟨latexrelease⟩                 {\bx@ZZ}{Extended float list}%
⟨*2ekernel | latexrelease⟩
\let\@elt\newinsert
⟨*2ekernel⟩
\def\@freelist{%
  \@elt\bx@A\@elt\bx@B\@elt\bx@C\@elt\bx@D\@elt\bx@E
  \@elt\bx@F\@elt\bx@G\@elt\bx@H\@elt\bx@I\@elt\bx@J
  \@elt\bx@K\@elt\bx@L\@elt\bx@M\@elt\bx@N
  \@elt\bx@O\@elt\bx@P\@elt\bx@Q\@elt\bx@R}
\@freelist
⟨/2ekernel⟩
\ifx\numexpr\@undefined\else
\def\reserved@a{%
  \@elt\bx@S\@elt\bx@T\@elt\bx@U\@elt\bx@V
  \@elt\bx@W\@elt\bx@X\@elt\bx@Y\@elt\bx@Z
  \@elt\bx@AA\@elt\bx@BB\@elt\bx@CC\@elt\bx@DD\@elt\bx@EE
  \@elt\bx@FF\@elt\bx@GG\@elt\bx@HH\@elt\bx@II\@elt\bx@JJ
  \@elt\bx@KK\@elt\bx@LL\@elt\bx@MM\@elt\bx@NN
  \@elt\bx@OO\@elt\bx@PP\@elt\bx@QQ\@elt\bx@RR
  \@elt\bx@SS\@elt\bx@TT\@elt\bx@UU\@elt\bx@VV
  \@elt\bx@WW\@elt\bx@XX\@elt\bx@YY\@elt\bx@ZZ}
\reserved@a
\def\@elt{\noexpand\@elt\noexpand}
\edef\@freelist{\@freelist\reserved@a}
\fi
\let\reserved@a\relax
\let\@elt\relax
⟨/2ekernel | latexrelease⟩
⟨latexrelease⟩\EndIncludeInRelease
⟨latexrelease⟩\IncludeInRelease{0000/00/00}%
⟨latexrelease⟩                 {\bx@ZZ}{Extended float list}%
⟨latexrelease⟩\def\@freelist{%
⟨latexrelease⟩  \@elt\bx@A\@elt\bx@B\@elt\bx@C\@elt\bx@D\@elt\bx@E
⟨latexrelease⟩  \@elt\bx@F\@elt\bx@G\@elt\bx@H\@elt\bx@I\@elt\bx@J
⟨latexrelease⟩  \@elt\bx@K\@elt\bx@L\@elt\bx@M\@elt\bx@N
⟨latexrelease⟩  \@elt\bx@O\@elt\bx@P\@elt\bx@Q\@elt\bx@R}
⟨latexrelease⟩  \insc@unt=234
⟨latexrelease⟩\EndIncludeInRelease
⟨*2ekernel⟩
```

```
\gdef\@toplist{}
\gdef\@botlist{}
\gdef\@midlist{}
\gdef\@currlist{}
```

```
\gdef\@deferlist{}
\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.

```
\gdef\@dbldeferlist{}
```

PAGE LAYOUT PARAMETERS

```
\newdimen\topmargin
\newdimen\oddsidemargin
\newdimen\evensidemargin
\let\@themargin=\oddsidemargin
\newdimen\headheight
\newdimen\headsep
\newdimen\footskip
\newdimen\textheight
\newdimen\textwidth
\newdimen\columnwidth
\newdimen\columnsep
\newdimen\columnseprule
\newdimen\marginparwidth
\newdimen\marginparsep
\newdimen\marginparpush
```

`\AtBeginDvi`
`\@begindvibox`

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.

```
\newbox\@begindvibox
\DeclareRobustCommand \AtBeginDvi [1]{%
  \global \setbox \@begindvibox
    \vbox{\unvbox \@begindvibox #1}%
}
```

`\@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?

```
\newdimen\@maxdepth
\@maxdepth = \maxdepth
```

`\paperheight`
`\paperwidth`

New `\paper`. . . registers.

```
\newdimen\paperheight
\newdimen\paperwidth
```

`\if@insert`
`\if@fcolmade`
`\if@specialpage`
`\if@firstcolumn`
`\if@twocolumn`
`\if@twoside`
`\if@reversemarginpar`
`\if@mparswitch`
`\col@number`

Local switches first:

```
\newif \if@insert
```

These should definitely be global:

```
\newif \if@fcolmade
\newif \if@specialpage \@specialpagefalse
```

These should be global but are not always set globally in other files.

```
\newif \if@firstcolumn \@firstcolumntrue
\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?

```
\newif \if@twoside     \@twosidefalse
\newif \if@reversemargin \@reversemarginfalse
\newif \if@mparswitch  \@mparswitchfalse
```

This counter has been imported from 'multicol'.

```
\newcount \col@number
\col@number \@ne
```

Historical LaTeX 2.09 comments (not necessarily accurate any more):
INTERNAL REGISTERS

```
\newcount\@topnum
\newdimen\@toproom
\newcount\@dbltopnum
\newdimen\@dbltoproom
\newcount\@botnum
\newdimen\@botroom
\newcount\@colnum
\newdimen\@textmin
\newdimen\@fpmin
\newdimen\@colht
\newdimen\@colroom
\newdimen\@pageht
\newdimen\@pagedp
\newdimen\@mparbottom \@mparbottom\z@
\newcount\@currtype
\newbox\@outputbox
\newbox\@leftcolumn
\newbox\@holdpg
```

```
\def\@thehead{\@oddhead} % initialization
\def\@thefoot{\@oddfoot}
```

End of historical LaTeX 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.

```
\def\clearpage{%
  \ifvmode
    \ifnum \@dbltopnum =\m@ne
      \ifdim \pagetotal <\topskip
        \hbox{}%
      \fi
    \fi
  \fi
  \newpage
  \write\m@ne{}%
  \vbox{}%
  \penalty -\@Mi
}
```

\cleardoublepage

```
\def\cleardoublepage{\clearpage\if@twoside \ifodd\c@page\else
    \hbox{}\newpage\if@twocolumn\hbox{}\newpage\fi\fi\fi}
⟨/2ekernel⟩
```

\onecolumn

```
⟨*2ekernel | fltrace⟩
\def\onecolumn{%
  \clearpage
  \global\columnwidth\textwidth
  \global\hsize\columnwidth
  \global\linewidth\columnwidth
  \global\@twocolumnfalse
  \col@number \@ne
  \@floatplacement}
```

\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.

```
⟨/2ekernel | fltrace⟩
⟨latexrelease⟩\IncludeInRelease{2017/04/15}%
⟨latexrelease⟩                  {\newpage}{Check depth of page}%
⟨*2ekernel | latexrelease | fltrace⟩
\def \newpage {%
  \if@noskipsec
    \ifx \@nodocument\relax
      \leavevmode
      \global \@noskipsecfalse
    \fi
  \fi
  \if@inlabel
    \leavevmode
    \global \@inlabelfalse
  \fi
  \if@nobreak \@nobreakfalse \everypar{}\fi
  \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 TEX. So we back up by that amount (or by \maxdepth if it is really huge), to mimic the normal behavior without the \newpage.

```
  \ifdim\prevdepth>\z@
    \vskip -%
      \ifdim\prevdepth>\maxdepth
        \maxdepth
      \else
        \prevdepth
      \fi
  \fi
  \vfil
  \penalty -\@M}
⟨/2ekernel | latexrelease | fltrace⟩
⟨latexrelease⟩\EndIncludeInRelease
⟨latexrelease⟩\IncludeInRelease{0000/00/00}%
⟨latexrelease⟩                 {\newpage}{Check depth of page}%
⟨latexrelease⟩\def \newpage {%
⟨latexrelease⟩  \if@noskipsec
⟨latexrelease⟩    \ifx \@nodocument\relax
⟨latexrelease⟩      \leavevmode
⟨latexrelease⟩      \global \@noskipsecfalse
⟨latexrelease⟩    \fi
⟨latexrelease⟩  \fi
⟨latexrelease⟩  \if@inlabel
⟨latexrelease⟩    \leavevmode
⟨latexrelease⟩    \global \@inlabelfalse
⟨latexrelease⟩  \fi
⟨latexrelease⟩  \if@nobreak \@nobreakfalse \everypar{}\fi
⟨latexrelease⟩  \par
⟨latexrelease⟩  \vfil
⟨latexrelease⟩  \penalty -\@M}
⟨latexrelease⟩\EndIncludeInRelease
⟨*2ekernel | fltrace⟩
```

`\@emptycol` It may be better to use an invisible rule rather than an empty box here.

```
\def \@emptycol {\vbox{}\penalty -\@M}
```

`\twocolumn` `\@topnewpage` There are several bug fixes to the two-column stuff here.

```
\def \twocolumn {%
  \clearpage
  \global\columnwidth\textwidth
  \global\advance\columnwidth-\columnsep
  \global\divide\columnwidth\tw@
  \global\hsize\columnwidth
  \global\linewidth\columnwidth
  \global\@twocolumntrue
  \global\@firstcolumntrue
  \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.

```
  \@ifnextchar [\@topnewpage\@floatplacement
}
```

Note that here, getting a box from the freelist can assume success since this comes just after a `\clearpage`.

```
\long\def \@topnewpage [#1]{%
  \@nodocument
  \@next\@currbox\@freelist{}{}%
  \global \setbox\@currbox
    \color@vbox
      \normalcolor
      \vbox {%
        \hsize\textwidth
        \@parboxrestore
        \col@number \@ne
        #1%
        \vskip -\dbltextfloatsep
             }%
    \color@endbox
```

Added size test and warning message; perhaps we should use an error message.

```
  \ifdim \ht\@currbox>\textheight
    \ht\@currbox \textheight
  \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.

```
  \global \count\@currbox \tw@
  \@tempdima -\ht\@currbox
  \advance \@tempdima -\dbltextfloatsep
  \global \advance \@colht \@tempdima
  \ifx \@dbltoplist \@empty
  \else
    \@latexerr{Float(s) lost}\@ehb
    \let \@dbltoplist \@empty
  \fi
  \@cons \@dbltoplist \@currbox
```

This setting of `\@dbltopnum` is used only to change the typesetting in `\@combinedblfloats`.

```
  \global \@dbltopnum \m@ne
⟨*trace⟩
    \fl@trace{dbltopnum set to -1 (= \the \@dbltopnum) (topnewpage)}%
⟨/trace⟩
```

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.

```
  \ifdim \@colht<2.5\baselineskip
    \@latex@warning@no@line {Optional argument of \noexpand\twocolumn
                too tall on page \thepage}%
    \@emptycol
    \if@firstcolumn
    \else
      \@emptycol
    \fi
  \else
    \global \vsize \@colht
    \global \@colroom \@colht
    \@floatplacement
  \fi
}
```

\output
\@specialoutput

This needs some small adjustments. We cannot guarantee that the float mechanism will 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.

```
\output {%
  \let \par \@@par
  \ifnum \outputpenalty<-\@M
    \@specialoutput
  \else
    \@makecol
    \@opcol
```

Moved to `\@opcol`: `\@floatplacement`.

```
    \@startcolumn
```

This loop could be replaced by an `\expandafter` tail recursion in `\@startcolumn`.

```
    \@whilesw \if@fcolmade \fi
      {%
⟨*trace⟩
       \fl@trace{PAGE: float \if@twocolumn column \else page \fi
                    completed}%
⟨/trace⟩
       \@opcol\@startcolumn}%
  \fi
  \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`.

```
    \ifdim \@colroom<1.5\baselineskip
      \ifdim \@colroom<\textheight
        \@latex@warning@no@line {Text page \thepage\space
                               contains only floats}%
        \@emptycol
%         \if@twocolumn
%           \if@firstcolumn
%           \else
%             \@emptycol
%           \fi
%         \fi
      \else
        \global \vsize \@colroom
      \fi
    \else
      \global \vsize \@colroom
    \fi
  \else
    \global \vsize \maxdimen
  \fi
}
```

Historical LaTeX 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 LaTeX 2.09 comments.

```
\gdef\@specialoutput{%
   \ifnum \outputpenalty>-\@Mii
     \@doclearpage
   \else
     \ifnum \outputpenalty<-\@Miii
       \ifnum \outputpenalty<-\@MM \deadcycles \z@ \fi
       \global \setbox\@holdpg \vbox {\unvbox\@cclv}%
     \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.

```
       \global \setbox\@holdpg \vbox{%
                      \unvbox\@holdpg
                      \unvbox\@cclv
```

We must now remove the box added by the float mechanism and the `\topskip` glue therefore added above it by TeX.

```
                      \setbox\@tempboxa \lastbox
                      \unskip
                                     }%
```

These two are needed as separate dimensions only by `\@addmarginpar`; for other purposes we put the whole size into `\@pageht` (see below).

```
       \@pagedp \dp\@holdpg
       \@pageht \ht\@holdpg
       \unvbox \@holdpg
       \@next\@currbox\@currlist{%
         \ifnum \count\@currbox>\z@
```

Putting the whole size into `\@pageht` (see above).

```
           \advance \@pageht \@pagedp
           \ifvoid\footins \else
             \advance \@pageht \ht\footins
             \advance \@pageht \skip\footins
             \advance \@pageht \dp\footins
           \fi
           \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?

```
             \ifdim \wd\@kludgeins=\z@
               \advance \@pageht \ht\@kludgeins
⟨*trace⟩
               \fl@trace {Extra size added: \the \ht\@kludgeins}%
⟨/trace⟩
             \fi
           \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.

```
           \@reinserts
           \@addtocurcol
         \else
           \@reinserts
           \@addmarginpar
         \fi
         }\@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.

```
       \ifnum \outputpenalty<\z@
         \if@nobreak
           \nobreak
         \else
           \addpenalty \interlinepenalty
         \fi
       \fi
     \fi
   \fi
}
⟨/2ekernel | fltrace⟩
```

`\@testwrongwidth`
`\f@depth`

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).

```
⟨latexrelease⟩\IncludeInRelease{2015/01/01}%
⟨latexrelease⟩                 {\@testwrongwidth}{float order in 2-column}%
⟨*2ekernel | latexrelease | fltrace⟩

\def\@testwrongwidth #1{%
  \ifdim\dp#1=\f@depth
⟨*trace⟩
    \fl@trace{\string#1
              \ifdim\f@depth=\z@ single \else double \fi
              column float -- ok}%
⟨/trace⟩
  \else
    \global\@testtrue
⟨*trace⟩
    \fl@trace{\string#1
              \ifdim\f@depth=\z@ double \else single \fi
              column float -- wrong}%
⟨/trace⟩
  \fi}%
```

Normally looking for single column floats, which have zero depth.

```
\let\f@depth\z@

⟨/2ekernel | latexrelease | fltrace⟩
⟨latexrelease⟩\EndIncludeInRelease
⟨latexrelease⟩\IncludeInRelease{0000/00/00}%
⟨latexrelease⟩                 {\@testwrongwidth}{float order in 2-column}%
⟨latexrelease⟩\let\@testwrongwidth\@undefined
⟨latexrelease⟩\let\f@depth\@undefined
⟨latexrelease⟩\EndIncludeInRelease
```

`\@doclearpage` 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.

```
⟨latexrelease⟩\IncludeInRelease{2015/01/01}{\@doclearpage}%
⟨latexrelease⟩                              {float order in 2-column}%
⟨*2ekernel | latexrelease⟩
\def \@doclearpage {%
     \ifvoid\footins

       \ifvbox\@kludgeins
         {\setbox \@tempboxa \box \@kludgeins}%
⟨*trace⟩
         \fl@trace {kludgeins box made void}%
⟨/trace⟩
       \fi
       \setbox\@tempboxa\vsplit\@cclv to\z@ \unvbox\@tempboxa
       \setbox\@tempboxa\box\@cclv
       \xdef\@deferlist{\@toplist\@botlist\@deferlist}%
       \global \let \@toplist \@empty
       \global \let \@botlist \@empty
       \global \@colroom \@colht
       \ifx \@currlist\@empty
       \else
          \@latexerr{Float(s) lost}\@ehb
          \global \let \@currlist \@empty
       \fi
       \@makefcolumn\@deferlist
       \@whilesw\if@fcolmade \fi{\@opcol\@makefcolumn\@deferlist}%
       \if@twocolumn
         \if@firstcolumn

           \xdef\@deferlist{\@dbltoplist\@deferlist}%

           \global \let \@dbltoplist \@empty
           \global \@colht \textheight
           \begingroup
              \@dblfloatplacement

              \@makefcolumn\@deferlist
              \@whilesw\if@fcolmade \fi{\@outputpage
                                         \@makefcolumn\@deferlist}%

           \endgroup
         \else
           \vbox{}\clearpage
         \fi
       \fi
```

the next line is needed to avoid losing floats in certain circumstances a single call to the original `\doclearpage` will now no longer output all floats.

```
       \ifx\@deferlist\@empty \else\clearpage \fi
     \else
       \setbox\@cclv\vbox{\box\@cclv\vfil}%
       \@makecol\@opcol
       \clearpage
     \fi
}%
⟨/2ekernel | latexrelease⟩
⟨latexrelease⟩\EndIncludeInRelease
⟨latexrelease⟩\IncludeInRelease{0000/00/00}{\@doclearpage}%
⟨latexrelease⟩                             {float order in 2-column}%
⟨latexrelease⟩\def \@doclearpage {%
⟨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.

```
⟨latexrelease⟩       \ifvbox\@kludgeins
⟨latexrelease⟩         {\setbox \@tempboxa \box \@kludgeins}%
⟨*trace⟩
⟨latexrelease⟩         \fl@trace {kludgeins box made void}%
⟨/trace⟩
⟨latexrelease⟩       \fi
⟨latexrelease⟩       \setbox\@tempboxa\vsplit\@cclv to\z@ \unvbox\@tempboxa
⟨latexrelease⟩       \setbox\@tempboxa\box\@cclv
⟨latexrelease⟩       \xdef\@deferlist{\@toplist\@botlist\@deferlist}%

⟨latexrelease⟩       \global \let \@toplist \@empty
⟨latexrelease⟩       \global \let \@botlist \@empty
⟨latexrelease⟩       \global \@colroom \@colht
⟨latexrelease⟩       \ifx \@currlist\@empty
⟨latexrelease⟩       \else
⟨latexrelease⟩          \@latexerr{Float(s) lost}\@ehb

⟨latexrelease⟩          \global \let \@currlist \@empty
⟨latexrelease⟩       \fi
⟨latexrelease⟩       \@makefcolumn\@deferlist
⟨latexrelease⟩       \@whilesw\if@fcolmade \fi
⟨latexrelease⟩                     {\@opcol\@makefcolumn\@deferlist}%
⟨latexrelease⟩       \if@twocolumn
⟨latexrelease⟩         \if@firstcolumn
⟨latexrelease⟩           \xdef\@dbldeferlist{\@dbltoplist\@dbldeferlist}%

⟨latexrelease⟩           \global \let \@dbltoplist \@empty
⟨latexrelease⟩           \global \@colht \textheight
⟨latexrelease⟩           \begingroup
⟨latexrelease⟩              \@dblfloatplacement
⟨latexrelease⟩              \@makefcolumn\@dbldeferlist
⟨latexrelease⟩              \@whilesw\if@fcolmade \fi
⟨latexrelease⟩                    {\@outputpage\@makefcolumn\@dbldeferlist}%
⟨latexrelease⟩           \endgroup
⟨latexrelease⟩         \else
⟨latexrelease⟩           \vbox{}\clearpage
⟨latexrelease⟩         \fi
⟨latexrelease⟩       \fi
```

```
⟨latexrelease⟩     \else
⟨latexrelease⟩       \setbox\@cclv\vbox{\box\@cclv\vfil}%
⟨latexrelease⟩       \@makecol\@opcol
⟨latexrelease⟩       \clearpage
⟨latexrelease⟩     \fi
⟨latexrelease⟩  }%
⟨latexrelease⟩\EndIncludeInRelease
```

\@opcol Several changes in detail here.

```
⟨*2ekernel | fltrace⟩
\def \@opcol {%
  \if@twocolumn
    \@outputdblcol
  \else
    \@outputpage
⟨*trace⟩
    \fl@trace{PAGE: one column (float? see above) page completed}%
⟨/trace⟩
```

Not needed since it comes after \@outputpage:

```
%   \global\@colht\textheight
  \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.

```
  \global \@mparbottom \z@ \global \@textfloatsheight \z@
  \@floatplacement
}
⟨/2ekernel | fltrace⟩
```

\@makecol We must rewrite this macro to alllow for variations in page-makeup required by changes in page-length.

This uses a different macro if a special-length column is being produced.

```
⟨*2ekernel⟩
\gdef \@makecol {%
   \ifvoid\footins
     \setbox\@outputbox \box\@cclv
   \else
     \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.

```
       \boxmaxdepth \@maxdepth

%       \@tempdima\dp\@cclv
       \unvbox \@cclv
%       \vskip-\@tempdima
       \vskip \skip\footins

       \color@begingroup
         \normalcolor
         \footnoterule
         \unvbox \footins
       \color@endgroup
       }%
   \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.

```
   \let\@elt\relax
   \xdef\@freelist{\@freelist\@midlist}%
```

```
   \global \let \@midlist \@empty
   \@combinefloats
```

The variations start here in case `\enlargethispage` has been used.

```
   \ifvbox\@kludgeins
     \@makespecialcolbox
   \else
```

This extra reboxing is only needed to add the `\@texttop` and `\@textbotttom` 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@`.

```
     \setbox\@outputbox \vbox to\@colht {%
%       \boxmaxdepth \maxdepth                     %??
       \@texttop
       \dimen@ \dp\@outputbox
       \unvbox \@outputbox
       \vskip -\dimen@
       \@textbottom
       }%
   \fi
   \global \maxdepth \@maxdepth
}
```

`\@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.

```
\gdef \@reinserts{%
  \ifvoid\footins\else\insert\footins{\unvbox\footins}\fi
  \ifvbox\@kludgeins\insert\@kludgeins
```

```
                      {\unvbox\@kludgeins}\fi
}
⟨/2ekernel⟩
```

\@makespecialcolbox This implements certain variations in page-makeup.

```
⟨*2ekernel | fltrace⟩
\gdef \@makespecialcolbox {%
⟨*trace⟩
   \fl@trace{Kludgeins ht \the\ht\@kludgeins\space
                           dp \the\dp\@kludgeins\space
                           wd \the\wd\@kludgeins}%
⟨/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’.

```
   \setbox\@outputbox \vbox {%
    \@texttop
    \dimen@ \dp\@outputbox
    \unvbox\@outputbox
    \vskip-\dimen@
    }%
   \@tempdima \@colht
   \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 TEX level!).

This needs TEX3 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 bothe 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.

```
     \advance \@tempdima -\ht\@outputbox
     \advance \@tempdima \pageshrink
⟨*trace⟩
     \fl@trace {Natural ht of col: \the \ht\@outputbox}%
     \fl@trace {\string \@colht: \the \@colht}%
     \fl@trace {Pageshrink added: \the \pageshrink}%
     \fl@trace {Hence, space added: \the \@tempdima}%
⟨/trace⟩
     \setbox\@outputbox \vbox to \@colht {%
%       \boxmaxdepth \maxdepth
```

```
       \unvbox\@outputbox
       \vskip \@tempdima
       \@textbottom
       }%
```

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.

```
   \else
     \advance \@tempdima -\ht\@kludgeins
⟨*trace⟩
     \fl@trace {Natural ht of col: \the \ht\@outputbox}%
     \fl@trace {\string \@colht: \the \@colht}%
     \fl@trace {Extra size added: -\the \ht \@kludgeins}%
     \fl@trace {Hence, height of inner box: \the \@tempdima}%
     \fl@trace {Max? pageshrink available: \the \pageshrink}%
⟨/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.

```
     \setbox \@outputbox \vbox to \@colht {%
       \vbox to \@tempdima {%
         \unvbox\@outputbox
         \@textbottom}%
       \vss}%
   \fi
```

Finally we need to explicitly make the insert box void.

```
   {\setbox \@tempboxa \box \@kludgeins}%
⟨*trace⟩
     \fl@trace {kludgeins box made void}%
⟨/trace⟩
}
⟨/2ekernel | fltrace⟩
```

`\@texttop`
`\@textbottom`

These do nothing as a default.

```
⟨*2ekernel⟩
\let \@texttop \relax
\let \@textbottom \relax
```

`\@resetactivechars`
`\@activechar@info`

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.

```
\def\@activechar@info #1{%
      \@latex@info@no@line {Active #1 character found while
                            output routine is active
                            \MessageBreak
                            This may be a bug in a package file
                            you are using}%
}
```

Do not put any spaces in this next bit!

```
\begingroup
\obeylines\obeyspaces%
\catcode`\'\active%
\gdef\@resetactivechars{%
\def^^M{\@activechar@info{EOL}\space}%
\def {\@activechar@info{space}\space}%
\let'\active@math@prime}%
\endgroup
```

\@outputpage
\@shipoutsetup
\@writesetup

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.

```
⟨/2ekernel⟩
⟨latexrelease⟩\IncludeInRelease{2017/04/15}%
⟨latexrelease⟩   {\@outputpage}{Reset language for hyphenation}%
⟨*2ekernel | latexrelease⟩
\def\@outputpage{%
```

The \endgroup is put in by \aftergroup.

```
\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.

```
  \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.

```
  \language\document@default@language
```

The \catcode`\ = 10 was removed as it was considered useless (presumably because nothing gets tokenised during shipout).

This was put in as some error produced active spaces in a mark, I think.

Why was the hyphen reset?

```
\@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.

```
\global\let\@@if@newlist\if@newlist
\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
```

```
\@parboxrestore
```

... to here was in the command `\@writesetup`.

```
\shipout \vbox{%
  \set@typeset@protect
  \aftergroup \endgroup
```

Correct? or just restore by ending the group?

```
  \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`.

```
\if@specialpage
  \global\@specialpagefalse\@nameuse{ps@\@specialstyle}%
\fi
\if@twoside
  \ifodd\count\z@ \let\@thehead\@oddhead \let\@thefoot\@oddfoot
       \let\@themargin\oddsidemargin
  \else \let\@thehead\@evenhead
     \let\@thefoot\@evenfoot \let\@themargin\evensidemargin
  \fi
\fi
```

The rest was always inside the box.

RmS 91/08/15: aded this line:

```
  \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.

```
  \normalsize
```

Reset the space factors.

```
  \normalsfcodes
```

Reset these here (previously reset separately for head and foot)

```
  \let\label\@gobble
  \let\index\@gobble
  \let\glossary\@gobble
```

```
  \baselineskip\z@skip \lineskip\z@skip \lineskiplimit\z@
```

... to here was in the command `\@shipoutsetup`.

```
    \@begindvi
    \vskip \topmargin
    \moveright\@themargin \vbox {%
      \setbox\@tempboxa \vbox to\headheight{%
        \vfil
        \color@hbox
          \normalcolor
          \hb@xt@\textwidth{\@thehead}%
        \color@endbox
```

22 Feb 87

```
        }%
      \dp\@tempboxa \z@
      \box\@tempboxa
      \vskip \headsep
      \box\@outputbox
      \baselineskip \footskip
      \color@hbox
        \normalcolor
        \hb@xt@\textwidth{\@thefoot}%
      \color@endbox
      }%
    }%
```

`\endgroup` now inserted by `\aftergroup`

Restore `\if@newlist`

```
  \global\let\if@newlist\@@if@newlist
```

```
  \global \@colht \textheight
  \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.

```
  \let\firstmark\botmark
}
```

```
⟨/2ekernel | latexrelease⟩
⟨latexrelease⟩\EndIncludeInRelease
⟨latexrelease⟩\IncludeInRelease{0000/00/00}%
⟨latexrelease⟩  {\@outputpage}{Reset language for hyphenation}%
⟨latexrelease⟩\def\@outputpage{%
⟨latexrelease⟩\begingroup
⟨latexrelease⟩  \let \protect \noexpand
⟨latexrelease⟩  \@resetactivechars
⟨latexrelease⟩  \global\let\@@if@newlist\if@newlist
⟨latexrelease⟩  \global\@newlistfalse
⟨latexrelease⟩  \@parboxrestore
⟨latexrelease⟩  \shipout \vbox{%
⟨latexrelease⟩    \set@typeset@protect
⟨latexrelease⟩    \aftergroup \endgroup
⟨latexrelease⟩    \aftergroup \set@typeset@protect
⟨latexrelease⟩  \if@specialpage
⟨latexrelease⟩    \global\@specialpagefalse\@nameuse{ps@\@specialstyle}%
⟨latexrelease⟩  \fi
⟨latexrelease⟩  \if@twoside
⟨latexrelease⟩    \ifodd\count\z@
⟨latexrelease⟩         \let\@thehead\@oddhead \let\@thefoot\@oddfoot
⟨latexrelease⟩         \let\@themargin\oddsidemargin
⟨latexrelease⟩    \else \let\@thehead\@evenhead
⟨latexrelease⟩       \let\@thefoot\@evenfoot \let\@themargin\evensidemargin
⟨latexrelease⟩    \fi
⟨latexrelease⟩  \fi
⟨latexrelease⟩  \reset@font
⟨latexrelease⟩  \normalsize
⟨latexrelease⟩  \normalsfcodes
⟨latexrelease⟩  \let\label\@gobble
⟨latexrelease⟩  \let\index\@gobble
⟨latexrelease⟩  \let\glossary\@gobble
⟨latexrelease⟩  \baselineskip\z@skip \lineskip\z@skip \lineskiplimit\z@
⟨latexrelease⟩    \@begindvi
⟨latexrelease⟩    \vskip \topmargin
⟨latexrelease⟩    \moveright\@themargin \vbox {%
⟨latexrelease⟩      \setbox\@tempboxa \vbox to\headheight{%
⟨latexrelease⟩        \vfil
⟨latexrelease⟩        \color@hbox
⟨latexrelease⟩          \normalcolor
⟨latexrelease⟩          \hb@xt@\textwidth{\@thehead}%
⟨latexrelease⟩        \color@endbox
⟨latexrelease⟩        }%
⟨latexrelease⟩      \dp\@tempboxa \z@
⟨latexrelease⟩      \box\@tempboxa
⟨latexrelease⟩      \vskip \headsep
⟨latexrelease⟩      \box\@outputbox
⟨latexrelease⟩      \baselineskip \footskip
⟨latexrelease⟩      \color@hbox
⟨latexrelease⟩        \normalcolor
⟨latexrelease⟩        \hb@xt@\textwidth{\@thefoot}%
⟨latexrelease⟩      \color@endbox
⟨latexrelease⟩      }%
⟨latexrelease⟩    }%
⟨latexrelease⟩  \global\let\if@newlist\@@if@newlist
⟨latexrelease⟩  \global \@colht \textheight
⟨latexrelease⟩  \stepcounter{page}%
⟨latexrelease⟩  \let\firstmark\botmark
⟨latexrelease⟩}
⟨latexrelease⟩\EndIncludeInRelease
⟨*2ekernel⟩
```

\@begindvi 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.

```
\def \@begindvi{%
  \unvbox \@begindvibox
  \global\let \@begindvi \@empty
}
```

\@combinefloats \@cflt \@cflb 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.

```
\def \@combinefloats {%
%    \boxmaxdepth \maxdepth
    \ifx \@toplist\@empty \else \@cflt \fi
    \ifx \@botlist\@empty \else \@cflb \fi
}
```

```
\def \@cflt{%
    \let \@elt \@comflelt
    \setbox\@tempboxa \vbox{}%
    \@toplist
    \setbox\@outputbox \vbox{%
                             \boxmaxdepth \maxdepth
                             \unvbox\@tempboxa
                             \vskip -\floatsep
                             \topfigrule
                             \vskip \textfloatsep
                             \unvbox\@outputbox
                             }%
    \let\@elt\relax
    \xdef\@freelist{\@freelist\@toplist}%
    \global\let\@toplist\@empty
}
```

```
\def \@cflb {%
    \let\@elt\@comflelt
    \setbox\@tempboxa \vbox{}%
    \@botlist
    \setbox\@outputbox \vbox{%
                             \unvbox\@outputbox
                             \vskip \textfloatsep
                             \botfigrule
                             \unvbox\@tempboxa
                             \vskip -\floatsep
                             }%
    \let\@elt\relax
    \xdef\@freelist{\@freelist\@botlist}%
    \global \let \@botlist\@empty
}
```

\@comflelt
\@comdblflelt
\@combinedblfloats

```
\def\@comflelt#1{\setbox\@tempboxa
      \vbox{\unvbox\@tempboxa\box #1\vskip\floatsep}}
\def\@comdblflelt#1{\setbox\@tempboxa
      \vbox{\unvbox\@tempboxa\box #1\vskip\dblfloatsep}}
\def \@combinedblfloats{%
  \ifx \@dbltoplist \@empty
  \else
    \setbox\@tempboxa \vbox{}%
    \let \@elt \@comdblflelt
    \@dbltoplist
    \let \@elt \relax
    \xdef \@freelist {\@freelist\@dbltoplist}%
    \global\let \@dbltoplist \@empty
    \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.

```
      {%\boxmaxdepth\maxdepth   %% probably not needed, CAR
       \unvbox\@tempboxa\vskip-\dblfloatsep
```

Here we need different typesetting if the top float comes from `\@topnewpage`.

```
       \ifnum \@dbltopnum>\m@ne
         \dblfigrule
       \fi
       \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 pdfTEX and LuaTEX. 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.

```
       \unvbox\@outputbox
      }%
  \fi
}
⟨/2ekernel⟩
```

\@startcolumn
\@startdblcolumn

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`.

```
⟨*2ekernel | fltrace⟩
\def \@startcolumn {%
  \global \@colroom \@colht
  \@tryfcolumn \@deferlist
  \if@fcolmade
⟨*trace⟩
    \fl@trace{PAGE: float \if@twocolumn column \else page \fi
                completed}%
⟨/trace⟩
  \else

    \begingroup
      \let \reserved@b \@deferlist
      \global \let \@deferlist \@empty
      \let \@elt \@scolelt
      \reserved@b
    \endgroup
  \fi
}
```

This one does not need to set \@colht.

```
⟨/2ekernel | fltrace⟩
⟨latexrelease | fltrace⟩\IncludeInRelease{2015/01/01}%
⟨latexrelease | fltrace⟩  {\@startdblcolumn}{float order in 2-column}%
⟨*2ekernel | latexrelease | fltrace⟩
\def \@startdblcolumn {%
  \@tryfcolumn \@deferlist
  \if@fcolmade
⟨fltrace⟩    \fl@trace{PAGE: double float page completed}%
  \else
    \begingroup
      \let \reserved@b \@deferlist
      \global \let \@deferlist \@empty
      \let \@elt \@sdblcolelt
      \reserved@b
    \endgroup
  \fi
}%
⟨/2ekernel | latexrelease | fltrace⟩
⟨latexrelease | fltrace⟩\EndIncludeInRelease
⟨latexrelease | fltrace⟩\IncludeInRelease{0000/00/00}%
⟨latexrelease | fltrace⟩  {\@startdblcolumn}{float order in 2-column}%
⟨latexrelease | fltrace⟩\def \@startdblcolumn {%
```

Not needed since this always comes after \@outputpage:

```
⟨latexrelease | fltrace⟩% \global \@colht \textheight
⟨latexrelease | fltrace⟩  \@tryfcolumn \@dbldeferlist
⟨latexrelease | fltrace⟩  \if@fcolmade
⟨*trace⟩
⟨latexrelease | fltrace⟩    \fl@trace{PAGE: double float page completed}%
⟨/trace⟩
⟨latexrelease | fltrace⟩  \else

⟨latexrelease | fltrace⟩    \begingroup
⟨latexrelease | fltrace⟩      \let \reserved@b \@dbldeferlist
⟨latexrelease | fltrace⟩      \global \let \@dbldeferlist \@empty
⟨latexrelease | fltrace⟩      \let \@elt \@sdblcolelt
⟨latexrelease | fltrace⟩      \reserved@b
⟨latexrelease | fltrace⟩    \endgroup
⟨latexrelease | fltrace⟩  \fi
```

```
⟨latexrelease | fltrace⟩}%
⟨latexrelease | fltrace⟩\EndIncludeInRelease
⟨*2ekernel | fltrace⟩
```

\@tryfcolumn Now tests if its list is empty before any further exertion.

```
\def \@tryfcolumn #1{%
  \global \@fcolmadefalse
  \ifx #1\@empty
  \else
⟨*trace⟩
     \fl@trace{PAGE: try float \if@twocolumn column/page\else page\fi
                  ---\string #1}%
     \fl@trace{----- \string #1: #1}%
⟨/trace⟩
    \xdef\@trylist{#1}%
    \global \let \@failedlist \@empty
    \begingroup
      \let \@elt \@xtryfc \@trylist
    \endgroup
    \if@fcolmade
      \@vtryfc #1%
    \fi
  \fi
}
⟨/2ekernel | fltrace⟩
⟨*2ekernel⟩
```

\@scolelt

```
\def\@scolelt#1{\def\@currbox{#1}\@addtonextcol}
```

\@sdblcolelt

```
\def\@sdblcolelt#1{\def\@currbox{#1}\@addtodblcol}
```

\@vtryfc

```
\def\@vtryfc #1{%
  \global\setbox\@outputbox\vbox{}%
  \let\@elt\@wtryfc
  \@flsucceed
  \global\setbox\@outputbox \vbox to\@colht{%
    \vskip \@fptop
    \vskip -\@fpsep
    \unvbox \@outputbox
    \vskip \@fpbot}%
  \let\@elt\relax
  \xdef #1{\@failedlist\@flfail}%
  \xdef\@freelist{\@freelist\@flsucceed}}
```

\@wtryfc

```
\def\@wtryfc #1{%
  \global\setbox\@outputbox\vbox{%
    \unvbox\@outputbox
    \vskip\@fpsep
    \box #1}}
```

\@xtryfc

```
⟨/2ekernel⟩
⟨latexrelease⟩\IncludeInRelease{2015/01/01}{\@xtryfc}%
⟨latexrelease⟩                             {float order in 2-column}%
⟨*2ekernel | latexrelease⟩
\def\@xtryfc #1{%
  \@next\reserved@a\@trylist{}{}%
  \@currtype \count #1%
  \divide\@currtype\@xxxii
  \multiply\@currtype\@xxxii
  \@bitor \@currtype \@failedlist
  \@testfp #1%
  \@testwrongwidth #1%
  \ifdim \ht #1>\@colht
     \@testtrue
  \fi
  \if@test
    \@cons\@failedlist #1%
  \else
    \@ytryfc #1%
  \fi}%
⟨/2ekernel | latexrelease⟩
⟨latexrelease⟩\EndIncludeInRelease
⟨latexrelease⟩\IncludeInRelease{0000/00/00}{\@xtryfc}%
⟨latexrelease⟩                             {float order in 2-column}%
⟨latexrelease⟩\def\@xtryfc #1{%
⟨latexrelease⟩  \@next\reserved@a\@trylist{}{}%
⟨latexrelease⟩  \@currtype \count #1%
⟨latexrelease⟩  \divide\@currtype\@xxxii
⟨latexrelease⟩  \multiply\@currtype\@xxxii
⟨latexrelease⟩  \@bitor \@currtype \@failedlist
⟨latexrelease⟩  \@testfp #1%
⟨latexrelease⟩  \ifdim \ht #1>\@colht
⟨latexrelease⟩    \@testtrue
⟨latexrelease⟩  \fi
⟨latexrelease⟩  \if@test
⟨latexrelease⟩    \@cons\@failedlist #1%
⟨latexrelease⟩  \else
⟨latexrelease⟩    \@ytryfc #1%
⟨latexrelease⟩  \fi}%
⟨latexrelease⟩\EndIncludeInRelease
⟨*2ekernel⟩
```

\@ytryfc

```
\def\@ytryfc #1{%
  \begingroup
    \gdef\@flsucceed{\@elt #1}%
    \global\let\@flfail\@empty
    \@tempdima\ht #1%
    \let\@elt\@ztryfc
    \@trylist
    \ifdim \@tempdima >\@fpmin
      \global\@fcolmadetrue
    \else
      \@cons\@failedlist #1%
    \fi
  \endgroup
  \if@fcolmade
    \let\@elt\@gobble
  \fi}
```

\@ztryfc

```
⟨/2ekernel⟩
⟨latexrelease⟩\IncludeInRelease{2015/01/01}{@ztryfc}%
⟨latexrelease⟩                          {float order in 2-column}%
⟨*2ekernel | latexrelease⟩
\def\@ztryfc #1{%
  \@tempcnta\count #1%
  \divide\@tempcnta\@xxxii
  \multiply\@tempcnta\@xxxii
  \@bitor \@tempcnta {\@failedlist \@flfail}%
  \@testfp #1%
```

not in fixfloats?

```
  \@testwrongwidth #1%
```

```
  \@tempdimb\@tempdima
  \advance\@tempdimb\ht #1%
  \advance\@tempdimb\@fpsep
  \ifdim \@tempdimb >\@colht
    \@testtrue
  \fi
  \if@test
    \@cons\@flfail #1%
  \else
    \@cons\@flsucceed #1%
    \@tempdima\@tempdimb
  \fi}%
⟨/2ekernel | latexrelease⟩
⟨latexrelease⟩\EndIncludeInRelease
⟨latexrelease⟩\IncludeInRelease{0000/00/00}{@ztryfc}%
⟨latexrelease⟩                          {float order in 2-column}%
⟨latexrelease⟩\def\@ztryfc #1{%
⟨latexrelease⟩  \@tempcnta \count#1%
⟨latexrelease⟩  \divide\@tempcnta\@xxxii
⟨latexrelease⟩  \multiply\@tempcnta\@xxxii
⟨latexrelease⟩  \@bitor \@tempcnta {\@failedlist \@flfail}%
⟨latexrelease⟩  \@testfp #1%
⟨latexrelease⟩  \@tempdimb\@tempdima
⟨latexrelease⟩  \advance\@tempdimb \ht#1%
⟨latexrelease⟩  \advance\@tempdimb\@fpsep
⟨latexrelease⟩  \ifdim \@tempdimb >\@colht
⟨latexrelease⟩    \@testtrue
⟨latexrelease⟩  \fi
⟨latexrelease⟩  \if@test
⟨latexrelease⟩    \@cons\@flfail #1%
⟨latexrelease⟩  \else
⟨latexrelease⟩    \@cons\@flsucceed #1%
```

```
⟨latexrelease⟩      \@tempdima\@tempdimb
⟨latexrelease⟩   \fi}%
⟨latexrelease⟩\EndIncludeInRelease
```

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.

```
⟨*2ekernel | fltrace⟩
\def \@addtobot {%
⟨*trace⟩
   \fl@trace{***Start addtobot}%
⟨/trace⟩
   \@getfpsbit 4\relax
⟨*trace⟩
   \fl@trace{fpstype \ifodd \@tempcnta OK \else not \fi bot:
                                                    \the \@fpstype}%
⟨/trace⟩
   \ifodd \@tempcnta
     \@flsetnum \@botnum
     \ifnum \@botnum>\z@
       \@tempswafalse
       \@flcheckspace \@botroom \@botlist
       \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?

```
         \global \maxdepth \z@
         \@flupdates \@botnum \@botroom \@botlist
⟨*trace⟩
         \fl@trace{colroom (after-bot) = \the \@colroom}%
         \fl@trace{colnum (after-bot) = \the \@colnum}%
         \fl@trace{botnum (after-bot) = \the \@botnum}%
         \fl@trace{***Success: bot}%
⟨/trace⟩
         \@inserttrue
       \fi
⟨*trace⟩
     \else
       \fl@trace{Fail: botnum = \the \@botnum:
                                 fpstype \the \@fpstype=ORD?}%
       \ifnum \@fpstype<\sixt@@n
         \fl@trace{ERROR: !b float not successful (addtobot)}%
       \fi
⟨/trace⟩
     \fi
   \fi
}
```

`\@addtotoporbot` Lots of changes.

```
\def \@addtotoporbot {%
⟨*trace⟩
   \fl@trace{***Start addtotoporbot}%
⟨/trace⟩
```

```
   \@getfpsbit \tw@
⟨*trace⟩
   \fl@trace{fpstype \ifodd \@tempcnta OK \else not \fi top:
                                                     \the \@fpstype}%
⟨/trace⟩
   \ifodd \@tempcnta
     \@flsetnum \@topnum
     \ifnum \@topnum>\z@
       \@tempswafalse
       \@flcheckspace \@toproom \@toplist
       \if@tempswa
         \@bitor\@currtype{\@midlist\@botlist}%
⟨*trace⟩
           \fl@trace{(mid+bot)list: \@midlist, \@botlist:
                              (addtotoporbot-before)}%
⟨/trace⟩
         \if@test
⟨*trace⟩
         \fl@trace{type already on list: mid or bot---sent to addtobot}%
⟨/trace⟩
         \else
          \@flupdates \@topnum \@toproom \@toplist
⟨*trace⟩
          \fl@trace{colroom (after-top) = \the \@colroom}%
          \fl@trace{colnum (after-top) = \the \@colnum}%
          \fl@trace{topnum (after-top) = \the \@topnum}%
          \fl@trace{***Success: top}%
⟨/trace⟩
          \@inserttrue
         \fi
       \fi
⟨*trace⟩
     \else
       \fl@trace{Fail: topnum = \the \@topnum: fpstype
                                            \the \@fpstype=ORD?}%
       \ifnum \@fpstype<\sixt@@n
         \fl@trace{ERROR: !t float not successful (addtotoporbot)}%
       \fi
⟨/trace⟩
     \fi
   \fi
   \if@insert
   \else
⟨*trace⟩
     \fl@trace{sent to addtobot (addtotoporbot)}%
⟨/trace⟩
     \@addtobot
   \fi
}
⟨/2ekernel | fltrace⟩
```

`\@addtocurcol` Lots of changes.

```
⟨latexrelease | fltrace | flafter⟩\IncludeInRelease{2015/01/01}%
⟨latexrelease | fltrace | flafter⟩  {\@addtocurcol}{float order in 2-column}%
```

```
⟨*2ekernel | latexrelease | fltrace | flafter⟩
\def \@addtocurcol {%
⟨*trace⟩
  \fl@trace{***Start addtocurcol}%
⟨/trace⟩
   \@insertfalse
   \@setfloattypecounts
   \ifnum \@fpstype=8
⟨*trace⟩
     \fl@trace{fpstype !p only (addtocurcol): \the \@fpstype = 8?}%
⟨/trace⟩
   \else
     \ifnum \@fpstype=24
⟨*trace⟩
       \fl@trace{fpstype p only (addtocurcol): \the \@fpstype = 24?}%
⟨/trace⟩
     \else
       \@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.

```
⟨*trace⟩
       \fl@trace{textfloatsheight (before) = \the \@textfloatsheight}%
⟨/trace⟩
       \advance \@textmin \@textfloatsheight
       \@reqcolroom \@pageht
```

This line must be removed since `\@specialoutput` changed.

```
%        \advance \@reqcolroom \@pagedp
⟨*trace⟩
       \fl@trace{textmin + textfloatsheight: \the \@textmin}%
       \fl@trace{page-so-far: \the \@reqcolroom}%
⟨/trace⟩
       \ifdim \@textmin>\@reqcolroom
         \@reqcolroom \@textmin
⟨*trace⟩
         \fl@trace{ORD? textmin being used}%
⟨/trace⟩
       \fi
       \advance \@reqcolroom \ht\@currbox
⟨*trace⟩
       \fl@trace{float size = \the \ht \@currbox (addtocurcol)}%
       \fl@trace{colroom = \the \@colroom (addtocurcol)}%
       \fl@trace{reqcolroom = \the \@reqcolroom (addtocurcol)}%
⟨/trace⟩
       \ifdim \@colroom>\@reqcolroom
         \@flsetnum \@colnum
         \ifnum \@colnum>\z@
           \@bitor\@currtype\@deferlist
```

We need to defer the float also if its width doesn't fit.

```
          \@testwrongwidth\@currbox
```

```
⟨*trace⟩
           \fl@trace{deferlist: \@deferlist: (addtocurcol-before)}%
⟨/trace⟩
           \if@test
⟨*trace⟩
             \fl@trace{type already on list: defer (addtocurcol)}%
⟨/trace⟩
           \else
             \@bitor\@currtype\@botlist
⟨*trace⟩
           \fl@trace{botlist: \@botlist: (addtocurcol-before)}%
⟨/trace⟩
             \if@test
⟨*trace⟩
               \fl@trace{type already on list: bot---sent to addtobot}%
⟨/trace⟩
               \@addtobot
             \else
⟨*trace⟩
               \fl@trace{fpstype \ifodd \@tempcnta OK \else not \fi
                      here: \the \@fpstype}%
⟨/trace⟩
               \ifodd \count\@currbox
                 \advance \@reqcolroom \intextsep
                 \ifdim \@colroom>\@reqcolroom
                   \global \advance \@colnum \m@ne
                   \global \advance \@textfloatsheight \ht\@currbox
```

This may sometimes give an overestimate.

```
                   \global \advance \@textfloatsheight 2\intextsep
                   \@cons \@midlist \@currbox
⟨*trace⟩
                 \fl@trace{***Success: here}%
                 \fl@trace{textfloatsheight (after-here) =
                      \the \@textfloatsheight}%
                 \fl@trace{colnum (after-here) = \the \@colnum}%
⟨/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).

```
                   \if@nobreak
                     \nobreak
                     \@nobreakfalse
```

```
                    \everypar{}%
                  \else
                    \addpenalty \interlinepenalty
                  \fi
                  \vskip \intextsep
                  \box\@currbox
                  \penalty\interlinepenalty
                  \vskip\intextsep
                  \ifnum\outputpenalty <-\@Mii \vskip -\parskip\fi
```

Typesetting ends here.

```
                  \outputpenalty \z@
                  \@inserttrue
⟨*trace⟩
                \else
                  \fl@trace{Fail---no room at 2nd test of colroom
                                 (addtocorcol \string\intextsep)}%
⟨/trace⟩
                \fi
              \fi
              \if@insert
              \else
```

Next set of docstrip guards are a bit weird, essentially `\@addtotoporbot` ends up inside the kernel and the `fltrace` package and `\@addtobot` shows up in the `flafter` package. Guess that could have been done a bit more obvious :-)

```
⟨*2ekernel | fltrace | latexrelease⟩
⟨*trace⟩
                \fl@trace{not here: sent to addtotoporbot}%
⟨/trace⟩
                \@addtotoporbot
⟨/2ekernel | fltrace | latexrelease⟩
⟨*!2ekernel&!fltrace&!latexrelease⟩
⟨*trace⟩
                \fl@trace{not here: sent to addtobot}%
⟨/trace⟩
                \@addtobot
⟨/!2ekernel&!fltrace&!latexrelease⟩
              \fi
            \fi
          \fi
⟨*trace⟩
        \else
          \fl@trace{Fail: colnum = \the \@colnum:
                       fpstype \the \@fpstype=ORD?}%
          \ifnum \@fpstype<\sixt@@n
            \fl@trace{ERROR: BANG float not successful (addtocurcol)}%
          \fi
⟨/trace⟩
        \fi
⟨*trace⟩
      \else
        \fl@trace{Fail---no room: fl box ht: \the \ht \@currbox
                                                     (addtocurcol)}%
⟨/trace⟩
```

```
       \fi
     \fi
   \fi
   \if@insert
   \else
     \@resethfps
⟨*trace⟩
     \fl@trace{put on deferlist (addtocurcol)}%
⟨/trace⟩
     \@cons\@deferlist\@currbox
⟨*trace⟩
     \fl@trace{deferlist: \@deferlist: (addtocurcol-after)}%
⟨/trace⟩
   \fi
}%
⟨/2ekernel | latexrelease | fltrace | flafter⟩
⟨latexrelease | fltrace | flafter⟩\EndIncludeInRelease
⟨latexrelease | fltrace | flafter⟩\IncludeInRelease{0000/00/00}%
⟨latexrelease | fltrace | flafter⟩  {\@addtocurcol}{float order in 2-column}%
⟨latexrelease | fltrace | flafter⟩\def \@addtocurcol {%
⟨*trace⟩
⟨latexrelease | fltrace | flafter⟩  \fl@trace{***Start addtocurcol}%
⟨/trace⟩
⟨latexrelease | fltrace | flafter⟩   \@insertfalse
⟨latexrelease | fltrace | flafter⟩   \@setfloattypecounts
⟨latexrelease | fltrace | flafter⟩   \ifnum \@fpstype=8
⟨*trace⟩
⟨latexrelease | fltrace | flafter⟩     \fl@trace{fpstype !p only (addtocurcol):
⟨latexrelease | fltrace | flafter⟩                            \the \@fpstype = 8?}%
⟨/trace⟩
⟨latexrelease | fltrace | flafter⟩   \else
⟨latexrelease | fltrace | flafter⟩     \ifnum \@fpstype=24
⟨*trace⟩
⟨latexrelease | fltrace | flafter⟩      \fl@trace{fpstype p only (addtocurcol):
⟨latexrelease | fltrace | flafter⟩                           \the \@fpstype = 24?}%
⟨/trace⟩
⟨latexrelease | fltrace | flafter⟩     \else
⟨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.

```
⟨*trace⟩
⟨latexrelease | fltrace | flafter⟩       \fl@trace{textfloatsheight (before) =
⟨latexrelease | fltrace | flafter⟩                        \the \@textfloatsheight}%
⟨/trace⟩
⟨latexrelease | fltrace | flafter⟩       \advance \@textmin \@textfloatsheight
⟨latexrelease | fltrace | flafter⟩       \@reqcolroom \@pageht
```

This line must be removed since `\@specialoutput` changed.

```
%       \advance \@reqcolroom \@pagedp
⟨*trace⟩
⟨latexrelease | fltrace | flafter⟩       \fl@trace{textmin + textfloatsheight:
⟨latexrelease | fltrace | flafter⟩                                 \the \@textmin}%
```

```
⟨latexrelease | fltrace | flafter⟩        \fl@trace{page-so-far: \the \@reqcolroom}%
⟨latexrelease | fltrace | flafter⟩
⟨/trace⟩
⟨latexrelease | fltrace | flafter⟩        \ifdim \@textmin>\@reqcolroom
⟨latexrelease | fltrace | flafter⟩          \@reqcolroom \@textmin
⟨*trace⟩
⟨latexrelease | fltrace | flafter⟩          \fl@trace{ORD? textmin being used}%
⟨/trace⟩
⟨latexrelease | fltrace | flafter⟩        \fi
⟨latexrelease | fltrace | flafter⟩        \advance \@reqcolroom \ht\@currbox
⟨*trace⟩
⟨latexrelease | fltrace | flafter⟩        \fl@trace{float size =
⟨latexrelease | fltrace | flafter⟩                \the \ht \@currbox (addtocurcol)}%
⟨latexrelease | fltrace | flafter⟩        \fl@trace{colroom =
⟨latexrelease | fltrace | flafter⟩                    \the \@colroom (addtocurcol)}%
⟨latexrelease | fltrace | flafter⟩        \fl@trace{reqcolroom =
⟨latexrelease | fltrace | flafter⟩                 \the \@reqcolroom (addtocurcol)}%
⟨/trace⟩
⟨latexrelease | fltrace | flafter⟩        \ifdim \@colroom>\@reqcolroom
⟨latexrelease | fltrace | flafter⟩          \@flsetnum \@colnum
⟨latexrelease | fltrace | flafter⟩          \ifnum \@colnum>\z@
⟨latexrelease | fltrace | flafter⟩            \@bitor\@currtype\@deferlist
⟨*trace⟩
⟨latexrelease | fltrace | flafter⟩            \fl@trace{deferlist:
⟨latexrelease | fltrace | flafter⟩               \@deferlist: (addtocurcol-before)}%
⟨/trace⟩
⟨latexrelease | fltrace | flafter⟩            \if@test
⟨*trace⟩
⟨latexrelease | fltrace | flafter⟩              \fl@trace{type already on list:
⟨latexrelease | fltrace | flafter⟩                             defer (addtocurcol)}%
⟨/trace⟩
⟨latexrelease | fltrace | flafter⟩            \else
⟨latexrelease | fltrace | flafter⟩              \@bitor\@currtype\@botlist
⟨*trace⟩
⟨latexrelease | fltrace | flafter⟩            \fl@trace{botlist: \@botlist:
⟨latexrelease | fltrace | flafter⟩                            (addtocurcol-before)}%
⟨/trace⟩
⟨latexrelease | fltrace | flafter⟩              \if@test
⟨*trace⟩
⟨latexrelease | fltrace | flafter⟩                \fl@trace{type already on list:
⟨latexrelease | fltrace | flafter⟩                          bot---sent to addtobot}%
⟨/trace⟩
⟨latexrelease | fltrace | flafter⟩                \@addtobot
⟨latexrelease | fltrace | flafter⟩              \else
⟨*trace⟩
⟨latexrelease | fltrace | flafter⟩                \fl@trace{fpstype
⟨latexrelease | fltrace | flafter⟩                \ifodd \@tempcnta OK \else not \fi
⟨latexrelease | fltrace | flafter⟩                here: \the \@fpstype}%
⟨/trace⟩
⟨latexrelease | fltrace | flafter⟩                \ifodd \count\@currbox
⟨latexrelease | fltrace | flafter⟩                  \advance \@reqcolroom \intextsep
⟨latexrelease | fltrace | flafter⟩                  \ifdim \@colroom>\@reqcolroom
⟨latexrelease | fltrace | flafter⟩                   \global \advance \@colnum \m@ne
⟨latexrelease | fltrace | flafter⟩                   \global \advance
```

```
⟨latexrelease | fltrace | flafter⟩                  \@textfloatsheight\ht\@currbox
```

This may sometimes give an overestimate.

```
⟨latexrelease | fltrace | flafter⟩                 \global \advance
⟨latexrelease | fltrace | flafter⟩                  \@textfloatsheight 2\intextsep
⟨latexrelease | fltrace | flafter⟩                  \@cons \@midlist \@currbox
⟨*trace⟩
⟨latexrelease | fltrace | flafter⟩                \fl@trace{***Success: here}%
⟨latexrelease | fltrace | flafter⟩                \fl@trace{textfloatsheight
⟨latexrelease | fltrace | flafter⟩                     (after-here) =
⟨latexrelease | fltrace | flafter⟩                     \the \@textfloatsheight}%
⟨latexrelease | fltrace | flafter⟩                \fl@trace{colnum (after-here) =
⟨latexrelease | fltrace | flafter⟩                         \the \@colnum}%
⟨/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).

```
⟨latexrelease | fltrace | flafter⟩                  \if@nobreak
⟨latexrelease | fltrace | flafter⟩                    \nobreak
⟨latexrelease | fltrace | flafter⟩                    \@nobreakfalse
⟨latexrelease | fltrace | flafter⟩                    \everypar{}%
⟨latexrelease | fltrace | flafter⟩                  \else
⟨latexrelease | fltrace | flafter⟩                    \addpenalty\interlinepenalty
⟨latexrelease | fltrace | flafter⟩                  \fi
⟨latexrelease | fltrace | flafter⟩                  \vskip \intextsep
⟨latexrelease | fltrace | flafter⟩                  \box\@currbox
⟨latexrelease | fltrace | flafter⟩                  \penalty\interlinepenalty
⟨latexrelease | fltrace | flafter⟩                  \vskip\intextsep
⟨latexrelease | fltrace | flafter⟩                  \ifnum\outputpenalty
⟨latexrelease | fltrace | flafter⟩                              <-\@Mii \vskip
⟨latexrelease | fltrace | flafter⟩                       -\parskip\fi
```

Typesetting ends here.

```
⟨latexrelease | fltrace | flafter⟩                  \outputpenalty \z@
⟨latexrelease | fltrace | flafter⟩                  \@inserttrue
⟨*trace⟩
⟨latexrelease | fltrace | flafter⟩                \else
⟨latexrelease | fltrace | flafter⟩  \fl@trace{Fail---no room at 2nd test of colroom
⟨latexrelease | fltrace | flafter⟩               (addtocorcol \string\intextsep)}%
⟨/trace⟩
⟨latexrelease | fltrace | flafter⟩                \fi
⟨latexrelease | fltrace | flafter⟩              \fi
⟨latexrelease | fltrace | flafter⟩              \if@insert
```

```
⟨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 :-)

```
⟨*2ekernel | fltrace⟩
⟨*trace⟩
⟨latexrelease | fltrace | flafter⟩      \fl@trace{not here: sent to addtotoporbot}%
⟨/trace⟩
⟨latexrelease | fltrace | flafter⟩                 \@addtotoporbot
⟨/2ekernel | fltrace⟩
⟨*!2ekernel&!autoload&!fltrace⟩
⟨*trace⟩
⟨latexrelease | fltrace | flafter⟩       \fl@trace{not here: sent to addtobot}%
⟨/trace⟩
⟨latexrelease | fltrace | flafter⟩                 \@addtobot
⟨/!2ekernel&!autoload&!fltrace⟩
⟨latexrelease | fltrace | flafter⟩               \fi
⟨latexrelease | fltrace | flafter⟩             \fi
⟨latexrelease | fltrace | flafter⟩           \fi
⟨*trace⟩
⟨latexrelease | fltrace | flafter⟩         \else
⟨latexrelease | fltrace | flafter⟩         \fl@trace{Fail: colnum = \the \@colnum:
⟨latexrelease | fltrace | flafter⟩                    fpstype \the \@fpstype=ORD?}%
⟨latexrelease | fltrace | flafter⟩         \ifnum \@fpstype<\sixt@@n
⟨latexrelease | fltrace | flafter⟩  \fl@trace{ERROR: BANG float not successful
⟨latexrelease | fltrace | flafter⟩                                  (addtocurcol)}%
⟨latexrelease | fltrace | flafter⟩           \fi
⟨/trace⟩
⟨latexrelease | fltrace | flafter⟩         \fi
⟨*trace⟩
⟨latexrelease | fltrace | flafter⟩       \else
⟨latexrelease | fltrace | flafter⟩         \fl@trace{Fail---no room: fl box ht:
⟨latexrelease | fltrace | flafter⟩               \the \ht \@currbox (addtocurcol)}%
⟨/trace⟩
⟨latexrelease | fltrace | flafter⟩       \fi
⟨latexrelease | fltrace | flafter⟩     \fi
⟨latexrelease | fltrace | flafter⟩   \fi
⟨latexrelease | fltrace | flafter⟩   \if@insert
⟨latexrelease | fltrace | flafter⟩   \else
⟨latexrelease | fltrace | flafter⟩     \@resethfps
⟨*trace⟩
⟨latexrelease | fltrace | flafter⟩     \fl@trace{put on deferlist (addtocurcol)}%
⟨/trace⟩
⟨latexrelease | fltrace | flafter⟩     \@cons\@deferlist\@currbox
⟨*trace⟩
⟨latexrelease | fltrace | flafter⟩     \fl@trace{deferlist: \@deferlist:
⟨latexrelease | fltrace | flafter⟩                          (addtocurcol-after)}%
⟨/trace⟩
⟨latexrelease | fltrace | flafter⟩   \fi
⟨latexrelease | fltrace | flafter⟩  }%
⟨latexrelease | fltrace | flafter⟩\EndIncludeInRelease
```

`\@addtonextcol` Lots of changes.

```
⟨latexrelease | fltrace⟩\IncludeInRelease{2015/01/01}
⟨latexrelease | fltrace⟩  {\@addtonextcol}{float order in 2-column}%
⟨*2ekernel | latexrelease | fltrace⟩
\def\@addtonextcol{%
  \begingroup
⟨*trace⟩
   \fl@trace{***Start addtonextcol}%
⟨/trace⟩
   \@insertfalse
   \@setfloattypecounts
   \ifnum \@fpstype=8
⟨*trace⟩
     \fl@trace{fpstype not curcol: \the \@fpstype = 8?}%
⟨/trace⟩
   \else
     \ifnum \@fpstype=24
⟨*trace⟩
       \fl@trace{fpstype not curcol: \the \@fpstype = 24?}%
⟨/trace⟩
     \else
       \@flsettextmin
⟨*trace⟩
       \fl@trace{text-so-far: 0pt (top of col)}%
⟨/trace⟩
       \@reqcolroom \ht\@currbox
⟨*trace⟩
       \fl@trace{float size: \the \@reqcolroom (addtonextcol)}%
⟨/trace⟩
       \advance \@reqcolroom \@textmin
⟨*trace⟩
       \fl@trace{colroom = \the \@colroom (addtonextcol)}%
       \fl@trace{reqcolroom = \the \@reqcolroom (addtonextcol)}%
⟨/trace⟩
       \ifdim \@colroom>\@reqcolroom
         \@flsetnum \@colnum
         \ifnum\@colnum>\z@
            \@bitor\@currtype\@deferlist
⟨*trace⟩
            \fl@trace{deferlist: \@deferlist: (addtonextcol-before)}%
⟨/trace⟩
            \@testwrongwidth\@currbox
            \if@test
⟨*trace⟩
              \fl@trace{type already on list: defer (addtonextcol)}%
⟨/trace⟩
            \else
⟨*trace⟩
              \fl@trace{sent to addtotoporbot (addtonextcol)}%
⟨/trace⟩
              \@addtotoporbot
            \fi
         \fi
⟨*trace⟩
```

```
       \else
         \fl@trace{Fail---no room: fl box ht: \the \ht \@currbox
                                                  (addtonextcol)}%
⟨/trace⟩
       \fi
     \fi
   \fi
   \if@insert
   \else
⟨*trace⟩
     \fl@trace{put back on deferlist (addtonextcol)}%
⟨/trace⟩
     \@cons\@deferlist\@currbox
⟨*trace⟩
     \fl@trace{deferlist: \@deferlist: (addtonextcol-after)}%
⟨/trace⟩
   \fi
⟨*trace⟩
  \fl@trace{End of addtonextcol -- locally counts:}%
  \fl@trace{col: \the\@colnum. top: \the \@topnum. bot: \the \@botnum.}%
⟨/trace⟩
  \endgroup
⟨*trace⟩
 \fl@trace{End of addtonextcol -- globally counts:}%
 \fl@trace{col: \the\@colnum. top: \the \@topnum. bot: \the \@botnum.}%
⟨/trace⟩
}%
⟨/2ekernel | latexrelease | fltrace⟩
⟨latexrelease | fltrace⟩\EndIncludeInRelease
⟨latexrelease | fltrace⟩\IncludeInRelease{0000/00/00}%
⟨latexrelease | fltrace⟩  {\@addtonextcol}{float order in 2-column}%
⟨latexrelease | fltrace⟩\def\@addtonextcol{%
⟨latexrelease | fltrace⟩  \begingroup
⟨*trace⟩
⟨latexrelease | fltrace⟩   \fl@trace{***Start addtonextcol}%
⟨/trace⟩
⟨latexrelease | fltrace⟩   \@insertfalse
⟨latexrelease | fltrace⟩   \@setfloattypecounts
⟨latexrelease | fltrace⟩   \ifnum \@fpstype=8
⟨*trace⟩
⟨latexrelease | fltrace⟩     \fl@trace{fpstype not curcol:
⟨latexrelease | fltrace⟩                    \the \@fpstype = 8?}%
⟨/trace⟩
⟨latexrelease | fltrace⟩   \else
⟨latexrelease | fltrace⟩     \ifnum \@fpstype=24
⟨*trace⟩
⟨latexrelease | fltrace⟩       \fl@trace{fpstype not curcol:
⟨latexrelease | fltrace⟩                       \the \@fpstype = 24?}%
⟨/trace⟩
⟨latexrelease | fltrace⟩     \else
⟨latexrelease | fltrace⟩       \@flsettextmin
⟨*trace⟩
⟨latexrelease | fltrace⟩       \fl@trace{text-so-far: 0pt (top of col)}%
⟨/trace⟩
```

```
⟨latexrelease | fltrace⟩       \@reqcolroom \ht\@currbox
⟨*trace⟩
⟨latexrelease | fltrace⟩       \fl@trace{float size:
⟨latexrelease | fltrace⟩               \the \@reqcolroom (addtonextcol)}%
⟨latexrelease | fltrace⟩
⟨/trace⟩
⟨latexrelease | fltrace⟩       \advance \@reqcolroom \@textmin
⟨*trace⟩
⟨latexrelease | fltrace⟩       \fl@trace{colroom =
⟨latexrelease | fltrace⟩                  \the \@colroom (addtonextcol)}%
⟨latexrelease | fltrace⟩       \fl@trace{reqcolroom =
⟨latexrelease | fltrace⟩               \the \@reqcolroom (addtonextcol)}%
⟨/trace⟩
⟨latexrelease | fltrace⟩       \ifdim \@colroom>\@reqcolroom
⟨latexrelease | fltrace⟩         \@flsetnum \@colnum
⟨latexrelease | fltrace⟩         \ifnum\@colnum>\z@
⟨latexrelease | fltrace⟩            \@bitor\@currtype\@deferlist
⟨*trace⟩
⟨latexrelease | fltrace⟩            \fl@trace{deferlist: \@deferlist:
⟨latexrelease | fltrace⟩                          (addtonextcol-before)}%
⟨/trace⟩
⟨latexrelease | fltrace⟩            \if@test
⟨*trace⟩
⟨latexrelease | fltrace⟩              \fl@trace{type already on list:
⟨latexrelease | fltrace⟩                           defer (addtonextcol)}%
⟨/trace⟩
⟨latexrelease | fltrace⟩            \else
⟨*trace⟩
⟨latexrelease | fltrace⟩              \fl@trace{sent to addtotoporbot
⟨latexrelease | fltrace⟩                                 (addtonextcol)}%
⟨/trace⟩
⟨latexrelease | fltrace⟩              \@addtotoporbot
⟨latexrelease | fltrace⟩            \fi
⟨latexrelease | fltrace⟩         \fi
⟨*trace⟩
⟨latexrelease | fltrace⟩       \else
⟨latexrelease | fltrace⟩         \fl@trace{Fail---no room: fl box ht:
⟨latexrelease | fltrace⟩              \the \ht \@currbox (addtonextcol)}%
⟨/trace⟩
⟨latexrelease | fltrace⟩       \fi
⟨latexrelease | fltrace⟩     \fi
⟨latexrelease | fltrace⟩   \fi
⟨latexrelease | fltrace⟩   \if@insert
⟨latexrelease | fltrace⟩   \else
⟨*trace⟩
⟨latexrelease | fltrace⟩     \fl@trace{put back on deferlist
⟨latexrelease | fltrace⟩                                (addtonextcol)}%
⟨/trace⟩
⟨latexrelease | fltrace⟩     \@cons\@deferlist\@currbox
⟨*trace⟩
⟨latexrelease | fltrace⟩     \fl@trace{deferlist: \@deferlist:
⟨latexrelease | fltrace⟩                           (addtonextcol-after)}%
⟨/trace⟩
⟨latexrelease | fltrace⟩   \fi
```

```
⟨*trace⟩
⟨latexrelease | fltrace⟩   \fl@trace{End of addtonextcol --
⟨latexrelease | fltrace⟩                              locally counts:}%
⟨latexrelease | fltrace⟩   \fl@trace{col: \the \@colnum.
⟨latexrelease | fltrace⟩      top: \the \@topnum. bot: \the \@botnum.}%
⟨/trace⟩
⟨latexrelease | fltrace⟩  \endgroup
⟨*trace⟩
⟨latexrelease | fltrace⟩  \fl@trace{End of addtonextcol --
⟨latexrelease | fltrace⟩                             globally counts:}%
⟨latexrelease | fltrace⟩  \fl@trace{col: \the \@colnum.
⟨latexrelease | fltrace⟩        top: \the \@topnum. bot: \the \@botnum.}%
⟨/trace⟩
⟨latexrelease | fltrace⟩}%
⟨latexrelease | fltrace⟩\EndIncludeInRelease
```

\@addtodblcol Lots of changes.

```
⟨latexrelease | fltrace⟩\IncludeInRelease{2015/01/01}%
⟨latexrelease | fltrace⟩  {\@addtodblcol}{float order in 2-column}%
⟨*2ekernel | latexrelease | fltrace⟩
\def\@addtodblcol{%
  \begingroup
⟨*trace⟩
  \fl@trace{***Start addtodblcol}%
⟨/trace⟩
   \@insertfalse
   \@setfloattypecounts
   \@getfpsbit \tw@
⟨*trace⟩
   \fl@trace{fpstype \ifodd \@tempcnta OK \else not \fi dbltop:
                                                     \the \@fpstype}%
⟨/trace⟩
   \ifodd\@tempcnta
     \@flsetnum \@dbltopnum
     \ifnum \@dbltopnum>\z@
       \@tempswafalse
       \ifdim \@dbltoproom>\ht\@currbox
         \@tempswatrue
⟨*trace⟩
         \fl@trace{Space OK: \@dbltoproom =
                \the \@dbltoproom > \the \ht \@currbox
                                        (dbltoproom)}%
⟨/trace⟩
       \else
⟨*trace⟩
         \fl@trace{fpstype: \the \@fpstype (addtodblcol)}%
⟨/trace⟩
         \ifnum \@fpstype<\sixt@@n
⟨*trace⟩
           \fl@trace{BANG float ignoring \@dbltoproom}%
           \fl@trace{\@spaces \@dbltoproom = \the \@dbltoproom.
                           Ht float: \the \ht \@currbox-BANG}%
⟨/trace⟩
```

Need to check that there is room on the page, using the local value of \@textmin to make the necessary adjustment to \@dbltoproom.

```
          \advance \@dbltoproom \@textmin
⟨*trace⟩
          \fl@trace{Local value of texmin: \the\@textmin}%
          \fl@trace{\@spaces space on page = \the \@dbltoproom.
                          Ht float: \the \ht \@currbox-BANG}%
⟨/trace⟩
          \ifdim \@dbltoproom>\ht\@currbox
            \@tempswatrue
⟨*trace⟩
            \fl@trace{Space OK BANG: space on page =
                        \the \@dbltoproom > \the \ht \@currbox}%
          \else
            \fl@trace{fpstype: \the \@fpstype}%
            \fl@trace{Fail---no room dbltoproom-BANG?:}%
            \fl@trace{\@spaces space on page = \the \@dbltoproom.
                          Ht float: \the \ht \@currbox}%
⟨/trace⟩
          \fi
          \advance \@dbltoproom -\@textmin
⟨*trace⟩
        \else
          \fl@trace{fpstype: \the \@fpstype}%
          \fl@trace{Fail---no room dbltoproom-ORD?:}%
          \fl@trace{\@spaces \@dbltoproom = \the \@dbltoproom.
                          Ht float: \the \ht \@currbox}%
⟨/trace⟩
        \fi
      \fi
      \if@tempswa
          \@bitor \@currtype \@deferlist
⟨*trace⟩
          \fl@trace{(dbl)deferlist: \@deferlist: (before)}%
⟨/trace⟩
```

not in fixfloats?

```
         \@testwrongwidth\@currbox
          \if@test
⟨*trace⟩
             \fl@trace{type already on list: (dbl)defer}%
⟨/trace⟩
          \else
             \@tempdima -\ht\@currbox
             \advance\@tempdima
               -\ifx \@dbltoplist\@empty \dbltextfloatsep \else
                                         \dblfloatsep \fi
             \global \advance \@dbltoproom \@tempdima
             \global \advance \@colht \@tempdima
             \global \advance \@dbltopnum \m@ne
             \@cons \@dbltoplist \@currbox
⟨*trace⟩
             \fl@trace{dbltopnum (after) = \the \@dbltopnum}%
             \fl@trace{***Success: dbltop}%
⟨/trace⟩
```

```
              \@inserttrue
           \fi
       \fi
⟨*trace⟩
     \else
       \fl@trace{Fail: dbltopnum = \the \@dbltopnum: fpstype
                                                  \the \@fpstype=ORD?}%
       \ifnum \@fpstype<\sixt@@n
         \fl@trace{ERROR: !t float not successful (addtodblcol)}%
       \fi
⟨/trace⟩
     \fi
   \fi
   \if@insert
   \else
⟨*trace⟩
     \fl@trace{put on deferlist}%
⟨/trace⟩
     \@cons\@deferlist\@currbox
⟨*trace⟩
     \fl@trace{(dbl)deferlist: \@deferlist: (after)}%
⟨/trace⟩
   \fi
⟨*trace⟩
   \fl@trace{End of addtodblcol -- locally count:}%
   \fl@trace{ dbltop: \the \@dbltopnum.}%
⟨/trace⟩
  \endgroup
⟨*trace⟩
  \fl@trace{End of addtodblcol -- globally count:}%
  \fl@trace{dbltop: \the \@dbltopnum.}%
⟨/trace⟩
}%
⟨/2ekernel | latexrelease | fltrace⟩
⟨latexrelease | fltrace⟩\EndIncludeInRelease
⟨latexrelease | fltrace⟩\IncludeInRelease{0000/00/00}%
⟨latexrelease | fltrace⟩  {\@addtodblcol}{float order in 2-column}%
⟨latexrelease | fltrace⟩\def\@addtodblcol{%
⟨latexrelease | fltrace⟩  \begingroup
⟨*trace⟩
⟨latexrelease | fltrace⟩  \fl@trace{***Start addtodblcol}%
⟨/trace⟩
⟨latexrelease | fltrace⟩   \@insertfalse
⟨latexrelease | fltrace⟩   \@setfloattypecounts
⟨latexrelease | fltrace⟩   \@getfpsbit \tw@
⟨*trace⟩
⟨latexrelease | fltrace⟩   \fl@trace{fpstype \ifodd \@tempcnta OK
⟨latexrelease | fltrace⟩                \else not \fi dbltop: \the \@fpstype}%
⟨/trace⟩
⟨latexrelease | fltrace⟩   \ifodd\@tempcnta
⟨latexrelease | fltrace⟩     \@flsetnum \@dbltopnum
⟨latexrelease | fltrace⟩     \ifnum \@dbltopnum>\z@
⟨latexrelease | fltrace⟩       \@tempswafalse
⟨latexrelease | fltrace⟩       \ifdim \@dbltoproom>\ht\@currbox
```

```
⟨latexrelease | fltrace⟩       \@tempswatrue
⟨*trace⟩
⟨latexrelease | fltrace⟩       \fl@trace{Space OK: \@dbltoproom =
⟨latexrelease | fltrace⟩              \the \@dbltoproom > \the \ht \@currbox
⟨latexrelease | fltrace⟩                                         (dbltoproom)}%
⟨/trace⟩
⟨latexrelease | fltrace⟩     \else
⟨*trace⟩
⟨latexrelease | fltrace⟩    \fl@trace{fpstype: \the \@fpstype (addtodblcol)}%
⟨/trace⟩
⟨latexrelease | fltrace⟩       \ifnum \@fpstype<\sixt@@n
⟨*trace⟩
⟨latexrelease | fltrace⟩         \fl@trace{BANG float ignoring \@dbltoproom}%
⟨latexrelease | fltrace⟩         \fl@trace{\@spaces \@dbltoproom =
⟨latexrelease | fltrace⟩                  \the \@dbltoproom.
⟨latexrelease | fltrace⟩                  Ht float: \the \ht \@currbox-BANG}%
⟨/trace⟩
```

Need to check that there is room on the page, using the local value of `\@textmin` to make the necessary adjustment to `\@dbltoproom`.

```
⟨latexrelease | fltrace⟩         \advance \@dbltoproom \@textmin
⟨*trace⟩
⟨latexrelease | fltrace⟩     \fl@trace{Local value of texmin: \the\@textmin}%
⟨latexrelease | fltrace⟩     \fl@trace{\@spaces space on page =
⟨latexrelease | fltrace⟩                 \the \@dbltoproom.
⟨latexrelease | fltrace⟩                  Ht float: \the \ht \@currbox-BANG}%
⟨/trace⟩
⟨latexrelease | fltrace⟩         \ifdim \@dbltoproom>\ht\@currbox
⟨latexrelease | fltrace⟩           \@tempswatrue
⟨*trace⟩
⟨latexrelease | fltrace⟩       \fl@trace{Space OK BANG: space on page =
⟨latexrelease | fltrace⟩               \the\@dbltoproom > \the\ht\@currbox}%
⟨latexrelease | fltrace⟩         \else
⟨latexrelease | fltrace⟩       \fl@trace{fpstype: \the \@fpstype}%
⟨latexrelease | fltrace⟩       \fl@trace{Fail---no room dbltoproom-BANG?:}%
⟨latexrelease | fltrace⟩       \fl@trace{\@spaces space on page =
⟨latexrelease | fltrace⟩                    \the \@dbltoproom.
⟨latexrelease | fltrace⟩                     Ht float: \the \ht \@currbox}%
⟨/trace⟩
⟨latexrelease | fltrace⟩         \fi
⟨latexrelease | fltrace⟩         \advance \@dbltoproom -\@textmin
⟨*trace⟩
⟨latexrelease | fltrace⟩       \else
⟨latexrelease | fltrace⟩         \fl@trace{fpstype: \the \@fpstype}%
⟨latexrelease | fltrace⟩         \fl@trace{Fail---no room dbltoproom-ORD?:}%
⟨latexrelease | fltrace⟩         \fl@trace{\@spaces \@dbltoproom =
⟨latexrelease | fltrace⟩            \the \@dbltoproom.
⟨latexrelease | fltrace⟩            Ht float: \the \ht \@currbox}%
⟨/trace⟩
⟨latexrelease | fltrace⟩       \fi
⟨latexrelease | fltrace⟩     \fi
⟨latexrelease | fltrace⟩     \if@tempswa
⟨latexrelease | fltrace⟩         \@bitor \@currtype \@dbldeferlist
⟨*trace⟩
```

```
⟨latexrelease | fltrace⟩           \fl@trace{dbldeferlist:
⟨latexrelease | fltrace⟩                      \@dbldeferlist: (before)}%
⟨/trace⟩
⟨latexrelease | fltrace⟩           \if@test
⟨*trace⟩
⟨latexrelease | fltrace⟩             \fl@trace{type already on list: dbldefer}%
⟨/trace⟩
⟨latexrelease | fltrace⟩           \else
⟨latexrelease | fltrace⟩              \@tempdima -\ht\@currbox
⟨latexrelease | fltrace⟩              \advance\@tempdima
⟨latexrelease | fltrace⟩                -\ifx \@dbltoplist\@empty
⟨latexrelease | fltrace⟩                      \dbltextfloatsep
⟨latexrelease | fltrace⟩                 \else \dblfloatsep \fi
⟨latexrelease | fltrace⟩              \global \advance \@dbltoproom \@tempdima
⟨latexrelease | fltrace⟩              \global \advance \@colht \@tempdima
⟨latexrelease | fltrace⟩              \global \advance \@dbltopnum \m@ne
⟨latexrelease | fltrace⟩              \@cons \@dbltoplist \@currbox
⟨*trace⟩
⟨latexrelease | fltrace⟩              \fl@trace{dbltopnum (after) =
⟨latexrelease | fltrace⟩                                    \the \@dbltopnum}%
⟨latexrelease | fltrace⟩              \fl@trace{***Success: dbltop}%
⟨/trace⟩
⟨latexrelease | fltrace⟩              \@inserttrue
⟨latexrelease | fltrace⟩           \fi
⟨latexrelease | fltrace⟩       \fi
⟨*trace⟩
⟨latexrelease | fltrace⟩     \else
⟨latexrelease | fltrace⟩       \fl@trace{Fail: dbltopnum = \the \@dbltopnum:
⟨latexrelease | fltrace⟩                          fpstype \the \@fpstype=ORD?}%
⟨latexrelease | fltrace⟩       \ifnum \@fpstype<\sixt@@n
⟨latexrelease | fltrace⟩         \fl@trace{ERROR: !t float not successful
⟨latexrelease | fltrace⟩                                      (addtodblcol)}%
⟨latexrelease | fltrace⟩       \fi
⟨/trace⟩
⟨latexrelease | fltrace⟩     \fi
⟨latexrelease | fltrace⟩   \fi
⟨latexrelease | fltrace⟩   \if@insert
⟨latexrelease | fltrace⟩   \else
⟨*trace⟩
⟨latexrelease | fltrace⟩     \fl@trace{put on dbldeferlist}%
⟨/trace⟩
⟨latexrelease | fltrace⟩     \@cons\@dbldeferlist\@currbox
⟨*trace⟩
⟨latexrelease | fltrace⟩     \fl@trace{dbldeferlist: \@dbldeferlist: (after)}%
⟨/trace⟩
⟨latexrelease | fltrace⟩   \fi
⟨*trace⟩
⟨latexrelease | fltrace⟩   \fl@trace{End of addtodblcol -- locally count:}%
⟨latexrelease | fltrace⟩   \fl@trace{ dbltop: \the \@dbltopnum.}%
⟨/trace⟩
⟨latexrelease | fltrace⟩  \endgroup
⟨*trace⟩
⟨latexrelease | fltrace⟩  \fl@trace{End of addtodblcol -- globally count:}%
⟨latexrelease | fltrace⟩  \fl@trace{dbltop: \the \@dbltopnum.}%
```

```
⟨/trace⟩
⟨latexrelease | fltrace⟩}%
⟨latexrelease | fltrace⟩\EndIncludeInRelease
```

\@addmarginpar

```
⟨*2ekernel⟩
\def\@addmarginpar{\@next\@marbox\@currlist{\@cons\@freelist\@marbox
    \@cons\@freelist\@currbox}\@latexbug\@tempcnta\@ne
    \if@twocolumn
        \if@firstcolumn \@tempcnta\m@ne \fi
    \else
      \if@mparswitch
         \ifodd\c@page \else\@tempcnta\m@ne \fi
      \fi
      \if@reversemargin \@tempcnta -\@tempcnta \fi
    \fi
    \ifnum\@tempcnta <\z@  \global\setbox\@marbox\box\@currbox \fi
    \@tempdima\@mparbottom
    \advance\@tempdima -\@pageht
    \advance\@tempdima\ht\@marbox
    \ifdim\@tempdima >\z@
      \@latex@warning@no@line {Marginpar on page \thepage\space moved}%
    \else
      \@tempdima\z@
    \fi
    \global\@mparbottom\@pageht
    \global\advance\@mparbottom\@tempdima
    \global\advance\@mparbottom\dp\@marbox
    \global\advance\@mparbottom\marginparpush
    \advance\@tempdima -\ht\@marbox
```

Putting box movement inside the ‘marbox’:

```
    \global\setbox \@marbox
                   \vbox {\vskip \@tempdima
                          \box \@marbox}%
    \global \ht\@marbox \z@
    \global \dp\@marbox \z@
```

Sticking (rather than gluing:-) the ‘marbox’ to the line above, changed vskip to kern:

```
    \kern -\@pagedp
    \nointerlineskip
    \hb@xt@\columnwidth
      {\ifnum \@tempcnta >\z@
          \hskip\columnwidth \hskip\marginparsep
       \else
          \hskip -\marginparsep \hskip -\marginparwidth
       \fi
       \box\@marbox \hss}%
```

For this reason the following code can vanish:

```
\nobreak             %% No longer needed.  CAR92/12
\vskip -\@tempdima   %% No longer needed.  CAR92/12
```

```
    \nointerlineskip
    \hbox{\vrule \@height\z@ \@width\z@ \@depth\@pagedp}}
```

75.1.1 Kludgeins

This part of the file is part of the implementation of the following two new commands for LaTeX2e.

```
\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 execpt 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 TeX do a lot of the necessary work. All we need to put into it is the amount by which the pagegoal should be changed.

```
\newinsert \@kludgeins
\global\dimen\@kludgeins \maxdimen
\global\count\@kludgeins 1000
```

`\enlargethispage`
`\enlargethispage*` The user command.

```
\gdef \enlargethispage {%
   \@ifstar
     {%
⟨*trace⟩
      \fl@trace{Enlarging page height * }%
⟨/trace⟩
      \@enlargepage{\hbox{\kern\p@}}}%
     {%
⟨*trace⟩
      \fl@trace{Enlarging page height exactly---}%
⟨/trace⟩
      \@enlargepage\@empty}%
}
```

`\@enlargepage` This actually inserts the insert, after checking for extreme values of the change.

```
\gdef\@enlargepage#1#2{%
⟨*trace⟩
   \fl@trace{\@spaces\@spaces by #2}%
⟨/trace⟩
   \@tempskipa#2\relax
   \ifdim \@tempskipa>.5\maxdimen
```

```
     \@latexerr{Suggested\space extra\space height\space
                (\the\@tempskipa)\space dangerously\space
                large}\@eha
   \else
     \ifdim \vsize<.5\maxdimen
⟨*trace⟩
       \fl@trace {Kludgeins added--pagegoal before: \the\pagegoal}%
⟨/trace⟩
       \@bsphack
         \insert\@kludgeins{#1\vskip-\@tempskipa}%
       \@esphack
```

This next bit is for tracing only:

```
⟨*trace⟩
       \ifvmode \par
         \fl@trace {Kludgeins added--pagegoal after: \the \pagegoal}%
       \fi
⟨/trace⟩
     \else
       \@latexerr{Page\space height\space already\space
                  too\space large}\@eha
     \fi
   \fi
}
⟨/2ekernel⟩
```

75.1.2 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 LaTeX2e.

```
\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 followinhg 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
\tracefloatsoff
\tracefloats
\fl@traceval
\tracefloatvals
\fl@tracemessage

Set-up tracing for floats independent of other tracing as it produces mega-output. Default is no tracing.

```
⟨*fltrace⟩
\def \fl@tracemessage #1{{\let\@elt\@empty\typeout{LaTeX2e: #1}}}
\def \tracefloats{\let \fl@trace \fl@tracemessage}
\def \tracefloatsoff {\let \fl@trace \@gobble}
\tracefloatsoff
\def \fl@traceval #1{\fl@trace{\string #1 = \the #1}}
\IncludeInRelease{2015/01/01}{\tracefloatvals}%
                          {trace float vals}%
\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.[11]

```
 \begingroup
  \@dblfloatplacement
  \@floatplacement
  \fl@trace{***Float placement parameters:}%
  \fl@traceval\@colnum
  \fl@traceval\@colroom
  \fl@traceval\@topnum
  \fl@traceval\@toproom
  \fl@traceval\@botnum
  \fl@traceval\@botroom
  \fl@traceval\@fpmin
  \fl@trace{\string\textfraction = \textfraction}%
  \fl@traceval\@dbltopnum
  \fl@traceval\@dbltoproom
  \fl@trace{\string\textfraction = \textfraction}%
  \fl@trace{toplist: \@toplist}%
  \fl@trace{botlist: \@botlist}%
  \fl@trace{midlist: \@midlist}%
  \fl@trace{deferlist: \@deferlist}%
  \fl@trace{dbltoplist: \@dbltoplist}%
%FMi  \fl@trace{dbldeferlist: \@dbldeferlist}%
 \endgroup
}
\EndIncludeInRelease
\IncludeInRelease{0000/00/00}{\tracefloatvals}%
                          {trace float vals}%
\def \tracefloatvals{%
 \begingroup
  \@dblfloatplacement
  \@floatplacement
  \fl@trace{***Float placement parameters:}%
  \fl@traceval\@colnum
  \fl@traceval\@colroom
  \fl@traceval\@topnum
```

[11] This is a somewhat questionable design.

```
  \fl@traceval\@toproom
  \fl@traceval\@botnum
  \fl@traceval\@botroom
  \fl@traceval\@fpmin
  \fl@trace{\string\textfraction = \textfraction}%
  \fl@traceval\@dbltopnum
  \fl@traceval\@dbltoproom
  \fl@trace{\string\textfraction = \textfraction}%
  \fl@trace{toplist: \@toplist}%
  \fl@trace{botlist: \@botlist}%
  \fl@trace{midlist: \@midlist}%
  \fl@trace{deferlist: \@deferlist}%
  \fl@trace{dbltoplist: \@dbltoplist}%
% next line only in old releases
  \fl@trace{dbldeferlist: \@dbldeferlist}%
 \endgroup
}
\EndIncludeInRelease
```

We need to make sure that fltrace comes before flafter to make the tracing work.

```
\@ifpackageloaded{flafter}
 {\PackageWarningNoLine
      {fltrace}{Load 'fltrace' before 'flafter'\MessageBreak
                Attempting to recover by reloading 'flafter'}%
```

Hide the fact that flafter was already loaded and then request it anew.

```
    \expandafter\let\csname ver@flafter.sty\endcsname\relax
    \def\reserved@a#1{%
      \expandafter\let\csname\string#1+flafter+IIR\endcsname\relax}%
    \reserved@a\@addtocurcol
    \reserved@a\@addtonextcol
    \RequirePackage{flafter}}{}
⟨/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.

```
⟨*flafter⟩
\providecommand\fl@trace[1]{}
⟨/flafter⟩
```

\suppressfloats
\@flstop

Float suppression commands: these set the relevant counter globally to zero. Thus they are overridden for a particular float by an ! specifier.

```
⟨*2ekernel⟩
\def \suppressfloats {%
   \@ifnextchar [%
     \@flstop
    {\global \@colnum \z@}%
}
```

Maybe this should be a loop over #1?

```
\def \@flstop [#1]{%
    \if t#1%
     \global \@topnum \z@
```

```
  \fi
  \if b#1%
    \global \@botnum \z@
  \fi
}
```

Manipulation of float placement and type; both their strings and the corresponding count registers.

`\@fpstype` `\@reqcolroom` `\@textfloatsheight`

First a new count register to go with `\@currtype`.

Then a new skip register, for information needed to remove the `\@maxsep` conservatism: it is possible that this could use a temporary register.

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.

```
\newcount \@fpstype
\newdimen \@reqcolroom
\newdimen \@textfloatsheight
⟨/2ekernel⟩
```

`\@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@\@captype \endcsname}%
   \edef \reserved@a {\the\@temptokena}%
   \@onelevel@sanitize \reserved@a
   \edef \@fps {\@fps\reserved@a}%
}
```

```
⟨*2ekernel | fltrace⟩
\def \@fpsadddefault {%
⟨*trace⟩
  \fl@trace{fps changed from: \@fps}%
⟨/trace⟩
  \edef \@fps {\@fps\csname fps@\@captype \endcsname}%
  \@latex@warning {%
    No positions in optional float specifier.\MessageBreak
    Default added (so using `\@fps')}%
}
```

`\@setfloattypecounts`

Sets counters `\@fpstype` and `\@currtype`.

BANG == bit4 of `\count\@currbox` = 0.

```
\def \@setfloattypecounts {%
  \@currtype \count\@currbox
  \@fpstype \count\@currbox
  \divide\@currtype\@xxxii \multiply\@currtype\@xxxii
  \advance \@fpstype -\@currtype
⟨*trace⟩
  \fl@trace{(mod 32) fpstype: \the \@fpstype}%
  \fl@trace{(mult of 32) currtype: \the \@currtype}%
% Tracing only: but some should be changed into real errors/warnings?
  \ifnum \@fpstype<\sixt@@n
    \ifnum \@fpstype=\z@
       \fl@trace{ERROR: no PLACEMENT, fpstype = \the \@fpstype = 0?}%
    \fi
    \ifnum \@fpstype=\@ne
      \fl@trace{WARNING: only h, fpstype = \the \@fpstype = 1?}%
    \fi
    \fl@trace{BANG float}%
  \else
    \ifnum \@fpstype=\sixt@@n
      \fl@trace{ERROR: no PLACEMENT, fpstype = \the \@fpstype = 16?}%
    \fi
    \ifnum \@fpstype=17
      \fl@trace{WARNING: only h, fpstype = \the \@fpstype = 17?}%
    \fi
    \fl@trace{ORD float}%
  \fi
⟨/trace⟩
}
⟨/2ekernel | fltrace⟩
```

Macros for getting, testing and setting bits of the fps.

`\@getfpsbit` Sets `\@tempcnta` to required bit of `\count\@currbox`.

```
⟨*2ekernel⟩
\def \@getfpsbit {%
   \@boxfpsbit \@currbox
}
```

`\@boxfpsbit` Used above.

```
\def \@boxfpsbit #1#2{%
   \@tempcnta \count#1%
   \divide \@tempcnta #2\relax
}
```

`\@testfp` New definition of the float page test.

```
\def \@testfp #1{%
   \@boxfpsbit #18\relax % Really ‘#1 8’ for human readers!
   \ifodd \@tempcnta
   \else
     \@testtrue
   \fi
}
```

`\@setfpsbit` Sets required bit of `\@tempcnta` (to 1).

```
\def \@setfpsbit #1{%
   \@tempcntb \@tempcnta
   \divide \@tempcntb #1\relax
   \ifodd \@tempcntb
   \else
     \advance \@tempcnta #1\relax
   \fi
}
⟨/2ekernel⟩
```

\@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.

```
⟨*2ekernel | fltrace⟩
\def \@resethfps {%
   \let\reserved@a\@empty
   \ifnum \@fpstype=\@ne
      \def \reserved@a {!}%
      \@fpstype 17
   \fi
   \ifnum \@fpstype=17
     \global \advance \count\@currbox \tw@
     \@latex@warning@no@line {%
       `\reserved@a h' float specifier changed to `\reserved@a ht'}%
⟨*trace⟩
     \fl@trace{%
        `t' added to `\reserved@a h'- new Count: \the \count\@currbox}%
⟨/trace⟩
   \fi
}
```

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.

```
\def \@flsetnum #1{%
⟨*trace⟩
   \fl@trace{fpstype: \the \@fpstype (flsetnum \string#1)}%
⟨/trace⟩
   \ifnum \@fpstype<\sixt@@n
     \ifnum #1=\z@
⟨*trace⟩
       \fl@trace{BANG float resetting \string#1 to 1}%
⟨/trace⟩
       #1\@ne
     \fi
   \fi
⟨*trace⟩
   \fl@trace{#1 (before) = \the #1}%
⟨/trace⟩
}
```

\@flsettextmin This ignores \textfraction space restriction in case BANG.

```
\def \@flsettextmin {%
⟨*trace⟩
   \fl@trace{fpstype: \the \@fpstype (flsettextmin)}%
⟨/trace⟩
   \ifnum \@fpstype<\sixt@@n
⟨*trace⟩
     \fl@trace{BANG ignoring textmin}%
⟨/trace⟩
     \@textmin \z@
   \else
     \@textmin \textfraction\@colht
⟨*trace⟩
     \fl@trace{ORD textmin = \the \@textmin}%
⟨/trace⟩
   \fi
}
```

\@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.

```
\def \@flcheckspace #1#2{%
   \advance \@reqcolroom
     \ifx #2\@empty \textfloatsep \else \floatsep \fi
⟨*trace⟩
   \fl@trace{colroom = \the \@colroom
                                   (flcheckspace \string#1 \string#2)}%
   \fl@trace{reqcolroom = \the \@reqcolroom
                                   (flcheckspace \string#1 \string#2)}%
⟨/trace⟩
   \ifdim \@colroom>\@reqcolroom
     \ifdim #1>\ht\@currbox
       \@tempswatrue
⟨*trace⟩
       \fl@trace{Space OK: #1 = \the #1 > \the \ht \@currbox
                                   (flcheckspace \string#1 \string#2)}%
⟨/trace⟩
     \else
⟨*trace⟩
       \fl@trace{fpstype: \the \@fpstype
                                   (flcheckspace \string#1 \string#2)}%
⟨/trace⟩
       \ifnum \@fpstype<\sixt@@n
⟨*trace⟩
         \fl@trace{BANG float ignoring #1
                                   (flcheckspace \string#1 \string#2):}%
         \fl@trace{\@spaces #1 = \the #1.  Ht float: \the \ht \@currbox
                                                          BANG}%
⟨/trace⟩
         \@tempswatrue
⟨*trace⟩
```

```
       \else
         \fl@trace{Fail---no room (flcheckspace \string#1 \string#2)
                       (fpstype \the \@fpstype=ORD?):}%
         \fl@trace{\@spaces #1 = \the #1.  Ht float: \the \ht \@currbox
                                                          ORD?}%
⟨/trace⟩
       \fi
     \fi
⟨*trace⟩
   \else
     \fl@trace{Fail---no room at 2nd test of colroom
                   (flcheckspace \string#1 \string#2)}%
⟨/trace⟩
   \fi
}
⟨/2ekernel | fltrace⟩
```

\@flupdates This updates everything when a float is placed.

```
⟨*2ekernel⟩
\def \@flupdates #1#2#3{%
   \global \advance #1\m@ne
   \global \advance \@colnum \m@ne
   \@tempdima -\ht\@currbox
   \advance \@tempdima
     -\ifx #3\@empty \textfloatsep \else \floatsep \fi
   \global \advance #2\@tempdima
   \global \advance \@colroom \@tempdima
   \@cons #3\@currbox
}
⟨/2ekernel⟩
```

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 `\dbltexfloatsep`'. 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 footmotes 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.

```
⟨latexrelease | fltrace⟩\IncludeInRelease{2017/01/01}%
⟨latexrelease | fltrace⟩   {\@makefcolumn}{negative height floats}%
⟨*2ekernel | fltrace | latexrelease⟩
\def\@makefcolumn #1{%
  \begingroup
```

```
    \@fpmin -\maxdimen
    \let \@testfp \@gobble
    \@tryfcolumn #1%
  \endgroup
⟨*trace⟩
  \if@fcolmade
    \fl@trace{PAGE: in \string\clearpage
                              \if@twocolumn ---twocolumn\fi---}%
    \fl@trace{----- float column/page completed from \string#1}%
  \fi
⟨/trace⟩
}
⟨latexrelease | fltrace⟩\EndIncludeInRelease
⟨latexrelease | fltrace⟩\IncludeInRelease{0000/00/00}%
⟨latexrelease | fltrace⟩  {\@makefcolumn}{negative height floats}%
⟨latexrelease | fltrace⟩\def\@makefcolumn #1{%
⟨latexrelease | fltrace⟩  \begingroup
⟨latexrelease | fltrace⟩    \@fpmin \z@
⟨latexrelease | fltrace⟩    \let \@testfp \@gobble
⟨latexrelease | fltrace⟩    \@tryfcolumn #1%
⟨latexrelease | fltrace⟩  \endgroup
⟨*trace⟩
⟨latexrelease | fltrace⟩  \if@fcolmade
⟨latexrelease | fltrace⟩    \fl@trace{PAGE: in \string\clearpage
⟨latexrelease | fltrace⟩                        \if@twocolumn ---twocolumn\fi---}%
⟨latexrelease | fltrace⟩    \fl@trace{----- float column/page completed
⟨latexrelease | fltrace⟩                        from \string#1}%
⟨latexrelease | fltrace⟩  \fi
⟨/trace⟩
⟨latexrelease | fltrace⟩}
⟨latexrelease | fltrace⟩\EndIncludeInRelease
⟨/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.

```
⟨latexrelease | fltrace⟩\IncludeInRelease{2015/01/01}%
⟨latexrelease | fltrace⟩  {\@outputdblcol}{2 column marks}%
⟨*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.

```
\def\@outputdblcol{%
  \if@firstcolumn
    \global\@firstcolumnfalse
```

Save the left column

```
    \global\setbox\@leftcolumn\copy\@outputbox
⟨fltrace⟩    \fl@trace{PAGE: first column boxed}%
```

Remember the marks from the first column

```
    \splitmaxdepth\maxdimen
    \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.

```
    \setbox\@outputbox\vbox{\unvbox\@outputbox\unskip}%
    \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.

```
   \toks@\expandafter{\topmark}%
   \xdef\@firstcoltopmark{\the\toks@}%
   \toks@\expandafter{\splitfirstmark}%
   \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.)

```
   \ifx\@firstcolfirstmark\@empty
     \global\let\@setmarks\relax
   \else
     \gdef\@setmarks{%
       \let\firstmark\@firstcolfirstmark
       \let\topmark\@firstcoltopmark}%
   \fi
```

End of change

```
 \else
   \global\@firstcolumntrue
   \setbox\@outputbox\vbox{%
    \hb@xt@\textwidth{%
       \hb@xt@\columnwidth{\box\@leftcolumn \hss}%
       \hfil
```

The color of the \vrule should be \normalcolor as to not inherit the color from the column.

```
       {\normalcolor\vrule \@width\columnseprule}%
       \hfil
      \hb@xt@\columnwidth{\box\@outputbox \hss}}}%
⟨fltrace⟩    \fl@trace{PAGE: second column also boxed}%
 \@combinedblfloats
```

Override current first and top with those of first column if necessary

```
   \@setmarks
```

End of change

```
   \@outputpage
⟨fltrace⟩    \fl@trace{PAGE: two column page completed}%
   \begingroup
     \@dblfloatplacement
     \@startdblcolumn
     \@whilesw\if@fcolmade \fi{\@outputpage
⟨fltrace⟩      \fl@trace{PAGE: double float page completed}%
    \@startdblcolumn}%
   \endgroup
 \fi}%
```

```
⟨latexrelease | fltrace⟩\EndIncludeInRelease
⟨latexrelease | fltrace⟩\IncludeInRelease{0000/00/00}%
⟨latexrelease | fltrace⟩  {\@outputdblcol}{2 column marks}%
⟨latexrelease | fltrace⟩\def\@outputdblcol{%
⟨latexrelease | fltrace⟩  \if@firstcolumn
⟨latexrelease | fltrace⟩    \global \@firstcolumnfalse
⟨latexrelease | fltrace⟩    \global \setbox\@leftcolumn \box\@outputbox
⟨*trace⟩
⟨latexrelease | fltrace⟩    \fl@trace{PAGE: first column boxed}%
⟨/trace⟩
⟨latexrelease | fltrace⟩  \else
⟨latexrelease | fltrace⟩    \global \@firstcolumntrue
⟨latexrelease | fltrace⟩    \setbox\@outputbox \vbox {%
⟨latexrelease | fltrace⟩                         \hb@xt@\textwidth {%
⟨latexrelease | fltrace⟩                           \hb@xt@\columnwidth {%
⟨latexrelease | fltrace⟩                             \box\@leftcolumn \hss}%
⟨latexrelease | fltrace⟩                           \hfil
⟨latexrelease | fltrace⟩                           {\normalcolor\vrule
⟨latexrelease | fltrace⟩                               \@width\columnseprule}%
⟨latexrelease | fltrace⟩                           \hfil
⟨latexrelease | fltrace⟩                           \hb@xt@\columnwidth {%
⟨latexrelease | fltrace⟩                             \box\@outputbox \hss}%
⟨latexrelease | fltrace⟩                                             }%
⟨latexrelease | fltrace⟩                              }%
⟨*trace⟩
⟨latexrelease | fltrace⟩    \fl@trace{PAGE: second column also boxed}%
⟨/trace⟩
⟨latexrelease | fltrace⟩    \@combinedblfloats
⟨latexrelease | fltrace⟩    \@outputpage
⟨*trace⟩
⟨latexrelease | fltrace⟩    \fl@trace{PAGE: two column page completed}%
⟨/trace⟩
⟨latexrelease | fltrace⟩    \begingroup
⟨latexrelease | fltrace⟩      \@dblfloatplacement
⟨latexrelease | fltrace⟩      \@startdblcolumn
```

This loop could be replaced by an \expandafter tail recursion in \@startdblcolumn.

```
⟨latexrelease | fltrace⟩      \@whilesw\if@fcolmade \fi
⟨latexrelease | fltrace⟩        {\@outputpage
⟨*trace⟩
⟨latexrelease | fltrace⟩      \fl@trace{PAGE: double float page completed}%
⟨/trace⟩
⟨latexrelease | fltrace⟩         \@startdblcolumn}%
⟨latexrelease | fltrace⟩    \endgroup
⟨latexrelease | fltrace⟩  \fi
⟨latexrelease | fltrace⟩}%
⟨latexrelease | fltrace⟩\EndIncludeInRelease
⟨/2ekernel | fltrace | latexrelease⟩
```

75.1.3 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.

```
⟨*2ekernel⟩
\newcount\c@topnumber
\setcounter{topnumber}{2}
```

`\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.

```
\newcommand\topfraction{.7}
```

`\c@bottomnumber` This counter holds the maximum number of floats that can appear at the bottom of a text page or column.

```
\newcount\c@bottomnumber
\setcounter{bottomnumber}{1}
```

`\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.

```
\newcommand\bottomfraction{.3}
```

`\c@totalnumber` This counter holds the maximum number of floats that can appear on any text page or column.

```
\newcount\c@totalnumber
\setcounter{totalnumber}{3}
```

`\textfraction` This macro holds the minimum proportion (as a decimal number) of a text page or column that must be occupied by text.

```
\newcommand\textfraction{.2}
```

`\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.

```
\newcommand\floatpagefraction{.5}
```

`\c@dbltopnumber` This counter holds the maximum number of double-column floats that can appear on the top of a two-column text page.

```
\newcount\c@dbltopnumber
\setcounter{dbltopnumber}{2}
```

`\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.

```
\newcommand\dbltopfraction{.7}
```

`\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.

```
\newcommand\dblfloatpagefraction{.5}
```

Floats on a text page

`\floatsep`
`\textfloatsep`
`\intextsep`

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.

```
\newskip\floatsep
\newskip\textfloatsep
\newskip\intextsep
\setlength\floatsep    {12\p@ \@plus 2\p@ \@minus 2\p@}
\setlength\textfloatsep{20\p@ \@plus 2\p@ \@minus 4\p@}
\setlength\intextsep   {12\p@ \@plus 2\p@ \@minus 2\p@}
```

`\dblfloatsep`
`\dbltextfloatsep`

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.

```
\newskip\dblfloatsep
\newskip\dbltextfloatsep
\setlength\dblfloatsep    {12\p@ \@plus 2\p@ \@minus 2\p@}
\setlength\dbltextfloatsep{20\p@ \@plus 2\p@ \@minus 4\p@}
```

Floats on their own page or column

`\@fptop`
`\@fpsep`
`\@fpbot`

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.

At the top of the page `\@fptop` is inserted; typically this supplies some stretchable whitespace. At the bottom of the page `\@fpbot` ais 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.

```
\newskip\@fptop
\newskip\@fpsep
\newskip\@fpbot
\setlength\@fptop{0\p@ \@plus 1fil}
\setlength\@fpsep{8\p@ \@plus 2fil}
\setlength\@fpbot{0\p@ \@plus 1fil}
```

`\@dblfptop`
`\@dblfpsep`
`\@dblfpbot`

Double-column 'float pages' in two-column mode use similar parameters.

```
\newskip\@dblfptop
\newskip\@dblfpsep
\newskip\@dblfpbot
\setlength\@dblfptop{0\p@ \@plus 1fil}
\setlength\@dblfpsep{8\p@ \@plus 2fil}
\setlength\@dblfpbot{0\p@ \@plus 1fil}
```

`\topfigrule`
`\botfigrule`
`\dblfigrule`

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.

```
\let\topfigrule=\relax
\let\botfigrule=\relax
\let\dblfigrule=\relax
⟨/2ekernel⟩
```

File N
ltclass.dtx

76 Introduction

This file implements the following declarations, which replace `\documentstyle` in LaTeX 2ε 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 LaTeX 2ε and LaTeX2.09 is that LaTeX 2ε 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.

77 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 LaTeX2.09 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 LaTeX2.09 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 `nopremable` 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.

77.1 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.

78 Class and Package interface

78.1 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`.

78.2 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`.

78.3 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}
\RequirePackage[baz]{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` `\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.

\@ifpackageloaded
\@ifclassloaded
To find out if a package has already been loaded, use
\@ifpackageloaded{⟨*package*⟩}{⟨*true*⟩}{⟨*false*⟩}.

\@ifpackagelater
\@ifclasslater
To find out if a package has already been loaded with a version equal to or more recent than ⟨*version*⟩, use
\@ifpackagelater{⟨*package*⟩}{⟨*version*⟩}{⟨*true*⟩}{⟨*false*⟩}.

\@ifpackagewith
\@ifclasswith
To find out if a package has already been loaded with at least the options ⟨*options*⟩, use \@ifpackagewith{⟨*package*⟩}{⟨*options*⟩}{⟨*true*⟩}{⟨*false*⟩}.

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).

78.4 Declaring new options

Options for classes and packages are built using the same macros.

\DeclareOption
\DeclareOption*
To define a builtin option, use \DeclareOption{⟨*name*⟩}{⟨*code*⟩}.

To define the default action to perform for local options which have not been declared, use \DeclareOption*{⟨*code*⟩}.

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.

78.5 Safe Input Macros

`\InputIfFileExists` `\InputIfFileExists{`⟨*file*⟩`}{`⟨*then*⟩`}{`⟨*else*⟩`}`
Inputs ⟨*file*⟩ if it exists. Immediately before the input, ⟨*then*⟩ is executed. Otherwise ⟨*else*⟩ is executed.

`\IfFileExists` As above, but does not input the file.

One thing you might like to put in the ⟨*else*⟩ clause is

`\@missingfileerror` 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,

`\input` This has been redefined from the LaTeX2.09 definition, in terms of the new commands `\InputIfFileExists` and `\@missingfileerror`.

`\listfiles` 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 `\ProvidesPackage` are listed alongside the file name. So files in standard (and other non-standard) distributions can put informative strings in this argument.

79 Implementation

```
⟨*2ekernel⟩
```

`\if@compatibility` The flag for compatibility mode.

```
\newif\if@compatibility
```

`\@documentclasshook` The hook called after the first `\documentclass` command. By default this checks to see if `\@normalsize` is undefined, and if so, sets it to `\normalsize`.

```
\def\@documentclasshook{%
   \ifx\@normalsize\@undefined
      \let\@normalsize\normalsize
   \fi
}
```

`\@declaredoptions` This list is automatically built by `\DeclareOption`. It is the list of options (separated by commas) declared in the class or package file and it defines the order in which the the corresponding `\ds@`⟨*option*⟩ commands are executed. All local ⟨*option*⟩s which are not declared will be processed in the order defined by the optional argument of `\documentclass` or `\usepackage`.

```
\let\@declaredoptions\@empty
```

`\@classoptionslist` List of options of the main class.

```
\let\@classoptionslist\relax
\@onlypreamble\@classoptionslist
```

\@unusedoptionlist

List of options of the main class that haven't been declared or loaded as class option files.

```
\let\@unusedoptionlist\@empty
\@onlypreamble\@unusedoptionlist
```

\CurrentOption

Name of current package or option.

```
\let\CurrentOption\@empty
```

\@currname

Name of current package or option.

```
\let\@currname\@empty
```

\@currext

The current file extension.

```
\global\let\@currext=\@empty
```

\@clsextension
\@pkgextension

The two possible values of `\@currext`.

```
\def\@clsextension{cls}
\def\@pkgextension{sty}
\@onlypreamble\@clsextension
\@onlypreamble\@pkgextension
```

\@pushfilename
\@popfilename
\@currnamestack

Commands to push and pop the file name and extension.
#1 current name.
#2 current extension.
#3 current catcode of @.
#4 Rest of the stack.

```
\def\@pushfilename{%
  \xdef\@currnamestack{%
    {\@currname}%
    {\@currext}%
    {\the\catcode`\@}%
    \@currnamestack}}
\@onlypreamble\@pushfilename
```

```
\def\@popfilename{\expandafter\@p@pfilename\@currnamestack\@nil}
\@onlypreamble\@popfilename
```

```
\def\@p@pfilename#1#2#3#4\@nil{%
  \gdef\@currname{#1}%
  \gdef\@currext{#2}%
  \catcode`\@#3\relax
  \gdef\@currnamestack{#4}}
\@onlypreamble\@p@pfilename
```

```
\gdef\@currnamestack{}
\@onlypreamble\@currnamestack
```

\@ptionlist

Returns the option list of the file.

```
\def\@ptionlist#1{%
  \@ifundefined{opt@#1}\@empty{\csname opt@#1\endcsname}}
\@onlypreamble\@ptionlist
```

\@ifpackageloaded
\@ifclassloaded

`\@ifpackageloaded{`⟨*name*⟩`}` Checks to see whether a file has been loaded.

```
\def\@ifpackageloaded{\@ifl@aded\@pkgextension}
\def\@ifclassloaded{\@ifl@aded\@clsextension}
\@onlypreamble\@ifpackageloaded
\@onlypreamble\@ifclassloaded
```

```
\def\@ifl@aded#1#2{%
  \expandafter\ifx\csname ver@#2.#1\endcsname\relax
    \expandafter\@secondoftwo
  \else
    \expandafter\@firstoftwo
  \fi}
\@onlypreamble\@ifl@aded
```

\@ifpackagelater
\@ifclasslater

\@ifpackagelater{⟨*name*⟩}{YYYY/MM/DD} Checks that the package loaded is more recent than the given date.

```
\def\@ifpackagelater{\@ifl@ter\@pkgextension}
\def\@ifclasslater{\@ifl@ter\@clsextension}
\@onlypreamble\@ifpackagelater
\@onlypreamble\@ifclasslater
```

```
\def\@ifl@ter#1#2{%
  \expandafter\@ifl@t@r
    \csname ver@#2.#1\endcsname}
\@onlypreamble\@ifl@ter
⟨/2ekernel⟩
```

This internal macro is also used in \NeedsTeXFormat.

```
⟨latexrelease⟩\IncludeInRelease{2018/04/01}%
⟨latexrelease⟩                 {\@ifl@t@r}{Guard against bad input}%
⟨*2ekernel | latexrelease⟩
\def\@ifl@t@r#1#2{%
  \ifnum\expandafter\@parse@version@#1//00\@nil<%
        \expandafter\@parse@version@#2//00\@nil
    \expandafter\@secondoftwo
  \else
    \expandafter\@firstoftwo
  \fi}
\def\@parse@version@#1{\@parse@version0#1}
⟨/2ekernel | latexrelease⟩
⟨latexrelease⟩\EndIncludeInRelease
⟨latexrelease⟩\IncludeInRelease{0000/00/00}%
⟨latexrelease⟩                 {\@ifl@t@r}{Guard against bad input}%
⟨latexrelease⟩\def\@ifl@t@r#1#2{%
⟨latexrelease⟩  \ifnum\expandafter\@parse@version#1//00\@nil<%
⟨latexrelease⟩        \expandafter\@parse@version#2//00\@nil
⟨latexrelease⟩    \expandafter\@secondoftwo
⟨latexrelease⟩  \else
⟨latexrelease⟩    \expandafter\@firstoftwo
⟨latexrelease⟩  \fi}
⟨latexrelease⟩\let\@parse@version@\@undefined
⟨latexrelease⟩\EndIncludeInRelease
⟨*2ekernel⟩
```

```
\@onlypreamble\@ifl@t@r
```

```
⟨/2ekernel⟩
⟨*2ekernel | latexreleasefirst⟩
\def\@parse@version#1/#2/#3#4#5\@nil{%
\@parse@version@dash#1-#2-#3#4\@nil
}
```

The `\if` test here ensures that an argument with no `/` or `-` produces 0 (actually 00).

```
\def\@parse@version@dash#1-#2-#3#4#5\@nil{%
  \if\relax#2\relax\else#1\fi#2#3#4 }
⟨/2ekernel | latexreleasefirst⟩
⟨*2ekernel⟩
```

`\@ifpackagewith`
`\@ifclasswith`

`\@ifpackagewith{`⟨*name*⟩`}{`⟨*option-list*⟩`}` Checks that ⟨*option-list*⟩ is a subset of the options **with** which ⟨*name*⟩ was loaded.

```
\def\@ifpackagewith{\@if@ptions\@pkgextension}
\def\@ifclasswith{\@if@ptions\@clsextension}
\@onlypreamble\@ifpackagewith
\@onlypreamble\@ifclasswith
```

```
\def\@if@ptions#1#2{%
  \@expandtwoargs\@if@pti@ns{\@ptionlist{#2.#1}}}
\@onlypreamble\@if@ptions
```

Probably shouldn't use `\CurrentOption` here… (changed to `\reserved@b`.)

```
⟨/2ekernel⟩
⟨latexrelease⟩\IncludeInRelease{2017/01/01}%
⟨latexrelease⟩                 {\@if@pti@ns}{Spaces in option clash check}%
⟨*2ekernel | latexrelease⟩
\def\@if@pti@ns#1#2{%
 \let\reserved@a\@firstoftwo
```

```
 \edef\reserved@b{\zap@space#2 \@empty}%
 \@for\reserved@b:=\reserved@b\do{%
   \ifx\reserved@b\@empty
   \else
     \expandafter\in@\expandafter{\expandafter,\reserved@b,}{,#1,}%
     \ifin@
     \else
       \let\reserved@a\@secondoftwo
     \fi
   \fi
 }%
 \reserved@a}
⟨/2ekernel | latexrelease⟩
⟨latexrelease⟩\EndIncludeInRelease
⟨latexrelease⟩\IncludeInRelease{0000/00/00}%
⟨latexrelease⟩                 {\@if@pti@ns}{Spaces in option clash check}%
⟨latexrelease⟩\def\@if@pti@ns#1#2{%
⟨latexrelease⟩ \let\reserved@a\@firstoftwo
⟨latexrelease⟩ \@for\reserved@b:=#2\do{%
⟨latexrelease⟩  \ifx\reserved@b\@empty
⟨latexrelease⟩   \else
⟨latexrelease⟩   \expandafter\in@\expandafter
⟨latexrelease⟩                   {\expandafter,\reserved@b,}{,#1,}%
⟨latexrelease⟩    \ifin@
⟨latexrelease⟩    \else
⟨latexrelease⟩     \let\reserved@a\@secondoftwo
⟨latexrelease⟩    \fi
⟨latexrelease⟩  \fi
⟨latexrelease⟩ }%
```

```
⟨latexrelease⟩ \reserved@a}
⟨latexrelease⟩\EndIncludeInRelease
⟨*2ekernel⟩
```

```
\@onlypreamble\@if@pti@ns
```

\ProvidesPackage Checks that the current filename is correct, and defines \ver@filename.

```
\def\ProvidesPackage#1{%
  \xdef\@gtempa{#1}%
  \ifx\@gtempa\@currname\else
    \@latex@warning@no@line{You have requested
      \@cls@pkg\space`\@currname',\MessageBreak
       but the \@cls@pkg\space provides `#1'}%
  \fi
  \@ifnextchar[\@pr@videpackage{\@pr@videpackage[]}}%]
\@onlypreamble\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.

```
⟨/2ekernel⟩
⟨*2ekernel | latexrelease⟩
⟨latexrelease⟩\IncludeInRelease{2020/02/02}%
⟨latexrelease⟩                 {\@pr@videpackage}{Protection for package info}%
\def\@pr@videpackage[#1]{%
  \expandafter\protected@xdef                %     <-- protected...
     \csname ver@\@currname.\@currext\endcsname{#1}%
  \ifx\@currext\@clsextension
    \typeout{Document Class: \@gtempa\space#1}%
  \else
    \protected@wlog{Package: \@gtempa\space#1}%   <--- protected
  \fi}
```

\protected@wlog This is like plain TEX's \wlog but gracefully handles protected commands.

```
\long\def\protected@wlog#1{\begingroup
  \set@display@protect
  \immediate \write \m@ne {#1}\endgroup }
```

```
⟨/2ekernel | latexrelease⟩
⟨latexrelease⟩\EndIncludeInRelease
⟨latexrelease⟩\IncludeInRelease{0000/00/00}%
⟨latexrelease⟩                 {\@pr@videpackage}{Protection for package info}%
⟨latexrelease⟩
⟨latexrelease⟩\def\@pr@videpackage[#1]{%
⟨latexrelease⟩  \expandafter\xdef\csname ver@\@currname.\@currext\endcsname{#1}%
⟨latexrelease⟩  \ifx\@currext\@clsextension
⟨latexrelease⟩    \typeout{Document Class: \@gtempa\space#1}%
⟨latexrelease⟩  \else
⟨latexrelease⟩    \wlog{Package: \@gtempa\space#1}%
⟨latexrelease⟩  \fi}
⟨latexrelease⟩\let\protected@wlog\@undefined
⟨latexrelease⟩
⟨latexrelease⟩\EndIncludeInRelease
⟨*2ekernel⟩
```

```
\@onlypreamble\@pr@videpackage
```

`\ProvidesClass`

Like `\ProvidesPackage`, but for classes.

```
\let\ProvidesClass\ProvidesPackage
\@onlypreamble\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`

```
\def\ProvidesFile#1{%
  \begingroup
    \catcode`\ 10 %
    \ifnum \endlinechar<256 %
      \ifnum \endlinechar>\m@ne
        \catcode\endlinechar 10 %
      \fi
    \fi
    \@makeother\/%
    \@makeother\&%
```

```
    \kernel@ifnextchar[{\@providesfile{#1}}{\@providesfile{#1}[]}}
```

During initex 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}
```

`\PassOptionsToPackage`
`\PassOptionsToClass`

If the package has been loaded, we check that it was first loaded with the options. Otherwise we add the option list to that of the package.

```
\def\@pass@ptions#1#2#3{%
  \expandafter\xdef\csname opt@#3.#1\endcsname{%
    \@ifundefined{opt@#3.#1}\@empty
      {\csname opt@#3.#1\endcsname,}%
    \zap@space#2 \@empty}}
\@onlypreamble\@pass@ptions
```

```
\def\PassOptionsToPackage{\@pass@ptions\@pkgextension}
\def\PassOptionsToClass{\@pass@ptions\@clsextension}
\@onlypreamble\PassOptionsToPackage
\@onlypreamble\PassOptionsToClass
```

`\DeclareOption`
`\DeclareOption*`

Adds an option as a `\ds@` command, or the default `\default@ds` command.

```
\def\DeclareOption{%
  \let\@fileswith@pti@ns\@badrequireerror
  \@ifstar\@defdefault@ds\@declareoption}
\long\def\@declareoption#1#2{%
   \xdef\@declaredoptions{\@declaredoptions,#1}%
   \toks@{#2}%
   \expandafter\edef\csname ds@#1\endcsname{\the\toks@}}
\long\def\@defdefault@ds#1{%
  \toks@{#1}%
  \edef\default@ds{\the\toks@}}
\@onlypreamble\DeclareOption
\@onlypreamble\@declareoption
\@onlypreamble\@defdefault@ds
```

\OptionNotUsed If we are in a class file, add \CurrentOption to the list of unused options. Otherwise, in a package file do nothing.

```
\def\OptionNotUsed{%
  \ifx\@currext\@clsextension
    \xdef\@unusedoptionlist{%
      \ifx\@unusedoptionlist\@empty\else\@unusedoptionlist,\fi
      \CurrentOption}%
  \fi}
\@onlypreamble\OptionNotUsed
```

\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*.

```
% \let\default@ds\OptionNotUsed
```

\ProcessOptions
\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.

```
\def\ProcessOptions{%
  \let\ds@\@empty
  \edef\@curroptions{\@ptionlist{\@currname.\@currext}}%
  \@ifstar\@xprocess@ptions\@process@ptions}
\@onlypreamble\ProcessOptions
```

```
\def\@process@ptions{%
  \@for\CurrentOption:=\@declaredoptions\do{%
    \ifx\CurrentOption\@empty\else
      \@expandtwoargs\in@{,\CurrentOption,}{%
         ,\ifx\@currext\@clsextension\else\@classoptionslist,\fi
         \@curroptions,}%
      \ifin@
        \@use@ption
        \expandafter\let\csname ds@\CurrentOption\endcsname\@empty
      \fi
    \fi}%
  \@process@pti@ns}
\@onlypreamble\@process@ptions
```

```
\def\@xprocess@ptions{%
  \ifx\@currext\@clsextension\else
    \@for\CurrentOption:=\@classoptionslist\do{%
      \ifx\CurrentOption\@empty\else
        \@expandtwoargs\in@{,\CurrentOption,}{,\@declaredoptions,}%
        \ifin@
          \@use@ption
          \expandafter\let\csname ds@\CurrentOption\endcsname\@empty
        \fi
      \fi}%
  \fi
  \@process@pti@ns}
\@onlypreamble\@xprocess@ptions
```

The common part of \ProcessOptions and \ProcessOptions*.

```
\def\@process@pti@ns{%
  \@for\CurrentOption:=\@curroptions\do{%
    \@ifundefined{ds@\CurrentOption}%
      {\@use@ption
       \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.

```
      \@use@ption}%
```

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.

```
  \@for\CurrentOption:=\@declaredoptions\do{%
    \expandafter\let\csname ds@\CurrentOption\endcsname\relax}%

  \let\CurrentOption\@empty
  \let\@fileswith@pti@ns\@@fileswith@pti@ns
  \AtEndOfPackage{\let\@unprocessedoptions\relax}}
\@onlypreamble\@process@pti@ns
```

\@options

\@options is a synonym for \ProcessOptions* for upward compatibility with LaTeX2.09 style files.

```
\def\@options{\ProcessOptions*}
\@onlypreamble\@options
```

\@use@ption

Execute the code for the current option.

```
\def\@use@ption{%
  \@expandtwoargs\@removeelement\CurrentOption
  \@unusedoptionlist\@unusedoptionlist
  \csname ds@\CurrentOption\endcsname}
\@onlypreamble\@use@ption
```

\ExecuteOptions

\ExecuteOptions{⟨*option-list*⟩} executes the code declared for each option.

```
⟨/2ekernel⟩
⟨latexrelease⟩\IncludeInRelease{2017/01/01}%
⟨latexrelease⟩                 {\ExecuteOptions}{Spaces in \ExecuteOptions}%
⟨*2ekernel | latexrelease⟩
\def\ExecuteOptions#1{%
```

Use \@fortmp here as it is anyway cleared during \@for loop so does not change any existing names.

```
  \edef\@fortmp{\zap@space#1 \@empty}%
  \def\reserved@a##1\@nil{%
    \@for\CurrentOption:=\@fortmp\do
             {\csname ds@\CurrentOption\endcsname}%
    \edef\CurrentOption{##1}}%
  \expandafter\reserved@a\CurrentOption\@nil}
```

```
⟨/2ekernel | latexrelease⟩
⟨latexrelease⟩\EndIncludeInRelease
⟨latexrelease⟩\IncludeInRelease{0000/00/00}%
⟨latexrelease⟩                 {\ExecuteOptions}{Spaces in \ExecuteOptions}%
⟨latexrelease⟩\def\ExecuteOptions#1{%
⟨latexrelease⟩  \def\reserved@a##1\@nil{%
⟨latexrelease⟩    \@for\CurrentOption:=#1\do
⟨latexrelease⟩              {\csname ds@\CurrentOption\endcsname}%
⟨latexrelease⟩    \edef\CurrentOption{##1}}%
⟨latexrelease⟩  \expandafter\reserved@a\CurrentOption\@nil}
⟨latexrelease⟩\EndIncludeInRelease
⟨*2ekernel⟩
```

```
\@onlypreamble\ExecuteOptions
```

The top-level commands, which just set some parameters then call the internal command, `\@fileswithoptions`.

`\documentclass` The main new-style class declaration.

```
\def\documentclass{%
  \let\documentclass\@twoclasseserror
  \if@compatibility\else\let\usepackage\RequirePackage\fi
  \@fileswithoptions\@clsextension}
\@onlypreamble\documentclass
```

`\documentstyle` 2.09 style class ‘style’ declaration.

```
\def\documentstyle{%
  \makeatletter\input{latex209.def}\makeatother
  \documentclass}
\@onlypreamble\documentstyle
```

`\RequirePackage` Load package if not already loaded.

```
\def\RequirePackage{%
  \@fileswithoptions\@pkgextension}
\@onlypreamble\RequirePackage
```

`\LoadClass` Load class.

```
\def\LoadClass{%
  \ifx\@currext\@pkgextension
     \@latex@error
      {\noexpand\LoadClass in package file}%
      {You may only use \noexpand\LoadClass in a class file.}%
  \fi
  \@fileswithoptions\@clsextension}
\@onlypreamble\LoadClass
```

`\@loadwithoptions` Pass the current option list on to a class or package. `#1` is `\@`*cls-or-pkg*`extension`, `#2` is `\RequirePackage` or `\LoadClass`, `#3` is the class or package to be loaded.

```
\def\@loadwithoptions#1#2#3{%
  \expandafter\let\csname opt@#3.#1\expandafter\endcsname
       \csname opt@\@currname.\@currext\endcsname
   #2{#3}}
\@onlypreamble\@loadwithoptions
```

\LoadClassWithOptions Load class ‘#1’ with the current option list.

```
\def\LoadClassWithOptions{%
  \@loadwithoptions\@clsextension\LoadClass}
\@onlypreamble\LoadClassWithOptions
```

\RequirePackageWithOptions Load package ‘#1’ with the current option list.

```
\def\RequirePackageWithOptions{%
  \AtEndOfPackage{\let\@unprocessedoptions\relax}%
  \@loadwithoptions\@pkgextension\RequirePackage}
\@onlypreamble\RequirePackageWithOptions
```

\usepackage To begin with, \usepackage produces an error. This is reset by \documentclass.

```
\def\usepackage#1#{%
  \@latex@error
    {\noexpand \usepackage before \string\documentclass}%
    {\noexpand \usepackage may only appear in the document
      preamble, i.e.,\MessageBreak
      between \noexpand\documentclass and
      \string\begin{document}.}%
  \@gobble}
\@onlypreamble\usepackage
```

\NeedsTeXFormat Check that the document is running on the correct system.

```
\def\NeedsTeXFormat#1{%
  \def\reserved@a{#1}%
  \ifx\reserved@a\fmtname
    \expandafter\@needsformat
  \else
     \@latex@error{This file needs format ‘\reserved@a’%
       \MessageBreak but this is ‘\fmtname’}{%
       The current input file will not be processed
       further,\MessageBreak
       because it was written for some other flavor of
       TeX.\MessageBreak\@ehd}%
```

If the file is not meant to be processed by LaTeX 2_ε we stop inputting it, but we do not end the run. We just end inputting the current file.

```
      \endinput \fi}
\@onlypreamble\NeedsTeXFormat
```

```
\def\@needsformat{%
  \@ifnextchar[%]
    \@needsf@rmat
    {}}
\@onlypreamble\@needsformat
```

```
\def\@needsf@rmat[#1]{%
    \@ifl@t@r\fmtversion{#1}{}%
    {\@latex@warning@no@line
        {You have requested release ‘#1’ of LaTeX,\MessageBreak
         but only release ‘\fmtversion’ is available}}}
\@onlypreamble\@needsf@rmat
```

`\zap@space`

`\zap@space foo`⟨*space*⟩`\@empty` removes all spaces from `foo` that are not protected by `{ }` groups.

```
\def\zap@space#1 #2{%
  #1%
  \ifx#2\@empty\else\expandafter\zap@space\fi
  #2}
```

`\@fileswithoptions`

The common part of `\documentclass` and `\usepackage`.

```
\def\@fileswithoptions#1{%
  \@ifnextchar[%]
    {\@fileswith@ptions#1}%
    {\@fileswith@ptions#1[]}}
\@onlypreamble\@fileswithoptions
```

```
\def\@fileswith@ptions#1[#2]#3{%
  \@ifnextchar[%]
  {\@fileswith@pti@ns#1[{#2}]#3}%
  {\@fileswith@pti@ns#1[{#2}]#3[]}}
\@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.

```
⟨/2ekernel⟩
⟨latexrelease⟩\IncludeInRelease{2017/01/01}%
⟨latexrelease⟩          {\@fileswith@pti@ns}{ifx tests in \@fileswith@pti@ns}%
⟨*2ekernel | latexrelease⟩
\def\@fileswith@pti@ns#1[#2]#3[#4]{%
  \ifx#1\@clsextension
    \ifx\@classoptionslist\relax
      \xdef\@classoptionslist{\zap@space#2 \@empty}%
      \def\reserved@a{%
        \@onefilewithoptions#3[{#2}][{#4}]#1%
        \@documentclasshook}%
    \else
      \def\reserved@a{%
        \@onefilewithoptions#3[{#2}][{#4}]#1}%
    \fi
  \else
```

build up a list of calls to `\@onefilewithoptions` (one for each package) without thrashing the parameter stack.

```
    \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)

```
      \ifx\@nnil##1\relax\else
```

If `\ifx\@nnil##1\n@nil` is true then `#1` is (presumably) empty (Older code used `\relax` which is slighly easier to get into `#1` by mistake, which would spoil this test.)

```
        \ifx\@nnil##1\@nnil\else
         \noexpand\@onefilewithoptions##1[{#2}][{#4}]%
         \noexpand\@pkgextension
        \fi
        \expandafter\reserved@b
      \fi}%
      \edef\reserved@a{\zap@space#3 \@empty}%
      \edef\reserved@a{\expandafter\reserved@b\reserved@a,\@nnil,}%
  \fi
  \reserved@a}
⟨/2ekernel | latexrelease⟩
```

```
⟨latexrelease⟩\EndIncludeInRelease
⟨latexrelease⟩\IncludeInRelease{0000/00/00}%
⟨latexrelease⟩          {\@fileswith@pti@ns}{ifx tests in \@fileswith@pti@ns}%
⟨latexrelease⟩\def\@fileswith@pti@ns#1[#2]#3[#4]{%
⟨latexrelease⟩  \ifx#1\@clsextension
⟨latexrelease⟩    \ifx\@classoptionslist\relax
⟨latexrelease⟩      \xdef\@classoptionslist{\zap@space#2 \@empty}%
⟨latexrelease⟩      \def\reserved@a{%
⟨latexrelease⟩        \@onefilewithoptions#3[{#2}][{#4}]#1%
⟨latexrelease⟩        \@documentclasshook}%
⟨latexrelease⟩    \else
⟨latexrelease⟩      \def\reserved@a{%
⟨latexrelease⟩        \@onefilewithoptions#3[{#2}][{#4}]#1}%
⟨latexrelease⟩    \fi
⟨latexrelease⟩  \else
⟨latexrelease⟩    \def\reserved@b##1,{%
⟨latexrelease⟩      \ifx\@nil##1\relax\else
⟨latexrelease⟩        \ifx\relax##1\relax\else
⟨latexrelease⟩         \noexpand\@onefilewithoptions##1[{#2}][{#4}]%
⟨latexrelease⟩         \noexpand\@pkgextension
⟨latexrelease⟩        \fi
⟨latexrelease⟩        \expandafter\reserved@b
⟨latexrelease⟩      \fi}%
⟨latexrelease⟩      \edef\reserved@a{\zap@space#3 \@empty}%
⟨latexrelease⟩      \edef\reserved@a{%
⟨latexrelease⟩        \expandafter\reserved@b\reserved@a,\@nil,}%
⟨latexrelease⟩  \fi
⟨latexrelease⟩  \reserved@a}
⟨latexrelease⟩\EndIncludeInRelease
⟨*2ekernel⟩
```

```
\@onlypreamble\@fileswith@pti@ns
```

Have the main argument as `#1`, so we only need one `\expandafter` above.

```
\def\@onefilewithoptions#1[#2][#3]#4{%
  \@pushfilename
  \xdef\@currname{#1}%
  \global\let\@currext#4%
  \let\CurrentOption\@empty
  \@reset@ptions
  \makeatletter
```

Grab everything in a macro, so the parameter stack is popped before any processing begins.

```
  \def\reserved@a{%
    \@ifl@aded\@currext{#1}%
      {\@if@ptions\@currext{#1}{#2}{}%
        {\@latex@error
            {Option clash for \@cls@pkg\space #1}%
            {The package #1 has already been loaded
             with options:\MessageBreak
             \space\space[\@ptionlist{#1.\@currext}]\MessageBreak
             There has now been an attempt to load it
              with options\MessageBreak
             \space\space[#2]\MessageBreak
             Adding the global options:\MessageBreak
             \space\space
                  \@ptionlist{#1.\@currext},#2\MessageBreak
             to your \noexpand\documentclass declaration may fix this.%
             \MessageBreak
             Try typing \space <return> \space to proceed.}}}%
      {\@pass@ptions\@currext{#2}{#1}%

       \global\expandafter
       \let\csname ver@\@currname.\@currext\endcsname\@empty
```

We initialize `\...-h@@k` here and only if we load the file so that it remains undefined otherwise.

```
       \expandafter\let\csname\@currname.\@currext-h@@k\endcsname\@empty
       \InputIfFileExists
         {\@currname.\@currext}%
         {}%
         {\@missingfileerror\@currname\@currext}%
```

`\@unprocessedoptions` will generate an error for each specified option in a package unless a `\ProcessOptions` has appeared in the package file.

```
    \let\@unprocessedoptions\@@unprocessedoptions
    \csname\@currname.\@currext-h@@k\endcsname
    \expandafter\let\csname\@currname.\@currext-h@@k\endcsname
              \@undefined
    \@unprocessedoptions}%
    \@ifl@ter\@currext{#1}{#3}{}%
      {\@latex@warning@no@line
         {You have requested,\on@line,
          version\MessageBreak
            `#3' of \@cls@pkg\space #1,\MessageBreak
          but only version\MessageBreak
           `\csname ver@#1.\@currext\endcsname'\MessageBreak
          is available}}%

    \ifx\@currext\@clsextension\let\LoadClass\@twoloadclasserror\fi
    \@popfilename
    \@reset@ptions}%
  \reserved@a}
\@onlypreamble\@onefilewithoptions
```

\@@fileswith@pti@ns

Save the definition (for error checking).

```
\let\@@fileswith@pti@ns\@fileswith@pti@ns
\@onlypreamble\@@fileswith@pti@ns
```

\@reset@ptions

Reset the default option, and clear lists of declared options.

```
\def\@reset@ptions{%
  \global\ifx\@currext\@clsextension
    \let\default@ds\OptionNotUsed
   \else
    \let\default@ds\@unknownoptionerror
  \fi
  \global\let\ds@\@empty
  \global\let\@declaredoptions\@empty}
\@onlypreamble\@reset@ptions
```

79.1 Hooks

Allow code do 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
\@enddocumenthook

Stuff to appear at the beginning or end of the document.

```
\ifx\@begindocumenthook\@undefined
  \let\@begindocumenthook\@empty
\fi
\let\@enddocumenthook\@empty
```

\g@addto@macro

Globally add to the end of a macro.

```
\long\def\g@addto@macro#1#2{%
  \begingroup
    \toks@\expandafter{#1#2}%
    \xdef#1{\the\toks@}%
  \endgroup}
```

\AtEndOfPackage
\AtEndOfClass
\AtBeginDocument
\AtEndDocument

The access functions.

```
\def\AtEndOfPackage{%
  \expandafter\g@addto@macro\csname\@currname.\@currext-h@@k\endcsname}
\let\AtEndOfClass\AtEndOfPackage
\@onlypreamble\AtEndOfPackage
\@onlypreamble\AtEndOfClass
```

```
\DeclareRobustCommand\AtBeginDocument{\g@addto@macro\@begindocumenthook}
\DeclareRobustCommand\AtEndDocument{\g@addto@macro\@enddocumenthook}
\@onlypreamble\AtBeginDocument
```

\@cls@pkg

The current file type.

```
\def\@cls@pkg{%
  \ifx\@currext\@clsextension
    document class%
  \else
    package%
  \fi}
\@onlypreamble\@cls@pkg
```

`\@unknownoptionerror` Bad option.

```
\def\@unknownoptionerror{%
  \@latex@error
    {Unknown option `\CurrentOption' for \@cls@pkg\space`\@currname'}%
    {The option `\CurrentOption' was not declared in
     \@cls@pkg\space`\@currname', perhaps you\MessageBreak
      misspelled its name.
     Try typing \space <return>
     \space to proceed.}}
\@onlypreamble\@unknownoptionerror
```

`\@@unprocessedoptions` Declare an error for each option, unless a `\ProcessOptions` occurred.

```
\def\@@unprocessedoptions{%
  \ifx\@currext\@pkgextension
    \edef\@curroptions{\@ptionlist{\@currname.\@currext}}%
    \@for\CurrentOption:=\@curroptions\do{%
        \ifx\CurrentOption\@empty\else\@unknownoptionerror\fi}%
  \fi}
\@onlypreamble\@unprocessedoptions
\@onlypreamble\@@unprocessedoptions
```

`\@badrequireerror` `\RequirePackage` or `\LoadClass` occurs in the options section.

```
\def\@badrequireerror#1[#2]#3[#4]{%
  \@latex@error
    {\noexpand\RequirePackage or \noexpand\LoadClass
         in Options Section}%
    {The \@cls@pkg\space `\@currname' is defective.\MessageBreak
     It attempts to load `#3' in the options section, i.e.,\MessageBreak
     between \noexpand\DeclareOption and \string\ProcessOptions.}}
\@onlypreamble\@badrequireerror
```

`\@twoloadclasserror` Two `\LoadClass` in a class.

```
\def\@twoloadclasserror{%
  \@latex@error
    {Two \noexpand\LoadClass commands}%
    {You may only use one \noexpand\LoadClass in a class file}}
\@onlypreamble\@twoloadclasserror
```

`\@twoclasseserror` Two `\documentclass` or `\documentstyle`.

```
\def\@twoclasseserror#1#{%
  \@latex@error
    {Two \noexpand\documentclass or \noexpand\documentstyle commands}%
    {The document may only declare one class.}\@gobble}
\@onlypreamble\@twoclasseserror
```

79.2 Providing shipment

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

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

`\filecontents`
`\endfilecontents` This environment implements inline files. The star-form does not write extra comments into the file.

```
% \changes{v1.3a}{2019/07/01}{Support UTF8 and spaces in
%                             filecontents environment file name}
% \changes{v1.3b}{2019/08/27}{Make various commands robust}
% \changes{v1.3c}{2019/09/11}{Support optional argument for filecontents}
% \changes{v1.3f}{2020/01/05}{Support more write streams in LuaTeX gh/238}
%
⟨/2ekernel⟩
⟨*2ekernel | latexrelease⟩
⟨latexrelease⟩\IncludeInRelease{2019/10/01}%
⟨latexrelease⟩                 {\filec@ntents}{Spaces in file names + optional arg}%
%
```

We use `@tempswa` to mean no preamble writing and reuse `@filesw` to indicate no overwriting:

```
\def\filecontents{\@tempswatrue\@fileswtrue
  \@ifnextchar[\filec@ntents@opt\filec@ntents
}
\@namedef{filecontents*}{\@tempswafalse\@fileswtrue
  \@ifnextchar[\filec@ntents@opt\filec@ntents
}
```

To handle the optional argument we execute for each option the command `\filec@ntents@OPTION` if it exist or complain about unknown option.

```
\def\filec@ntents@opt[#1]{%
  \edef\@fortmp{\zap@space#1 \@empty}%
  \@for\reserved@a:=\@fortmp\do{%
    \ifcsname filec@ntents@\reserved@a\endcsname
      \csname filec@ntents@\reserved@a\endcsname
    \else
    \@latex@error{Unknown filecontents option \reserved@a}%
       {Valid options are force (or overwrite), nosearch, noheader}%
    \fi}%
  \filec@ntents
}
```

Option `force`) (or `overwrite`) changes the overwriting switch

```
\let\filec@ntents@force\@fileswfalse
\let\filec@ntents@overwrite\@fileswfalse  % alternative name
```

and option `noheader` the preamble switch (which is equivalent to using the star form of the environment).

```
\let\filec@ntents@noheader\@tempswafalse
```

Option `nosearch` only checks the current directory not the how TEX tree for the existence of the file to write.

```
\def\filec@ntents@nosearch{%
  \let\filec@ntents@checkdir\@currdir
  \def\filec@ntents@where{in current directory}}
```

By default we search the whole tree:

```
\let\filec@ntents@checkdir\@empty
\def\filec@ntents@where{exists on the system}
```

```
\begingroup%
\@tempcnta=1
\loop
```

```
  \catcode\@tempcnta=12  %
  \advance\@tempcnta\@ne %
\ifnum\@tempcnta<32      %
\repeat                  %
\catcode`\*=11 %
\catcode`\^^M\active%
\catcode`\^^L\active\let^^L\relax%
\catcode`\^^I\active%

\gdef\filec@ntents#1{%
  \set@curr@file{\filec@ntents@checkdir#1}%
  \edef\q@curr@file{\expandafter\quote@name\expandafter{\@curr@file}}%
```

LuaTEX has more writes (and 18 is safe here).

```
  \chardef\reserved@c\ifx\directlua\@undefined 15 \else 127 \fi%
  \openin\@inputcheck\q@curr@file \space %
  \ifeof\@inputcheck%
    \@latex@warning@no@line%
        {Writing file `\@currdir\@curr@file'}%

    \ch@ck7\reserved@c\write\relax%
    \immediate\openout\reserved@c\q@curr@file\relax%
  \else%

    \if@filesw%
      \@latex@warning@no@line%
          {File `\@curr@file' already \filec@ntents@where.\MessageBreak%
             Not generating it from this source}%
      \let\write\@gobbletwo%
      \let\closeout\@gobble%
    \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!

```
      \edef\reserved@a{#1}%
      \edef\reserved@a{\detokenize\expandafter{\reserved@a}}%
      \edef\reserved@b{\detokenize\expandafter{\jobname}}%
      \ifx\reserved@a\reserved@b%
        \@fileswtrue%
      \else%
        \edef\reserved@b{\reserved@b\detokenize{.tex}}%
        \ifx\reserved@a\reserved@b
          \@fileswtrue%
        \fi%
      \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.

```
      \ch@ck7\reserved@c\write\relax%
      \if@filesw%  % Foul ... trying to overwrite \jobname!
      \@latex@error{Trying to overwrite `\jobname.tex'}{You can't %
        write to the file you are reading from!\MessageBreak%
        Data is written to screen instead.}%
      \else%
        \@latex@warning@no@line%
           {Writing or overwriting file `\@currdir\@curr@file'}%
        \immediate\openout\reserved@c\q@curr@file\relax%
      \fi%
    \fi%
  \fi%
```

Closing the `\@inputcheck` is done here to avoid having to do this in each branch.

```
\closein\@inputcheck%
\if@tempswa%
```

```
  \immediate\write\reserved@c{%
    \@percentchar\@percentchar\space%
        \expandafter\@gobble\string\LaTeX2e file `\@curr@file'^^J%
    \@percentchar\@percentchar\space  generated by the %
      `\@currenvir' \expandafter\@gobblefour\string\newenvironment^^J%
    \@percentchar\@percentchar\space from source `\jobname' on %
       \number\year/\two@digits\month/\two@digits\day.^^J%
    \@percentchar\@percentchar}%
\fi%
\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.

```
\count@ 128\relax%
\loop%
  \catcode\count@ 11\relax%
  \advance\count@ \@ne%
  \ifnum\count@<\@cclvi%
\repeat%
```

```
\edef\E{\@backslashchar end\string{\@currenvir\string}}%
\edef\reserved@b{%
  \def\noexpand\reserved@b%
       ####1\E####2\E####3\relax}%
\reserved@b{%
  \ifx\relax##3\relax%
```

There was no `\end{filecontents}`

```
      \immediate\write\reserved@c{##1}%
  \else%
```

There was a `\end{filecontents}`, so stop this time.

```
    \edef^^M{\noexpand\end{\@currenvir}}%
    \ifx\relax##1\relax%
    \else%
```

Text before the `\end`, write it with a warning.

```
        \@latex@warning{Writing text `##1' before %
           \string\end{\@currenvir}\MessageBreak as last line of \@curr@file}%
      \immediate\write\reserved@c{##1}%
    \fi%
    \ifx\relax##2\relax%
    \else%
```

Text after the \end, ignore it with a warning.

```
         \@latex@warning{%
           Ignoring text ‘##2’ after \string\end{\@currenvir}}%
      \fi%
    \fi%
    ^^M}%
```

```
  \catcode‘\^^L\active%
  \let\L\@undefined%
  \def^^L{\expandafter\ifx\csname L\endcsname\relax\fi ^^J^^J}%
  \catcode‘\^^I\active%
  \let\I\@undefined%
  \def^^I{\expandafter\ifx\csname I\endcsname\relax\fi\space}%
  \catcode‘\^^M\active%
  \edef^^M##1^^M{%
    \noexpand\reserved@b##1\E\E\relax}}%
\endgroup%
```

```
⟨/2ekernel | latexrelease⟩
⟨latexrelease⟩\EndIncludeInRelease
⟨latexrelease⟩\IncludeInRelease{0000/00/00}%
⟨latexrelease⟩                 {\filec@ntents}{Spaces in file names + optional arg}%
⟨latexrelease⟩
⟨latexrelease⟩\let\filec@ntents@opt        \@undefined
⟨latexrelease⟩\let\filec@ntents@force      \@undefined
⟨latexrelease⟩\let\filec@ntents@overwrite  \@undefined
⟨latexrelease⟩\let\filec@ntents@noheader   \@undefined
⟨latexrelease⟩\let\filec@ntents@nosearch   \@undefined
⟨latexrelease⟩\let\filec@ntents@checkdir   \@undefined
⟨latexrelease⟩\let\filec@ntents@where      \@undefined
⟨latexrelease⟩
⟨latexrelease⟩\begingroup%
⟨latexrelease⟩\@tempcnta=1
⟨latexrelease⟩\loop
⟨latexrelease⟩  \catcode\@tempcnta=12  %
⟨latexrelease⟩  \advance\@tempcnta\@ne %
⟨latexrelease⟩\ifnum\@tempcnta<32      %
⟨latexrelease⟩\repeat                  %
⟨latexrelease⟩\catcode‘\*=11 %
⟨latexrelease⟩\catcode‘\^^M\active%
⟨latexrelease⟩\catcode‘\^^L\active\let^^L\relax%
⟨latexrelease⟩\catcode‘\^^I\active%
⟨latexrelease⟩
⟨latexrelease⟩\gdef\filec@ntents#1{%
⟨latexrelease⟩  \openin\@inputcheck#1 %
⟨latexrelease⟩  \ifeof\@inputcheck%
⟨latexrelease⟩    \@latex@warning@no@line%
⟨latexrelease⟩        {Writing file ‘\@currdir#1’}%
⟨latexrelease⟩    \chardef\reserved@c15 %
⟨latexrelease⟩    \ch@ck7\reserved@c\write%
⟨latexrelease⟩    \immediate\openout\reserved@c#1\relax%
⟨latexrelease⟩  \else%
⟨latexrelease⟩    \closein\@inputcheck%
⟨latexrelease⟩    \@latex@warning@no@line%
⟨latexrelease⟩            {File ‘#1’ already exists on the system.\MessageBreak%
⟨latexrelease⟩             Not generating it from this source}%
⟨latexrelease⟩    \let\write\@gobbletwo%
⟨latexrelease⟩    \let\closeout\@gobble%
⟨latexrelease⟩  \fi%
⟨latexrelease⟩  \if@tempswa%
⟨latexrelease⟩    \immediate\write\reserved@c{%
⟨latexrelease⟩      \@percentchar\@percentchar\space%
⟨latexrelease⟩          \expandafter\@gobble\string\LaTeX2e file ‘#1’^^J%
⟨latexrelease⟩      \@percentchar\@percentchar\space  generated by the %
⟨latexrelease⟩        ‘\@currenvir’ \expandafter\@gobblefour\string\newenvironment^^J%
⟨latexrelease⟩      \@percentchar\@percentchar\space from source ‘\jobname’ on %
⟨latexrelease⟩         \number\year/\two@digits\month/\two@digits\day.^^J%
⟨latexrelease⟩      \@percentchar\@percentchar}%
⟨latexrelease⟩  \fi%
⟨latexrelease⟩  \let\do\@makeother\dospecials%
⟨latexrelease⟩  \count@ 128\relax%
⟨latexrelease⟩  \loop%
⟨latexrelease⟩    \catcode\count@ 11\relax%
⟨latexrelease⟩    \advance\count@ \@ne%
⟨latexrelease⟩    \ifnum\count@<\@cclvi%
⟨latexrelease⟩  \repeat%
⟨latexrelease⟩  \edef\E{\@backslashchar end\string{\@currenvir\string}}%
⟨latexrelease⟩  \edef\reserved@b{%
⟨latexrelease⟩    \def\noexpand\reserved@b%
⟨latexrelease⟩         ####1\E####2\E####3\relax}%
⟨latexrelease⟩  \reserved@b{%
⟨latexrelease⟩    \ifx\relax##3\relax%
⟨latexrelease⟩      \immediate\write\reserved@c{##1}%
⟨latexrelease⟩    \else%
⟨latexrelease⟩      \edef^^M{\noexpand\end{\@currenvir}}%
⟨latexrelease⟩      \ifx\relax##1\relax%
⟨latexrelease⟩      \else%
⟨latexrelease⟩          \@latex@warning{Writing text ‘##1’ before %
⟨latexrelease⟩             \string\end{\@currenvir}\MessageBreak as last line of #1}%
⟨latexrelease⟩        \immediate\write\reserved@c{##1}%
⟨latexrelease⟩      \fi%
⟨latexrelease⟩      \ifx\relax##2\relax%
⟨latexrelease⟩      \else%
⟨latexrelease⟩         \@latex@warning{%
⟨latexrelease⟩           Ignoring text ‘##2’ after \string\end{\@currenvir}}%
⟨latexrelease⟩      \fi%
⟨latexrelease⟩    \fi%
⟨latexrelease⟩    ^^M}%
⟨latexrelease⟩
⟨latexrelease⟩  \catcode‘\^^L\active%
⟨latexrelease⟩  \let\L\@undefined%
⟨latexrelease⟩  \def^^L{\expandafter\ifx\csname L\endcsname\relax\fi ^^J^^J}%
⟨latexrelease⟩  \catcode‘\^^I\active%
⟨latexrelease⟩  \let\I\@undefined%
⟨latexrelease⟩  \def^^I{\expandafter\ifx\csname I\endcsname\relax\fi\space}%
⟨latexrelease⟩  \catcode‘\^^M\active%
⟨latexrelease⟩  \edef^^M##1^^M{%
⟨latexrelease⟩    \noexpand\reserved@b##1\E\E\relax}}%
⟨latexrelease⟩\endgroup%
```

```
⟨latexrelease⟩\EndIncludeInRelease
⟨*2ekernel⟩
```

```
\begingroup
\catcode`|=\catcode`\%
\catcode`\%=12
\catcode`\*=11
\gdef\@percentchar{%}
\gdef\endfilecontents{|
  \immediate\closeout\reserved@c
  \def\T##1##2##3{|
  \ifx##1\@undefined\else
    \@latex@warning@no@line{##2 has been converted to Blank ##3e}|
  \fi}|
  \T\L{Form Feed}{Lin}|
  \T\I{Tab}{Spac}|
  \immediate\write\@unused{}}
\global\let\endfilecontents*\endfilecontents
```

We no longer prevent the code to be used after begin document (no rollback needed for this change).

```
%\@onlypreamble\filecontents
%\@onlypreamble\endfilecontents
%\@onlypreamble\filecontents*
%\@onlypreamble\endfilecontents*
\endgroup
%\@onlypreamble\filec@ntents
```

80 Package/class rollback mechanism

```
⟨/2ekernel⟩
⟨*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.

```
⟨*tracerollback⟩
%\let\pkgcls@debug\typeout
\let\pkgcls@debug\@gobble
⟨/tracerollback⟩
```

\requestedLaTeXdate The macro (!) \requestedLaTeXdate holds the globally requested rollback date (via latexrelease) or zero if no such request was made.

```
\def\requestedLaTeXdate{0}
```

\pkgcls@targetdate
\pkgcls@targetlabel
\pkgcls@innerdate

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 verion 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 TeX largest legal number. While loading a file this can be used to provide conditionals that select code based on the request.

```
\ifx\pkgcls@targetdate\@undefined
  \newcount\pkgcls@targetdate
\fi
\let\pkgcls@targetlabel\@empty
\def\pkgcls@innerdate{\maxdimen}
```

`\pkgcls@candidate` `\pkgcls@releasedate` 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.

```
\let\pkgcls@candidate\@empty
\let\pkgcls@releasedate\@empty
```

`\load@onefilewithoptions` `\@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.

```
\ifx\load@onefilewithoptions\@undefined
 \let\load@onefilewithoptions\@onefilewithoptions
```

```
 \def\@onefilewithoptions#1[#2][#3]#4{%
```

First a bit of tracing normally disabled.

```
⟨*tracerollback⟩
  \pkgcls@debug{--- File loaded request (\noexpand\usepackage or ...)}%
  \pkgcls@debug{\@spaces 1: #1}%
  \pkgcls@debug{\@spaces 2: #2}%
  \pkgcls@debug{\@spaces 3: #3}%
  \pkgcls@debug{\@spaces 4: #4}%
⟨/tracerollback⟩
```

Two of the arguments are needed later on in error/warning messages so we save them.

```
  \def\pkgcls@name{#1}%                  % for info message
  \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`.

```
  \pkgcls@parse@date@arg{#3}%
```

When determining the correct release to load we keep track of candiates in `\pkgcls@candidate` and initially we don't have any:

```
  \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.

```
  \begingroup
  \edef\reserved@a{%
    \endgroup
    \unexpanded{\load@onefilewithoptions#1[#2]}%
    [\pkgcls@mindate]%
    \unexpanded{#4}}%
   \reserved@a
 }
\fi
```

\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 tht 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		\pkgcls@targetdate	\pkgcls@targetlabel	\pkgcls@mindate
⟨*empty*⟩	→	⟨*global-rollbackdate-as-number*⟩	⟨*empty*⟩	⟨*empty*⟩
⟨*date*⟩	→	⟨*global-rollbackdate-as-number*⟩	⟨*empty*⟩	⟨*date*⟩
=⟨*date*⟩	→	⟨*date-as-number*⟩	⟨*empty*⟩	⟨*empty*⟩
=⟨*version*⟩	→	1	⟨*version*⟩	⟨*empty*⟩
⟨*other*⟩	→	⟨*global-rollbackdate-as-number*⟩	⟨*empty*⟩	⟨*other*⟩

where ⟨*global-rollbackdate-as-number*⟩ is a date request given via latexrelease or if there wasn't one 0.

```
\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.

```
  \ifx\@nil#1\@nil
    \pkgcls@targetdate\requestedLaTeXdate\relax
    \let\pkgcls@targetlabel\@empty
    \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.

```
   \else
     \pkgcls@parse@date@arg@#1=\@nil\relax
   \fi
 }
```

The actual parsing work then happens in `\pkgcls@parse@date@arg@`:

```
\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.

```
  \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.

```
    \ifx\@nil#1\@nil
      \@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 `=`.

```
      \ifnum \pkgcls@targetdate=\z@
        \pkgcls@targetdate\@ne
        \def\pkgcls@innerdate{\maxdimen}%
        \pkgcls@parse@date@arg@version#2%
      \else
        \edef\pkgcls@innerdate{\the\pkgcls@targetdate}%
      \fi
      \let\pkgcls@mindate\@empty
    \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`):

```
      \requestedLaTeXdate\relax
      \let\pkgcls@targetlabel\@empty
      \def\pkgcls@innerdate{\maxdimen}%
      \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.

```
      \ifnum \pkgcls@targetdate > \z@
        \ifnum \@parse@version0#1//00\@nil > \pkgcls@targetdate
          \@latex@warning@no@line{Suspicious rollback/min-date date given\MessageBreak
            A minimal date of #1 has been specified for
             \@cls@pkg\MessageBreak '\pkgcls@name'.\MessageBreak
             But this is in conflict
             with a rollback request to \requestedpatchdate}
        \fi
      \fi
    \fi
}
```

Strip off the trailing = and assign the version name to `\pkgcls@targetlabel`.

```
\def\pkgcls@parse@date@arg@version#1={%
  \def\pkgcls@targetlabel{#1}}
```

`\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 ⟨*pkg/cls-name*⟩-⟨*date*⟩.⟨*extension*⟩ but this is not enforced and through this argument one can overwrite it.

```
\def\DeclareRelease#1#2#3{%
  \ifnum\pkgcls@targetdate>\z@  % some sort of rollback request
⟨*tracerollback⟩
    \pkgcls@debug{---\string\DeclareRelease:}%
    \pkgcls@debug{\@spaces 1: #1}%
    \pkgcls@debug{\@spaces 2: #2}%
    \pkgcls@debug{\@spaces 3: #3}%
⟨/tracerollback⟩
```

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.

```
    \ifx\@nil#2\@nil
      \ifnum\pkgcls@targetdate=\@ne  % named request
        \def\reserved@a{#1}%
        \ifx\pkgcls@targetlabel\reserved@a
          \pkgcls@use@this@release{#3}{}%
⟨*tracerollback⟩
        \else
          \pkgcls@debug{Label doesn't match}%
⟨/tracerollback⟩
        \fi
⟨*tracerollback⟩
      \else
        \pkgcls@debug{Date request: ignored}%
⟨/tracerollback⟩
      \fi
    \else
```

If the value of `\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.

```
      \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.

```
        \ifnum\@parse@version#2//00\@nil
             >\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.

```
          \ifx\pkgcls@candidate\@empty
            \pkgcls@rollbackdate@error{#2}%
            \pkgcls@use@this@release{#3}{#2}%
          \else
            \pkgcls@use@this@release\pkgcls@candidate
                                     \pkgcls@releasedate
          \fi
        \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.

```
            \def\pkgcls@candidate{#3}%
            \def\pkgcls@releasedate{#2}%
⟨*tracerollback⟩
            \pkgcls@debug{New candidate: #3}%
⟨/tracerollback⟩
        \fi
      \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.

```
        \def\reserved@a{#1}%
        \ifx\pkgcls@targetlabel\reserved@a
          \pkgcls@use@this@release{#3}{#2}%
⟨*tracerollback⟩
        \else
          \pkgcls@debug{Label doesn't match}%
⟨/tracerollback⟩
        \fi
      \fi
    \fi
  \fi
}
```

`\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.

```
\def\pkgcls@use@this@release#1#2{%
```

Before that we record the selection made inside the transcript.

```
  \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 bypassed[12] and then we finally load the correct release.

[12] The 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.

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!

```
  \pkgcls@targetdate\z@
  \@@input #1\relax
  \endinput
}
```

\pkgcls@show@selection

This command records what selection was made. As that is needed in two places (and it is rather lengthly) 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.

```
\def\pkgcls@show@selection#1#2{%
⟨*tracerollback⟩
  \pkgcls@debug{Result: use  #1}%
⟨/tracerollback⟩
  \GenericInfo
   {\@spaces\@spaces\space}{Rollback for
    \@cls@pkg\space'\@currname' requested ->
    \ifnum\pkgcls@targetdate>\@ne
       date
       \ifnum\requestedLaTeXdate=\pkgcls@targetdate
          \requestedpatchdate
       \else
          \expandafter\@gobble\pkgcls@arg
       \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.

```
       Best approximation is
    \else
       version '\pkgcls@targetlabel'.\MessageBreak
       This corresponds to
    \fi
    \ifx\@nil#2\@nil
       a special release%
    \else
       the release introduced on #2%
    \fi
    \@gobble}%
}
```

\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.

```
\def\pkgcls@rollbackdate@error#1{%
  \@latex@error{Suspicious rollback date given}%
     {The \@cls@pkg\space'\@currname'  claims that it
      came into existence on #1 which\MessageBreak
      is after your requested rollback date --- so
      something is wrong here.\MessageBreak
      Continue and we use the earliest known release.}}
```

\DeclareCurrentRelease

This declares the date (and possible name) of the current version of a package or class.

```
\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).

```
  \ifnum\pkgcls@targetdate>\z@  % some sort of rollback request
⟨*tracerollback⟩
    \pkgcls@debug{---DeclareCurrentRelease}%
    \pkgcls@debug{   1: #1}%
    \pkgcls@debug{   2: #2}%
⟨/tracerollback⟩
```

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.

```
    \ifnum\pkgcls@targetdate>\@ne  % a date request
      \ifnum\@parse@version#2//00\@nil
           >\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).

```
        \ifx\pkgcls@candidate\@empty
          \pkgcls@rollbackdate@error{#2}%
        \else
          \pkgcls@use@this@release\pkgcls@candidate
                                  \pkgcls@releasedate
        \fi
```

Otherwise the curent file is the right release, so we record that in the transcript and then carry on.

```
      \else
        \pkgcls@show@selection{current version}{#2}%
      \fi
    \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.

```
      \def\reserved@a{#1}%
      \ifx\pkgcls@targetlabel\reserved@a
        \pkgcls@show@selection{current version}{#2}%
      \else
        \@latex@error{Requested version '\pkgcls@targetlabel' for
          \@cls@pkg\space'\@currname' is unknown}\@ehc
      \fi
    \fi
  \fi
}
```

\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.

```
\DeclareRobustCommand\IfTargetDateBefore[1]{%
  \ifnum\pkgcls@innerdate <%
        \expandafter\@parse@version\expandafter0#1//00\@nil
    \typeout{Exclude code introduced on #1}%
    \expandafter\@firstoftwo
  \else
    \typeout{Include code introduced on #1}%
    \expandafter\@secondoftwo
  \fi
}
```

⟨/2ekernel | latexreleasefirst⟩

81 After Preamble

Finally we declare a package that allows all the commands declared above to be \@onlypreamble to be used after \begin{document}.

```
⟨*afterpreamble⟩
\NeedsTeXFormat{LaTeX2e}
\ProvidesPackage{pkgindoc}
         [1994/10/20 v1.1 Package Interface in Document (DPC)]
\def\reserved@a#1\do\@classoptionslist#2\do\filec@ntents#3\relax{%
  \gdef\@preamblecmds{#1#3}}
\expandafter\reserved@a\@preamblecmds\relax
⟨/afterpreamble⟩
```

File O
lthyphen.dtx

This file contains the code for loading hyphenation patterns into LaTeX. Most of this will end up in a file called `hyphen.ltx`. If you wish to customize your LaTeX 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 LaTeX 2_ε.

```
⟨*driver⟩
\documentclass{ltxdoc}
\begin{document}
\DocInput{lthyphen.dtx}
\end{document}
⟨/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 TeX'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.

```
⟨*default⟩
\InputIfFileExists{hyphen.tex}%
   {\message{Loading hyphenation patterns for US english.}%
    \language=0
    \lefthyphenmin=2 \righthyphenmin=3 }%
```

Otherwise, since we cannot do anything without any hyphenation patterns, an error message is printed and the IniTeX run is terminated by invoking `\@@end` (which is the LaTeX 2_ε name for TeX's `\end` primitive).

```
   {\errhelp{The configuration for hyphenation is incorrectly
             installed.^^J%
             If you don’t understand this error message you need
             to seek^^Jexpert advice.}%
    \errmessage{OOPS! I can’t find any hyphenation patterns for
                US english.^^J \space Think of getting some or the
                latex2e setup will never succeed}\@@end}
⟨/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 P
ltluatex.dtx

82 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 LaTeX 2ε kernel level plus as a loadable file which can be used with plain TeX and LaTeX.

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 LaTeX 2ε kernel from 2015 onward these registers are part of the block in the extended area reserved by the kernel (prior to 2015 the LaTeX 2ε kernel did not provide any functionality for the extended allocation area).

83 Core TeX functionality

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

`\newattribute` `\newattribute{`⟨*attribute*⟩`}`
Defines a named `\attribute`, indexed from 1 (*i.e.* `\attribute0` is never defined). Attributes initially have the marker value `-"7FFFFFFF` (‘unset’) set by the engine.

`\newcatcodetable` `\newcatcodetable{`⟨*catcodetable*⟩`}`
Defines a named `\catcodetable`, indexed from 1 (`\catcodetable0` is never assigned). A new catcode table will be populated with exactly those values assigned by IniTeX (as described in the LuaTeX manual).

`\newluafunction` `\newluafunction{`⟨*function*⟩`}`
Defines a named `\luafunction`, indexed from 1. (Lua indexes tables from 1 so `\luafunction0` is not available).

`\newwhatsit` `\newwhatsit{`⟨*whatsit*⟩`}`
Defines a custom `\whatsit`, indexed from 1.

`\newluabytecode` `\newluabytecode{`⟨*bytecode*⟩`}`

Allocates a number for Lua bytecode register, indexed from 1.

`\newluachunkname` `newluachunkname{`⟨*chunkname*⟩`}`
Allocates a number for Lua chunk register, indexed from 1. Also enters the name of the regiser (without backslash) into the `lua.name` table to be used in stack traces.

`\catcodetable@initex`
`\catcodetable@string`
`\catcodetable@latex`
`\catcodetable@atletter`
`\setattribute`
`\unsetattribute`

Predefined category code tables with the obvious assignments. Note that the `latex` and `atletter` tables set the full Unicode range to the codes predefined by the kernel.

`\setattribute{`⟨*attribute*⟩`}{`⟨*value*⟩`}`
`\unsetattribute{`⟨*attribute*⟩`}`
Set and unset attributes in a manner analogous to `\setlength`. Note that attributes take a marker value when unset so this operation is distinct from setting the value to zero.

84 Plain TEX interface

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

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

85 Lua functionality

85.1 Allocators in Lua

`new_attribute` `luatexbase.new_attribute(`⟨*attribute*⟩`)`
Returns an allocation number for the ⟨*attribute*⟩, indexed from 1. The attribute will be initialised with the marker value `-"7FFFFFFF` (‘unset’). The attribute allocation sequence is shared with the TEX code but this function does *not* define a token using `\attributedef`. The attribute name is recorded in the `attributes` table. A metatable is provided so that the table syntax can be used consistently for attributes declared in TEX or Lua.

`new_whatsit` `luatexbase.new_whatsit(`⟨*whatsit*⟩`)`
Returns an allocation number for the custom ⟨*whatsit*⟩, indexed from 1.

`new_bytecode` `luatexbase.new_bytecode(`⟨*bytecode*⟩`)`
Returns an allocation number for a bytecode register, indexed from 1. The optional ⟨*name*⟩ argument is just used for logging.

`new_chunkname` `luatexbase.new_chunkname(`⟨*chunkname*⟩`)`
Returns an allocation number for a Lua chunk name for use with `\directlua` and `\latelua`, indexed from 1. The number is returned and also ⟨*name*⟩ argument is added to the `lua.name` array at that index.

`new_luafunction` `luatexbase.new_luafunction(`⟨*functionname*⟩`)`
Returns an allocation number for a lua function for use with `\luafunction`, `\lateluafunction`, and `\luadef`, indexed from 1. The optional ⟨*functionname*⟩ argument is just used for logging.

These functions all require access to a named TEX count register to manage their allocations. The standard names are those defined above for access from TEX, *e.g.* “e@alloc@attribute@count, but these can be adjusted by defining the variable ⟨*type*⟩_count_name before loading ltluatex.lua, for example

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

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

85.2 Lua access to TEX register numbers

registernumber luatexbase.registernumer(⟨*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 LuaTEX's internal tables. After for example \newattribute\myattrib, \myattrib would be defined by (say) \myattrib=\attribute15. luatexbase.registernumer("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}

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

If the demonstration code is processed with LuaLATEX 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
sixt@@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).

85.3 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 LaTeX 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 LaTeX'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.

85.4 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.

disable_callback `luatexbase.disable_callback(`⟨*callback*⟩`)` Sets the ⟨*callback*⟩ to `false` as described in the LuaTEX manual for the underlying `callback.register` built-in. Callbacks will only be set to false (and thus be skipped entirely) if there are no functions registered using the callback.

callback_descriptions A list of the descriptions of functions registered to the specified callback is returned. `{}` is returned if there are no functions registered.

create_callback `luatexbase.create_callback(`⟨*name*⟩,metatype,⟨*default*⟩`)` Defines a user defined callback. The last argument is a default function or `false`.

call_callback `luatexbase.call_callback(`⟨*name*⟩,. . . `)` Calls a user defined callback with the supplied arguments.

86 Implementation

```
⟨*2ekernel | tex | latexrelease⟩
⟨2ekernel | latexrelease⟩\ifx\directlua\@undefined\else
```

86.1 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’.

```
⟨latexrelease⟩\IncludeInRelease{2015/10/01}
⟨latexrelease⟩                  {\newluafunction}{LuaTeX}%
\ifnum\luatexversion<60 %
  \wlog{***************************************************}
  \wlog{* LuaTeX version too old for ltluatex support *}
  \wlog{***************************************************}
  \expandafter\endinput
\fi
```

86.2 Older LATEX/Plain TEX setup

```
⟨*tex⟩
```

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

```
\directlua{tex.enableprimitives("",tex.extraprimitives("luatex"))}
```

```
\ifx\e@alloc\@undefined
```

In pre-2014 LATEX, or plain TEX, load `etex.{sty,src}`.

```
  \ifx\documentclass\@undefined
    \ifx\loccount\@undefined
      \input{etex.src}%
    \fi
    \catcode`\@=11 %
    \outer\expandafter\def\csname newfam\endcsname
                          {\alloc@8\fam\chardef\et@xmaxfam}
  \else
    \RequirePackage{etex}
    \expandafter\def\csname newfam\endcsname
                    {\alloc@8\fam\chardef\et@xmaxfam}
    \expandafter\let\expandafter\new@mathgroup\csname newfam\endcsname
  \fi
```

86.2.1 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.

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

luatex/xetex also allow more math fam.

```
\edef \et@xmaxfam {\ifx\Umathchar\@undefined\sixt@@n\else\@cclvi\fi}
```

```
\count 270=\et@xmaxregs % locally allocates \count registers
\count 271=\et@xmaxregs % ditto for \dimen registers
\count 272=\et@xmaxregs % ditto for \skip registers
\count 273=\et@xmaxregs % ditto for \muskip registers
\count 274=\et@xmaxregs % ditto for \box registers
\count 275=\et@xmaxregs % ditto for \toks registers
\count 276=\et@xmaxregs % ditto for \marks classes
```

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

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

End of proposed changes to `etex.src`

86.2.2 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.

```
\expandafter\let\csname newcount\expandafter\expandafter\endcsname
                \csname globcount\endcsname
\expandafter\let\csname newdimen\expandafter\expandafter\endcsname
                \csname globdimen\endcsname
\expandafter\let\csname newskip\expandafter\expandafter\endcsname
                \csname globskip\endcsname
\expandafter\let\csname newbox\expandafter\expandafter\endcsname
                \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.

```
\chardef\e@alloc@top=65535
\let\e@alloc@chardef\chardef
```

```
\def\e@alloc#1#2#3#4#5#6{%
  \global\advance#3\@ne
  \e@ch@ck{#3}{#4}{#5}#1%
  \allocationnumber#3\relax
  \global#2#6\allocationnumber
  \wlog{\string#6=\string#1\the\allocationnumber}}%
\gdef\e@ch@ck#1#2#3#4{%
  \ifnum#1<#2\else
    \ifnum#1=#2\relax
      #1\@cclvi
      \ifx\count#4\advance#1 10 \fi
    \fi
    \ifnum#1<#3\relax
    \else
      \errmessage{No room for a new \string#4}%
    \fi
  \fi}%
```

Two simple LaTeX macros used in `ltlatex.sty`.

```
\long\def\@gobble#1{}
\long\def\@firstofone#1{#1}
```

Fix up allocations not to clash with `etex.src`.

```
\expandafter\csname newcount\endcsname\e@alloc@attribute@count
\expandafter\csname newcount\endcsname\e@alloc@ccodetable@count
\expandafter\csname newcount\endcsname\e@alloc@luafunction@count
\expandafter\csname newcount\endcsname\e@alloc@whatsit@count
\expandafter\csname newcount\endcsname\e@alloc@bytecode@count
\expandafter\csname newcount\endcsname\e@alloc@luachunk@count
```

End of conditional setup for plain TeX / old LaTeX.

```
\fi
⟨/tex⟩
```

86.3 Attributes

`\newattribute`

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

```
\ifx\e@alloc@attribute@count\@undefined
  \countdef\e@alloc@attribute@count=258
\fi
\def\newattribute#1{%
  \e@alloc\attribute\attributedef
    \e@alloc@attribute@count\m@ne\e@alloc@top#1%
}
\e@alloc@attribute@count=\z@
```

`\setattribute`
`\unsetattribute`

Handy utilities.

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

86.4 Category code tables

`\newcatcodetable`

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

```
\ifx\e@alloc@ccodetable@count\@undefined
  \countdef\e@alloc@ccodetable@count=259
\fi
\def\newcatcodetable#1{%
  \e@alloc\catcodetable\chardef
    \e@alloc@ccodetable@count\m@ne{"8000}#1%
  \initcatcodetable\allocationnumber
}
\e@alloc@ccodetable@count=\z@
```

\catcodetable@initex
\catcodetable@string
\catcodetable@latex
\catcodetable@atletter

Save a small set of standard tables. The Unicode data is read here in using a parser simplified from that in `load-unicode-data`: only the nature of letters needs to be detected.

```
\newcatcodetable\catcodetable@initex
\newcatcodetable\catcodetable@string
\begingroup
  \def\setrangecatcode#1#2#3{%
    \ifnum#1>#2 %
      \expandafter\@gobble
    \else
      \expandafter\@firstofone
    \fi
      {%
        \catcode#1=#3 %
        \expandafter\setrangecatcode\expandafter
          {\number\numexpr#1 + 1\relax}{#2}{#3}
      }%
  }
  \@firstofone{%
    \catcodetable\catcodetable@initex
      \catcode0=12 %
      \catcode13=12 %
      \catcode37=12 %
      \setrangecatcode{65}{90}{12}%
      \setrangecatcode{97}{122}{12}%
      \catcode92=12 %
      \catcode127=12 %
      \savecatcodetable\catcodetable@string
    \endgroup
  }%
\newcatcodetable\catcodetable@latex
\newcatcodetable\catcodetable@atletter
\begingroup
  \def\parseunicodedataI#1;#2;#3;#4\relax{%
    \parseunicodedataII#1;#3;#2 First>\relax
  }%
  \def\parseunicodedataII#1;#2;#3 First>#4\relax{%
    \ifx\relax#4\relax
      \expandafter\parseunicodedataIII
    \else
      \expandafter\parseunicodedataIV
    \fi
      {#1}#2\relax%
  }%
  \def\parseunicodedataIII#1#2#3\relax{%
    \ifnum 0%
      \if L#21\fi
      \if M#21\fi
      >0 %
      \catcode"#1=11 %
    \fi
  }%
  \def\parseunicodedataIV#1#2#3\relax{%
    \read\unicoderead to \unicodedataline
    \if L#2%
      \count0="#1 %
      \expandafter\parseunicodedataV\unicodedataline\relax
    \fi
  }%
  \def\parseunicodedataV#1;#2\relax{%
    \loop
      \unless\ifnum\count0>"#1 %
        \catcode\count0=11 %
        \advance\count0 by 1 %
    \repeat
  }%
  \def\storedpar{\par}%
  \chardef\unicoderead=\numexpr\count16 + 1\relax
  \openin\unicoderead=UnicodeData.txt %
  \loop\unless\ifeof\unicoderead %
    \read\unicoderead to \unicodedataline
    \unless\ifx\unicodedataline\storedpar
      \expandafter\parseunicodedataI\unicodedataline\relax
    \fi
  \repeat
  \closein\unicoderead
  \@firstofone{%
    \catcode64=12 %
    \savecatcodetable\catcodetable@latex
    \catcode64=11 %
    \savecatcodetable\catcodetable@atletter
   }
\endgroup
```

86.5 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.

```
\ifx\e@alloc@luafunction@count\@undefined
  \countdef\e@alloc@luafunction@count=260
\fi
\def\newluafunction{%
  \e@alloc\luafunction\e@alloc@chardef
    \e@alloc@luafunction@count\m@ne\e@alloc@top
}
```

```
\e@alloc@luafunction@count=\z@
```

86.6 Custom whatsits

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

```
\ifx\e@alloc@whatsit@count\@undefined
  \countdef\e@alloc@whatsit@count=261
\fi
\def\newwhatsit#1{%
  \e@alloc\whatsit\e@alloc@chardef
    \e@alloc@whatsit@count\m@ne\e@alloc@top#1%
}
\e@alloc@whatsit@count=\z@
```

86.7 Lua bytecode registers

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

```
\ifx\e@alloc@bytecode@count\@undefined
  \countdef\e@alloc@bytecode@count=262
\fi
\def\newluabytecode#1{%
  \e@alloc\luabytecode\e@alloc@chardef
    \e@alloc@bytecode@count\m@ne\e@alloc@top#1%
}
\e@alloc@bytecode@count=\z@
```

86.8 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.

```
\ifx\e@alloc@luachunk@count\@undefined
  \countdef\e@alloc@luachunk@count=263
\fi
\def\newluachunkname#1{%
  \e@alloc\luachunk\e@alloc@chardef
    \e@alloc@luachunk@count\m@ne\e@alloc@top#1%
    {\escapechar\m@ne
    \directlua{lua.name[\the\allocationnumber]="\string#1"}}%
}
\e@alloc@luachunk@count=\z@
```

86.9 Lua loader

Load the Lua code at the start of every job. For the conversion of TEX 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.

```
⟨2ekernel⟩\everyjob\expandafter{%
⟨2ekernel⟩  \the\everyjob
  \begingroup
    \attributedef\attributezero=0 %
    \chardef     \charzero     =0 %
```

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

```
    \countdef    \CountZero    =0 %
    \dimendef    \dimenzero    =0 %
    \mathchardef \mathcharzero =0 %
    \muskipdef   \muskipzero   =0 %
    \skipdef     \skipzero     =0 %
    \toksdef     \tokszero     =0 %
    \directlua{require("ltluatex")}
  \endgroup
⟨2ekernel⟩}
⟨latexrelease⟩\EndIncludeInRelease
```

```
⟨latexrelease⟩\IncludeInRelease{0000/00/00}
⟨latexrelease⟩                 {\newluafunction}{LuaTeX}%
⟨latexrelease⟩\let\e@alloc@attribute@count\@undefined
⟨latexrelease⟩\let\newattribute\@undefined
⟨latexrelease⟩\let\setattribute\@undefined
⟨latexrelease⟩\let\unsetattribute\@undefined
⟨latexrelease⟩\let\e@alloc@ccodetable@count\@undefined
⟨latexrelease⟩\let\newcatcodetable\@undefined
⟨latexrelease⟩\let\catcodetable@initex\@undefined
⟨latexrelease⟩\let\catcodetable@string\@undefined
⟨latexrelease⟩\let\catcodetable@latex\@undefined
⟨latexrelease⟩\let\catcodetable@atletter\@undefined
⟨latexrelease⟩\let\e@alloc@luafunction@count\@undefined
⟨latexrelease⟩\let\newluafunction\@undefined
⟨latexrelease⟩\let\e@alloc@luafunction@count\@undefined
⟨latexrelease⟩\let\newwhatsit\@undefined
⟨latexrelease⟩\let\e@alloc@whatsit@count\@undefined
⟨latexrelease⟩\let\newluabytecode\@undefined
⟨latexrelease⟩\let\e@alloc@bytecode@count\@undefined
⟨latexrelease⟩\let\newluachunkname\@undefined
⟨latexrelease⟩\let\e@alloc@luachunk@count\@undefined
⟨latexrelease⟩\directlua{luatexbase.uninstall()}
⟨latexrelease⟩\EndIncludeInRelease
```

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

```
⟨latexrelease⟩\IncludeInRelease{2017/01/01}%
⟨latexrelease⟩                 {\fontencoding}{TU in everyjob}%
⟨latexrelease⟩\fontencoding{TU}\let\encodingdefault\f@encoding
⟨latexrelease⟩\ifx\directlua\@undefined\else
⟨2ekernel⟩\everyjob\expandafter{%
⟨2ekernel⟩  \the\everyjob
⟨*2ekernel, latexrelease⟩
  \directlua{%
  if xpcall(function ()%
             require('luaotfload-main')%
            end,texio.write_nl) then %
  local _void = luaotfload.main ()%
  else %
  texio.write_nl('Error in luaotfload: reverting to OT1')%
  tex.print('\string\\def\string\\encodingdefault{OT1}')%
  end %
  }%
  \let\f@encoding\encodingdefault
  \expandafter\let\csname ver@luaotfload.sty\endcsname\fmtversion
⟨/2ekernel, latexrelease⟩
⟨latexrelease⟩\fi
⟨2ekernel⟩  }
⟨latexrelease⟩\EndIncludeInRelease
⟨latexrelease⟩\IncludeInRelease{0000/00/00}%
⟨latexrelease⟩                 {\fontencoding}{TU in everyjob}%
⟨latexrelease⟩\fontencoding{OT1}\let\encodingdefault\f@encoding
⟨latexrelease⟩\EndIncludeInRelease
```

```
⟨2ekernel | latexrelease⟩\fi
⟨/2ekernel | tex | latexrelease⟩
```

86.10 Lua module preliminaries

```
⟨*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.

```
luatexbase       = luatexbase or { }
local luatexbase = luatexbase
```

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

```
local string_gsub      = string.gsub
local tex_count        = tex.count
local tex_setattribute = tex.setattribute
local tex_setcount     = tex.setcount
local texio_write_nl   = texio.write_nl
```

```
local luatexbase_warning
local luatexbase_error
```

86.11 Lua module utilities

86.11.1 Module tracking

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

```
local modules = modules or { }
```

provides_module Local function to write to the log.

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

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

```
local function provides_module(info)
  if not (info and info.name) then
    luatexbase_error("Missing module name for provides_module")
  end
  local function spaced(text)
    return text and (" " .. text) or ""
  end
  luatexbase_log(
    "Lua module: " .. info.name
      .. spaced(info.date)
      .. spaced(info.version)
      .. spaced(info.description)
  )
  modules[info.name] = info
end
luatexbase.provides_module = provides_module
```

86.11.2 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.

```
local function msg_format(mod, msg_type, text)
  local leader = ""
  local cont
  local first_head
  if mod == "LaTeX" then
    cont = string_gsub(leader, ".", " ")
    first_head = leader .. "LaTeX: "
  else
    first_head = leader .. "Module "  .. msg_type
    cont = "(" .. mod .. ")"
      .. string_gsub(first_head, ".", " ")
    first_head =  leader .. "Module "  .. mod .. " " .. msg_type  .. ":"
  end
  if msg_type == "Error" then
    first_head = "\n" .. first_head
  end
  if string.sub(text,-1) ~= "\n" then
    text = text .. " "
  end
  return first_head .. " "
    .. string_gsub(
         text
 .. "on input line "
         .. tex.inputlineno, "\n", "\n" .. cont .. " "
      )
   .. "\n"
end
```

module_info
module_warning
module_error

Write messages.

```
local function module_info(mod, text)
  texio_write_nl("log", msg_format(mod, "Info", text))
end
luatexbase.module_info = module_info
```

```
local function module_warning(mod, text)
  texio_write_nl("term and log",msg_format(mod, "Warning", text))
end
luatexbase.module_warning = module_warning
local function module_error(mod, text)
  error(msg_format(mod, "Error", text))
end
luatexbase.module_error = module_error
```

Dedicated versions for the rest of the code here.

```
function luatexbase_warning(text)
  module_warning("luatexbase", text)
end
function luatexbase_error(text)
  module_error("luatexbase", text)
end
```

86.12 Accessing register numbers from Lua

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

```
local luaregisterbasetable = { }
local registermap = {
  attributezero = "assign_attr"    ,
  charzero      = "char_given"     ,
  CountZero     = "assign_int"     ,
  dimenzero     = "assign_dimen"   ,
  mathcharzero  = "math_given"     ,
  muskipzero    = "assign_mu_skip" ,
  skipzero      = "assign_skip"    ,
  tokszero      = "assign_toks"    ,
}
local createtoken
if tex.luatexversion > 81 then
  createtoken = token.create
elseif tex.luatexversion > 79 then
  createtoken = newtoken.create
end
local hashtokens    = tex.hashtokens()
local luatexversion = tex.luatexversion
for i,j in pairs (registermap) do
  if luatexversion < 80 then
    luaregisterbasetable[hashtokens[i][1]] =
      hashtokens[i][2]
  else
    luaregisterbasetable[j] = createtoken(i).mode
  end
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.

```
local registernumber
if luatexversion < 80 then
  function registernumber(name)
    local nt = hashtokens[name]
    if(nt and luaregisterbasetable[nt[1]]) then
      return nt[2] - luaregisterbasetable[nt[1]]
    else
      return false
    end
  end
else
  function registernumber(name)
    local nt = createtoken(name)
    if(luaregisterbasetable[nt.cmdname]) then
      return nt.mode - luaregisterbasetable[nt.cmdname]
    else
      return false
    end
  end
end
luatexbase.registernumber = registernumber
```

86.13 Attribute allocation

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

```
local attributes=setmetatable(
{},
{
__index = function(t,key)
return registernumber(key) or nil
end}
)
luatexbase.attributes = attributes
```

```
local attribute_count_name =
                     attribute_count_name or "e@alloc@attribute@count"
local function new_attribute(name)
  tex_setcount("global", attribute_count_name,
                          tex_count[attribute_count_name] + 1)
  if tex_count[attribute_count_name] > 65534 then
    luatexbase_error("No room for a new \\attribute")
  end
  attributes[name]= tex_count[attribute_count_name]
  luatexbase_log("Lua-only attribute " .. name .. " = " ..
                 tex_count[attribute_count_name])
  return tex_count[attribute_count_name]
end
luatexbase.new_attribute = new_attribute
```

86.14 Custom whatsit allocation

new_whatsit Much the same as for attribute allocation in Lua.

```
local whatsit_count_name = whatsit_count_name or "e@alloc@whatsit@count"
local function new_whatsit(name)
  tex_setcount("global", whatsit_count_name,
                         tex_count[whatsit_count_name] + 1)
  if tex_count[whatsit_count_name] > 65534 then
    luatexbase_error("No room for a new custom whatsit")
  end
  luatexbase_log("Custom whatsit " .. (name or "") .. " = " ..
                 tex_count[whatsit_count_name])
  return tex_count[whatsit_count_name]
end
luatexbase.new_whatsit = new_whatsit
```

86.15 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.

```
local bytecode_count_name =
                         bytecode_count_name or "e@alloc@bytecode@count"
local function new_bytecode(name)
  tex_setcount("global", bytecode_count_name,
                         tex_count[bytecode_count_name] + 1)
  if tex_count[bytecode_count_name] > 65534 then
    luatexbase_error("No room for a new bytecode register")
  end
  luatexbase_log("Lua bytecode " .. (name or "") .. " = " ..
                 tex_count[bytecode_count_name])
  return tex_count[bytecode_count_name]
end
luatexbase.new_bytecode = new_bytecode
```

86.16 Lua chunk name allocation

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

```
local chunkname_count_name =
                        chunkname_count_name or "e@alloc@luachunk@count"
local function new_chunkname(name)
  tex_setcount("global", chunkname_count_name,
                         tex_count[chunkname_count_name] + 1)
  local chunkname_count = tex_count[chunkname_count_name]
  chunkname_count = chunkname_count + 1
  if chunkname_count > 65534 then
    luatexbase_error("No room for a new chunkname")
  end
  lua.name[chunkname_count]=name
  luatexbase_log("Lua chunkname " .. (name or "") .. " = " ..
                 chunkname_count .. "\n")
  return chunkname_count
end
luatexbase.new_chunkname = new_chunkname
```

86.17 Lua function allocation

new_luafunction Much the same as for attribute allocation in Lua. The optional ⟨*name*⟩ argument is used in the log if given.

```
local luafunction_count_name =
                         luafunction_count_name or "e@alloc@luafunction@count"
local function new_luafunction(name)
  tex_setcount("global", luafunction_count_name,
                         tex_count[luafunction_count_name] + 1)
  if tex_count[luafunction_count_name] > 65534 then
    luatexbase_error("No room for a new luafunction register")
  end
  luatexbase_log("Lua function " .. (name or "") .. " = " ..
                 tex_count[luafunction_count_name])
  return tex_count[luafunction_count_name]
end
luatexbase.new_luafunction = new_luafunction
```

86.18 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.

86.18.1 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.

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

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

```
local list, data, exclusive, simple, reverselist = 1, 2, 3, 4, 5
local types   = {
  list        = list,
  data        = data,
  exclusive   = exclusive,
  simple      = simple,
  reverselist = reverselist,
}
```

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.)

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

Section 8.2: file discovery callbacks.

```
  find_read_file     = exclusive,
  find_write_file    = exclusive,
  find_font_file     = data,
  find_output_file   = data,
  find_format_file   = data,
  find_vf_file       = data,
  find_map_file      = data,
  find_enc_file      = data,
  find_pk_file       = data,
  find_data_file     = data,
  find_opentype_file = data,
  find_truetype_file = data,
  find_type1_file    = data,
  find_image_file    = data,

  open_read_file     = exclusive,
  read_font_file     = exclusive,
  read_vf_file       = exclusive,
  read_map_file      = exclusive,
  read_enc_file      = exclusive,
  read_pk_file       = exclusive,
  read_data_file     = exclusive,
  read_truetype_file = exclusive,
  read_type1_file    = exclusive,
  read_opentype_file = exclusive,
```

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

```
  find_cidmap_file   = data,
  read_cidmap_file   = exclusive,
```

Section 8.3: data processing callbacks.

```
  process_input_buffer  = data,
  process_output_buffer = data,
  process_jobname       = data,
```

Section 8.4: node list processing callbacks.

```
  contribute_filter      = simple,
  buildpage_filter       = simple,
  build_page_insert      = exclusive,
  pre_linebreak_filter   = list,
  linebreak_filter       = exclusive,
  append_to_vlist_filter = exclusive,
  post_linebreak_filter  = reverselist,
  hpack_filter           = list,
  vpack_filter           = list,
  hpack_quality          = list,
  vpack_quality          = list,
  pre_output_filter      = list,
  process_rule           = exclusive,
  hyphenate             = simple,
  ligaturing            = simple,
  kerning               = simple,
  insert_local_par      = simple,
  pre_mlist_to_hlist_filter = list,
  mlist_to_hlist        = exclusive,
  post_mlist_to_hlist_filter = reverselist,
  new_graf              = simple,
```

Section 8.5: information reporting callbacks.

```
  pre_dump            = simple,
  start_run           = simple,
  stop_run            = simple,
  start_page_number   = simple,
  stop_page_number    = simple,
  show_error_hook     = simple,
  show_warning_message = simple,
  show_error_message  = simple,
  show_lua_error_hook = simple,
  start_file          = simple,
  stop_file           = simple,
  call_edit           = simple,
  finish_synctex      = simple,
  wrapup_run          = simple,
```

Section 8.6: PDF-related callbacks.

```
  finish_pdffile            = data,
  finish_pdfpage            = data,
  page_objnum_provider      = data,
  page_order_index          = data,
  process_pdf_image_content = data,
```

Section 8.7: font-related callbacks.

```
  define_font                     = exclusive,
  glyph_info                      = exclusive,
  glyph_not_found                 = exclusive,
  glyph_stream_provider           = exclusive,
  make_extensible                 = exclusive,
  font_descriptor_objnum_provider = exclusive,

}
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.

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

86.18.2 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.

```
local function data_handler(name)
  return function(data, ...)
    for _,i in ipairs(callbacklist[name]) do
      data = i.func(data,...)
    end
    return data
  end
end
```

Default for user-defined `data` callbacks without explicit default.

```
local function data_handler_default(value)
  return value
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.

```
local function exclusive_handler(name)
  return function(...)
    return callbacklist[name][1].func(...)
  end
end
```

Handler for `list` callbacks.

```
local function list_handler(name)
  return function(head, ...)
    local ret
    local alltrue = true
    for _,i in ipairs(callbacklist[name]) do
      ret = i.func(head, ...)
      if ret == false then
        luatexbase_warning(
          "Function ‘" .. i.description .. "’ returned false\n"
            .. "in callback ‘" .. name .."’"
         )
         break
      end
      if ret ~= true then
        alltrue = false
        head = ret
      end
    end
    return alltrue and true or head
  end
end
```

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

```
local function list_handler_default()
  return true
end
```

Handler for `reverselist` callbacks.

```
local function reverselist_handler(name)
  return function(head, ...)
    local ret
    local alltrue = true
    local callbacks = callbacklist[name]
    for i = #callbacks, 1, -1 do
      local cb = callbacks[i]
      ret = cb.func(head, ...)
      if ret == false then
        luatexbase_warning(
          "Function ‘" .. cb.description .. "’ returned false\n"
            .. "in callback ‘" .. name .."’"
         )
         break
      end
      if ret ~= true then
        alltrue = false
        head = ret
      end
    end
    return alltrue and true or head
  end
end
```

Handler for `simple` callbacks.

```
local function simple_handler(name)
```

```
  return function(...)
    for _,i in ipairs(callbacklist[name]) do
      i.func(...)
    end
  end
end
```

Default for user-defined simple callbacks without explicit default.

```
local function simple_handler_default()
end
```

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

```
local handlers  = {
  [data]        = data_handler,
  [exclusive]   = exclusive_handler,
  [list]        = list_handler,
  [reverselist] = reverselist_handler,
  [simple]      = simple_handler,
}
local defaults = {
  [data]        = data_handler_default,
  [exclusive]   = nil,
  [list]        = list_handler_default,
  [reverselist] = list_handler_default,
  [simple]      = simple_handler_default,
}
```

86.18.3 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.

```
local user_callbacks_defaults = {
  pre_mlist_to_hlist_filter = list_handler_default,
  mlist_to_hlist = node.mlist_to_hlist,
  post_mlist_to_hlist_filter = list_handler_default,
}
```

create_callback The allocator itself.

```
local function create_callback(name, ctype, default)
  local ctype_id = types[ctype]
  if not name  or name  == ""
  or not ctype_id
  then
    luatexbase_error("Unable to create callback:\n" ..
                     "valid callback name and type required")
  end
  if callbacktypes[name] then
    luatexbase_error("Unable to create callback `" .. name ..
                     "':\ncallback is already defined")
  end
  default = default or defaults[ctype_id]
  if not default then
```

```
    luatexbase_error("Unable to create callback ‘" .. name ..
                     "’:\ndefault is required for ‘" .. ctype ..
                     "’ callbacks")
  elseif type (default) ~= "function" then
    luatexbase_error("Unable to create callback ‘" .. name ..
                     "’:\ndefault is not a function")
  end
  user_callbacks_defaults[name] = default
  callbacktypes[name] = ctype_id
end
luatexbase.create_callback = create_callback
```

call_callback

Call a user defined callback. First check arguments.

```
local function call_callback(name,...)
  if not name or name == "" then
    luatexbase_error("Unable to create callback:\n" ..
                     "valid callback name required")
  end
  if user_callbacks_defaults[name] == nil then
    luatexbase_error("Unable to call callback ‘" .. name
                     .. "’:\nunknown or empty")
   end
  local l = callbacklist[name]
  local f
  if not l then
    f = user_callbacks_defaults[name]
  else
    f = handlers[callbacktypes[name]](name)
  end
  return f(...)
end
luatexbase.call_callback=call_callback
```

add_to_callback

Add a function to a callback. First check arguments.

```
local function add_to_callback(name, func, description)
  if not name or name == "" then
    luatexbase_error("Unable to register callback:\n" ..
                     "valid callback name required")
  end
  if not callbacktypes[name] or
    type(func) ~= "function" or
    not description or
    description == "" then
    luatexbase_error(
      "Unable to register callback.\n\n"
        .. "Correct usage:\n"
        .. "add_to_callback(<callback>, <function>, <description>)"
    )
  end
```

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

```
  local l = callbacklist[name]
  if l == nil then
```

```
    l = { }
    callbacklist[name] = l
```

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

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

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

```
  local f = {
    func        = func,
    description = description,
  }
  local priority = #l + 1
  if callbacktypes[name] == exclusive then
    if #l == 1 then
      luatexbase_error(
        "Cannot add second callback to exclusive function\n‘" ..
        name .. "’")
    end
  end
  table.insert(l, priority, f)
```

Keep user informed.

```
  luatexbase_log(
    "Inserting ‘" .. description .. "’ at position "
      .. priority .. " in ‘" .. name .. "’."
  )
end
luatexbase.add_to_callback = add_to_callback
```

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

```
local function remove_from_callback(name, description)
  if not name or name == "" then
    luatexbase_error("Unable to remove function from callback:\n" ..
                     "valid callback name required")
  end
  if not callbacktypes[name] or
    not description or
    description == "" then
    luatexbase_error(
      "Unable to remove function from callback.\n\n"
        .. "Correct usage:\n"
        .. "remove_from_callback(<callback>, <description>)"
    )
  end
  local l = callbacklist[name]
  if not l then
    luatexbase_error(
      "No callback list for ‘" .. name .. "’\n")
  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.

```
  local index = false
  for i,j in ipairs(l) do
    if j.description == description then
      index = i
      break
    end
  end
  if not index then
    luatexbase_error(
      "No callback ‘" .. description .. "’ registered for ‘" ..
      name .. "’\n")
  end
  local cb = l[index]
  table.remove(l, index)
  luatexbase_log(
    "Removing  ‘" .. description .. "’ from ‘" .. name .. "’."
  )
  if #l == 0 then
    callbacklist[name] = nil
    callback_register(name, nil)
  end
  return cb.func,cb.description
end
luatexbase.remove_from_callback = remove_from_callback
```

in_callback

Look for a function description in a callback.

```
local function in_callback(name, description)
  if not name
    or name == ""
    or not callbacklist[name]
    or not callbacktypes[name]
    or not description then
      return false
  end
  for _, i in pairs(callbacklist[name]) do
    if i.description == description then
      return true
    end
  end
  return false
end
luatexbase.in_callback = in_callback
```

disable_callback

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

```
local function disable_callback(name)
  if(callbacklist[name] == nil) then
    callback_register(name, false)
  else
    luatexbase_error("Callback list for " .. name .. " not empty")
  end
end
luatexbase.disable_callback = disable_callback
```

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

```
local function callback_descriptions (name)
  local d = {}
  if not name
    or name == ""
    or not callbacklist[name]
    or not callbacktypes[name]
    then
    return d
  else
  for k, i in pairs(callbacklist[name]) do
    d[k]= i.description
    end
  end
  return d
end
luatexbase.callback_descriptions =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!

```
local function uninstall()
  module_info(
    "luatexbase",
    "Uninstalling kernel luatexbase code"
  )
  callback.register = callback_register
  luatexbase = nil
end
luatexbase.uninstall = uninstall
```

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

```
callback_register("mlist_to_hlist", function(head, display_type, need_penalties)
  local current = call_callback("pre_mlist_to_hlist_filter", head, display_type, need_penalti
  if current == false then
    flush_list(head)
    return nil
  elseif current == true then
    current = head
  end
  current = call_callback("mlist_to_hlist", current, display_type, need_penalties)
  local post = call_callback("post_mlist_to_hlist_filter", current, display_type, need_penalt
  if post == true then
    return current
  elseif post == false then
    flush_list(current)
    return nil
  end
  return post
end)
```

⟨/lua⟩

Reset the catcode of `@`.

```
⟨tex⟩\catcode`\@=\etatcatcode\relax
```

File Q
ltfinal.dtx

87 Final settings

This section contains the final settings for LaTeX. It initialises some debugging and typesetting parameters, sets the default \catcodes and uc/lc codes, and inputs the hyphenation file.

87.1 Debugging

By default, LaTeX shows statistics:

```
⟨*2ekernel⟩
\tracingstats1
```

87.2 Typesetting parameters

\@lowpenalty
\@medpenalty
\@highpenalty

These are penalties used internally.

```
\newcount\@lowpenalty
\newcount\@medpenalty
\newcount\@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.

```
⟨/2ekernel⟩
⟨*2ekernel | latexrelease⟩
⟨latexrelease⟩\IncludeInRelease{2015/01/01}%
⟨latexrelease⟩                 {\newmarks}{Extended Allocation}%
\ifx\marks\@undefined\else
\def\newmarks{%
  \e@alloc\marks \e@alloc@chardef{\count256}\m@ne\e@alloc@top}
\fi
⟨/2ekernel | latexrelease⟩
⟨latexrelease⟩\EndIncludeInRelease
⟨latexrelease⟩\IncludeInRelease{0000/00/00}%
⟨latexrelease⟩                 {\newmarks}{Extended Allocation}%
⟨latexrelease⟩\let\newmarks\@undefined
⟨latexrelease⟩\EndIncludeInRelease
⟨*2ekernel⟩
```

\newXeTeXintercharclass
\xe@alloc@intercharclass
\e@alloc@intercharclass@top

Allocate \XeTeXintercharclass types if xetex is active. previously defined in xetex.ini.

```
⟨/2ekernel⟩
⟨*2ekernel | latexrelease⟩
⟨latexrelease⟩\IncludeInRelease{2015/01/01}%
⟨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).

```
\ifx\XeTeXcharclass\@undefined
\else
```

```
\ifdim\the\XeTeXversion\XeTeXrevision\p@>0.99993\p@
  \chardef\e@alloc@intercharclass@top=4095
\else
  \chardef\e@alloc@intercharclass@top=255
\fi
\def\newXeTeXintercharclass{%
 \e@alloc\XeTeXcharclass
   \chardef\xe@alloc@intercharclass\m@ne\e@alloc@intercharclass@top}
\fi
⟨/2ekernel | latexrelease⟩
⟨latexrelease⟩\EndIncludeInRelease
⟨latexrelease⟩\IncludeInRelease{0000/00/00}%
⟨latexrelease⟩              {\newXeTeXintercharclass}{Extended Allocation}%
⟨latexrelease⟩ \ifx\XeTeXcharclass\@undefined
⟨latexrelease⟩ \else
⟨latexrelease⟩    \def\xe@alloc@#1#2#3#4#5{\global\advance#1\@ne
⟨latexrelease⟩     \xe@ch@ck#1#4#2%
⟨latexrelease⟩     \allocationnumber#1%
⟨latexrelease⟩     \global#3#5\allocationnumber
⟨latexrelease⟩     \wlog{\string#5=\string#2\the\allocationnumber}}
⟨latexrelease⟩    \def\xe@ch@ck#1#2#3{%
⟨latexrelease⟩     \ifnum#1<#2\else
⟨latexrelease⟩      \errmessage{No room for a new #3}%
⟨latexrelease⟩     \fi}
⟨latexrelease⟩    \def\newXeTeXintercharclass{%
⟨latexrelease⟩     \xe@alloc@\xe@alloc@intercharclass
⟨latexrelease⟩                    \XeTeXcharclass\chardef\@cclv}
⟨latexrelease⟩ \fi
⟨latexrelease⟩\EndIncludeInRelease
⟨*2ekernel | latexrelease⟩
⟨latexrelease⟩\IncludeInRelease{2016/02/01}%
⟨latexrelease⟩  {\xe@alloc@intercharclass}{Start of XeTeX class allocator}%
\ifx\XeTeXcharclass\@undefined
\else
  \countdef\xe@alloc@intercharclass=257
  \xe@alloc@intercharclass=\z@
\fi
⟨/2ekernel | latexrelease⟩
⟨latexrelease⟩\EndIncludeInRelease
⟨latexrelease⟩\IncludeInRelease{2015/01/01}%
⟨latexrelease⟩  {\xe@alloc@intercharclass}{Start of XeTeX class allocator}%
⟨latexrelease⟩ \ifx\XeTeXcharclass\@undefined
⟨latexrelease⟩ \else
⟨latexrelease⟩   \xe@alloc@intercharclass=\thr@@
⟨latexrelease⟩ \fi
⟨latexrelease⟩\EndIncludeInRelease
⟨latexrelease⟩\IncludeInRelease{0000/00/00}%
⟨latexrelease⟩  {\xe@alloc@intercharclass}{Start of XeTeX class allocator}%
⟨latexrelease⟩ \ifx\XeTeXcharclass\@undefined
⟨latexrelease⟩ \else
⟨latexrelease⟩   \newcount\xe@alloc@intercharclass
⟨latexrelease⟩   \xe@alloc@intercharclass=\thr@@
⟨latexrelease⟩ \fi
```

```
⟨latexrelease⟩\EndIncludeInRelease
⟨*2ekernel⟩
```

The default values of the picture and `\fbox` parameters:

```
\unitlength = 1pt
\fboxsep = 3pt
\fboxrule = .4pt
```

The saved value of TEX's `\maxdepth`:

```
\@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}`

```
\vsize = 1000pt
\@colroom = \vsize
\@colht = \vsize
```

Initialise `\textheight` `\textwidth` and page style, to avoid internal errors if they are not set by the class.

```
\textheight=.5\maxdimen
\textwidth=\textheight
\ps@empty
```

87.3 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: XeTEX follows the standard TEX route of building patterns into the format. LuaTEX 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`.

```
\ifnum 0%
  \ifx\Umathcode\@undefined\else 1\fi
  \ifx\XeTeXmathcode\@undefined\else 1\fi
  >\z@
  \message{ Unicode character data,}
  \input{load-unicode-data}
⟨/2ekernel⟩
⟨latexrelease⟩\IncludeInRelease{2016/02/01}%
⟨latexrelease⟩  {\XeTeXintercharclasses}{XeTeX character classes}%
⟨latexrelease⟩  \ifx\XeTeXinterchartoks\undefined
⟨latexrelease⟩  \else
⟨latexrelease⟩    \begingroup
⟨latexrelease⟩      \chardef\XeTeXcharclassID = 0 %
⟨latexrelease⟩      \chardef\XeTeXcharclassOP = 0 %
⟨latexrelease⟩      \chardef\XeTeXcharclassCL = 0 %
⟨latexrelease⟩      \chardef\XeTeXcharclassEX = 0 %
⟨latexrelease⟩      \chardef\XeTeXcharclassIS = 0 %
```

```
⟨latexrelease⟩      \chardef\XeTeXcharclassNS = 0 %
⟨latexrelease⟩      \chardef\XeTeXcharclassCM = 0 %
⟨latexrelease⟩      \input{load-unicode-xetex-classes}
⟨latexrelease⟩    \endgroup
⟨latexrelease⟩    \global\let\xtxHanGlue\undefined
⟨latexrelease⟩    \global\let\xtxHanSpace\undefined
⟨latexrelease⟩    \global\XeTeXinterchartoks 0 1 = {}
⟨latexrelease⟩    \global\XeTeXinterchartoks 0 2 = {}
⟨latexrelease⟩    \global\XeTeXinterchartoks 0 3 = {}
⟨latexrelease⟩    \global\XeTeXinterchartoks 1 0 = {}
⟨latexrelease⟩    \global\XeTeXinterchartoks 2 0 = {}
⟨latexrelease⟩    \global\XeTeXinterchartoks 3 0 = {}
⟨latexrelease⟩    \global\XeTeXinterchartoks 1 1 = {}
⟨latexrelease⟩    \global\XeTeXinterchartoks 1 2 = {}
⟨latexrelease⟩    \global\XeTeXinterchartoks 1 3 = {}
⟨latexrelease⟩    \global\XeTeXinterchartoks 2 1 = {}
⟨latexrelease⟩    \global\XeTeXinterchartoks 2 2 = {}
⟨latexrelease⟩    \global\XeTeXinterchartoks 2 3 = {}
⟨latexrelease⟩    \global\XeTeXinterchartoks 3 1 = {}
⟨latexrelease⟩    \global\XeTeXinterchartoks 3 2 = {}
⟨latexrelease⟩    \global\XeTeXinterchartoks 3 3 = {}
⟨latexrelease⟩  \fi
⟨latexrelease⟩\EndIncludeInRelease
⟨latexrelease⟩\IncludeInRelease{0000/00/00}%
⟨latexrelease⟩  {\XeTeXintercharclasses}{XeTeX character classes}%
⟨latexrelease⟩  \ifx\XeTeXinterchartoks\undefined
⟨latexrelease⟩  \else
⟨latexrelease⟩   \input{load-unicode-xetex-classes}
⟨latexrelease⟩   \gdef\xtxHanGlue{\hskip0pt plus 0.1em\relax}
⟨latexrelease⟩   \gdef\xtxHanSpace{\hskip0.2em plus 0.2em minus 0.1em\relax}
⟨latexrelease⟩   \global\XeTeXinterchartoks 0 1 = {\xtxHanSpace}
⟨latexrelease⟩   \global\XeTeXinterchartoks 0 2 = {\xtxHanSpace}
⟨latexrelease⟩   \global\XeTeXinterchartoks 0 3 = {\nobreak\xtxHanSpace}
⟨latexrelease⟩   \global\XeTeXinterchartoks 1 0 = {\xtxHanSpace}
⟨latexrelease⟩   \global\XeTeXinterchartoks 2 0 = {\nobreak\xtxHanSpace}
⟨latexrelease⟩   \global\XeTeXinterchartoks 3 0 = {\xtxHanSpace}
⟨latexrelease⟩   \global\XeTeXinterchartoks 1 1 = {\xtxHanGlue}
⟨latexrelease⟩   \global\XeTeXinterchartoks 1 2 = {\xtxHanGlue}
⟨latexrelease⟩   \global\XeTeXinterchartoks 1 3 = {\nobreak\xtxHanGlue}
⟨latexrelease⟩   \global\XeTeXinterchartoks 2 1 = {\nobreak\xtxHanGlue}
⟨latexrelease⟩   \global\XeTeXinterchartoks 2 2 = {\nobreak\xtxHanGlue}
⟨latexrelease⟩   \global\XeTeXinterchartoks 2 3 = {\xtxHanGlue}
⟨latexrelease⟩   \global\XeTeXinterchartoks 3 1 = {\xtxHanGlue}
⟨latexrelease⟩   \global\XeTeXinterchartoks 3 2 = {\xtxHanGlue}
⟨latexrelease⟩   \global\XeTeXinterchartoks 3 3 = {\nobreak\xtxHanGlue}
⟨latexrelease⟩  \fi
⟨latexrelease⟩\EndIncludeInRelease
⟨*2ekernel⟩
```

There is one over-ride that makes sense here (see below for the same for 8-bit engines): setting the lccode for - to itself.

```
  \lccode`\- =`\- % default hyphen char
```

The alternative is that a “traditional” engine is in use.

```
\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.

```
\def\reserved@a#1#2{%
   \@tempcnta#1\relax
   \@tempcntb#2\relax
   \reserved@b
}
\def\reserved@b{%
   \ifnum\@tempcnta>\@tempcntb\else
      \reserved@c\@tempcnta
      \advance\@tempcnta\@ne
      \expandafter\reserved@b
   \fi
}
```

Depending on the TEX version, we might not be allowed to do this for non-ASCII characters.

```
\def\reserved@c#1{%
   \count@=#1\advance\count@ by -"20
   \uccode#1=\count@
   \lccode#1=#1
}
\reserved@a{`\a}{`\z}
\reserved@a{"A0}{"BC}
\reserved@a{"E0}{"FF}
```

The upper case characters need their `\uccode` and `\lccode` values set, and their `\sfcode` set to 999.

```
\def\reserved@c#1{%
   \count@=#1\advance\count@ by "20
   \uccode#1=#1
   \lccode#1=\count@
   \sfcode#1=999
}
\reserved@a{`\A}{`\Z}
\reserved@a{"80}{"9C}
\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.

```
\uccode`\^^Y=`\I     % dotless i
\lccode`\^^Y=`\^^Y   % dotless i
\uccode`\^^Z=`\J     % dotless j, ae in OT1
\lccode`\^^Z=`\^^Z   % dotless j, ae in OT1
\lccode`\^^9d=`\i    % dotted I
\uccode`\^^9d=`\^^9d % dotted I
\lccode`\^^9e=`\^^9e % d-bar
\uccode`\^^9e=`\^^d0 % d-bar
```

Finally here is one that helps hyphenation in the OT1 encoding.

```
\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).

```
\lccode`\- =`\-   % default hyphen char
\lccode 127=127   % alternate hyphen char
\lccode 23 =23    % textcompwordmark in T1
```

End of the conditional to select either Unicode or T1 encoding defaults.

```
\fi
```

This is as good a place as any to active a few XeTEX-specific settings

```
\ifx\XeTeXuseglyphmetrics\@undefined
\else
  \XeTeXuseglyphmetrics=1 %
  \XeTeXdashbreakstate=1 %
\fi
```

87.4 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 `\catcode`s are set, so local hyphenation files can use 8-bit input.

We try to load the customized hyphenation description file.

```
\InputIfFileExists{hyphen.cfg}
           {\typeout{===========================================^^J%
                      Local configuration file hyphen.cfg used^^J%
                     ===========================================}%
             \def\@addtofilelist##1{\xdef\@filelist{\@filelist,##1}}%
           }
           {\input{hyphen.ltx}}
\let\@addtofilelist\@gobble
```

`\l@nohyphenation`

```
\ifx\l@nohyphenation \@undefined
  \newlanguage\l@nohyphenation
\fi
```

`\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.

```
⟨/2ekernel⟩
⟨*2ekernel | latexrelease⟩
⟨latexrelease⟩\IncludeInRelease{2017/04/15}%
⟨latexrelease⟩                 {\document@default@language}{Save language for hyphenation}%
\let\document@default@language\m@ne
⟨/2ekernel | latexrelease⟩
⟨latexrelease⟩\EndIncludeInRelease
⟨latexrelease⟩\IncludeInRelease{0000/00/00}%
⟨latexrelease⟩                 {\document@default@language}{Save language for hyphenation}%
⟨latexrelease⟩\let\document@default@language\@undefined
⟨latexrelease⟩\EndIncludeInRelease
⟨*2ekernel⟩
```

87.5 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.

```
\ifdim \font@submax >\z@
   \@font@warning{Size substitutions with differences\MessageBreak
                 up to \font@submax\space have occurred.\MessageBreak
                \MessageBreak
                Please check the transcript file
                carefully\MessageBreak
                and redo the format generation if necessary!
                \@gobbletwo}%
   \errhelp{Only stopped, to give you time to
            read the above message.}
   \errmessage{}
```

We reset the macro. Otherwise every user will get a warning on every job.

```
\def\font@submax{0pt}
\fi
```

87.6 Input encoding

Starting with the 2018 LaTeX release default the inputencoding to UTF-8. Unless the format is being used with luatex, xetex, enctex or mltex.

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.

```
⟨/2ekernel⟩
⟨*2ekernel | latexrelease⟩
```

Check that a classic 8-bit tex engine is being used (LaTeX or PDFLaTeX).

```
⟨latexrelease⟩\IncludeInRelease{2018/04/01}%
⟨latexrelease⟩                 {\UTFviii@invalid}{UTF-8 default}%
```

Skip this section in Unicode TeX, or if MLTeX and EncTeX are enabled.

```
\ifnum0%
  \ifx\Umathchar\@undefined\else 1\fi
  \ifx\mubyte\@undefined\else 1\fi
  \ifx\charsubdef\@undefined\else 1\fi
  =\z@
```

```
\def\saved@space@catcode{10}
\let\@inpenc@test\relax
\def\IeC{%
  \ifx\protect\@typeset@protect
    \expandafter\@firstofone
  \else
    \noexpand\IeC
  \fi
}
```

Make characters active for UTF-8 input formats

```
\@tempcnta=1
\loop
  \catcode\@tempcnta=13  %
  \advance\@tempcnta\@ne %
\ifnum\@tempcnta<32      %
\repeat                  %
\catcode0=15  % null
\catcode9=10  % tab
\catcode10=12 % ctrl J
\catcode12=13 % ctrl L
\catcode13=5  % newline
\@tempcnta=128
\loop
  \catcode\@tempcnta=13
  \advance\@tempcnta\@ne
\ifnum\@tempcnta<256
\repeat
```

`\UseRawInputEncoding` Reset 8 bit characters to catcode 12 so the input endcoing matches the "Raw" font encoding. Useful for special behaviours, or for compatibility with older LaTeX formats.

```
\def\UseRawInputEncoding{%
\let\inputencodingname\@undefined                     % revert
\let\DeclareFontEncoding@\DeclareFontEncoding@saved   % revert
\let\DeclareUnicodeCharacter\@undefined               % revert
\@tempcnta=1
\loop
  \catcode\@tempcnta=15  %
  \advance\@tempcnta\@ne %
\ifnum\@tempcnta<32      %
\repeat                  %
\catcode0=15  % null
\catcode9=10  % tab
\catcode10=12 % ctrl J
\catcode12=13 % ctrl L
\catcode13=5  % newline
\@tempcnta=128
\loop
  \catcode\@tempcnta=12
  \advance\@tempcnta\@ne
\ifnum\@tempcnta<256
\repeat
}
```

`\DeclareFontEncoding@saved` Saved version of `\DeclareFontEncoding@` before `utf8.def` modifies it for use in `\UseRawInputEncoding` above.

```
\let\DeclareFontEncoding@saved\DeclareFontEncoding@
```

```
\edef\inputencodingname{utf8}%
\input{utf8.def}
\let\UTFviii@undefined@err@@\UTFviii@undefined@err
\let\UTFviii@invalid@err@@\UTFviii@invalid@err
```

```
\let\UTFviii@two@octets@@\UTFviii@two@octets
\let\UTFviii@three@octets@@\UTFviii@three@octets
\let\UTFviii@four@octets@@\UTFviii@four@octets
⟨2ekernel⟩\def\UTFviii@undefined@err#1{\@gobble#1}%
⟨2ekernel⟩\let\UTFviii@invalid@err\string
⟨2ekernel⟩\let\UTFviii@two@octets\string
⟨2ekernel⟩\let\UTFviii@three@octets\string
⟨2ekernel⟩\let\UTFviii@four@octets\string
⟨2ekernel⟩\everyjob\expandafter{\the\everyjob
⟨2ekernel⟩\let\UTFviii@undefined@err\UTFviii@undefined@err@@
⟨2ekernel⟩\let\UTFviii@invalid@err\UTFviii@invalid@err@@
⟨2ekernel⟩\let\UTFviii@two@octets\UTFviii@two@octets@@
⟨2ekernel⟩\let\UTFviii@three@octets\UTFviii@three@octets@@
⟨2ekernel⟩\let\UTFviii@four@octets\UTFviii@four@octets@@
⟨2ekernel⟩}
\let\@inpenc@test\@undefined
\let\saved@space@catcode\@undefined
```

For formats not set up for UTF-8 default, set the C0 controls to catcode 15.

```
\else
\@tempcnta=0
\loop
  \catcode\@tempcnta=15  %
  \advance\@tempcnta\@ne %
\ifnum\@tempcnta<32      %
\repeat                  %
\catcode0=15  % null
\catcode9=10  % tab
\catcode10=12 % ctrl J
\catcode12=13 % ctrl L
\catcode13=5  % newline
```

```
\let\UseRawInputEncoding\relax
```

This ends the skipped code in Unicode engines:

```
\fi
⟨/2ekernel | latexrelease⟩
⟨latexrelease⟩\EndIncludeInRelease
⟨latexrelease⟩\IncludeInRelease{0000/00/00}%
⟨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.

```
⟨latexrelease⟩  \let\UTFviii@two@octets@combine\@undefined
⟨latexrelease⟩  \let\UTFviii@three@octets@combine\@undefined
⟨latexrelease⟩  \let\UTFviii@four@octets@combine\@undefined
⟨latexrelease⟩  \let\UTFviii@two@octets@string\@undefined
⟨latexrelease⟩  \let\UTFviii@three@octets@string\@undefined
⟨latexrelease⟩  \let\UTFviii@four@octets@string\@undefined
⟨latexrelease⟩  \let\UTFviii@two@octets@noexpand\@undefined
⟨latexrelease⟩  \let\UTFviii@three@octets@noexpand\@undefined
⟨latexrelease⟩  \let\UTFviii@four@octets@noexpand\@undefined
```

```
⟨latexrelease⟩\@tempcnta=0
⟨latexrelease⟩\loop
⟨latexrelease⟩  \catcode\@tempcnta=15
```

```
⟨latexrelease⟩  \advance\@tempcnta\@ne
⟨latexrelease⟩\ifnum\@tempcnta<32
⟨latexrelease⟩\repeat       %
⟨latexrelease⟩\catcode9=10  % tab
⟨latexrelease⟩\catcode10=12 % ctrl J
⟨latexrelease⟩\catcode12=13 % ctrl L
⟨latexrelease⟩\catcode13=5  % newline
⟨latexrelease⟩\@tempcnta=128
⟨latexrelease⟩\loop
⟨latexrelease⟩\catcode\@tempcnta=12
⟨latexrelease⟩\advance\@tempcnta\@ne
⟨latexrelease⟩\ifnum\@tempcnta<256
⟨latexrelease⟩\repeat
⟨latexrelease⟩\let\IeC\@undefined
⟨latexrelease⟩\def\DeclareFontEncoding@#1#2#3{%
⟨latexrelease⟩  \expandafter
⟨latexrelease⟩  \ifx\csname T@#1\endcsname\relax
⟨latexrelease⟩     \def\cdp@elt{\noexpand\cdp@elt}%
⟨latexrelease⟩     \xdef\cdp@list{\cdp@list\cdp@elt{#1}%
⟨latexrelease⟩                    {\default@family}{\default@series}%
⟨latexrelease⟩                    {\default@shape}}%
⟨latexrelease⟩     \expandafter\let\csname#1-cmd\endcsname\@changed@cmd
⟨latexrelease⟩  \else
⟨latexrelease⟩     \@font@info{Redeclaring font encoding #1}%
⟨latexrelease⟩  \fi
⟨latexrelease⟩  \global\@namedef{T@#1}{#2}%
⟨latexrelease⟩  \global\@namedef{M@#1}{\default@M#3}%
⟨latexrelease⟩  \xdef\LastDeclaredEncoding{#1}%
⟨latexrelease⟩  }
⟨latexrelease⟩  \let\UseRawInputEncoding\@undefined
⟨latexrelease⟩  \let\DeclareFontEncoding@saved\@undefined
⟨latexrelease⟩  \let\inputencodingname\@undefined
⟨latexrelease⟩\EndIncludeInRelease
```

```
⟨*2ekernel⟩
%    \begin{macrocode}
%
% We temporarily define |\reserved@a| to apply |\reserved@c| to all the
% numbers in the range of its arguments.
%    \begin{macrocode}
\def\reserved@a#1#2{%
   \@tempcnta#1\relax
   \@tempcntb#2\relax
   \reserved@b
}
\def\reserved@b{%
   \ifnum\@tempcnta>\@tempcntb\else
      \reserved@c\@tempcnta
      \advance\@tempcnta\@ne
      \expandafter\reserved@b
   \fi
}
```

Set the special catcodes (although some of these are useless, since an error will have occurred if the catcodes have changed). Note that `^^J` has catcode ‘other’ for use in warning messages.

```
\catcode`\ =10
\catcode`\#=6
\catcode`\$=3
\catcode`\%=14
\catcode`\&=4
\catcode`\\=0
\catcode`\^=7
\catcode`\_=8
\catcode`\{=1
\catcode`\}=2
\catcode`\~=13
\catcode`\@=11
\catcode`\^^I=10
\catcode`\^^J=12
\catcode`\^^L=13
\catcode`\^^M=5
```

Set the ‘other’ catcodes.

```
\def\reserved@c#1{\catcode#1=12\relax}
\reserved@c{`\!}
\reserved@c{`\"}
\reserved@a{`\'}{`\?}
\reserved@c{`\[}
\reserved@c{`\]}
\reserved@c{`\`}
\reserved@c{`\|}
```

Set the ‘letter’ catcodes.

```
\def\reserved@c#1{\catcode#1=11\relax}
\reserved@a{`\A}{`\Z}
\reserved@a{`\a}{`\z}
```

All the characters in the range 0–31 and 127–255 are illegal, *except* tab (`^^I`), nl (`^^J`), ff (`^^L`) and cr (`^^M`).

87.7 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 TEX version, we might not be allowed to do this for non-ASCII characters. For the Unicode engines (XeTEX and LuaTEX) 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.

```
\ifnum 0%
  \ifx\Umathcode\@undefined\else 1\fi
  \ifx\XeTeXmathcode\@undefined\else 1\fi
  >\z@
\else
\def\reserved@c#1{%
   \count@=#1\advance\count@ by -"20
   \uccode#1=\count@
   \lccode#1=#1
}
```

```
\reserved@a{`\a}{`\z}
\reserved@a{"A0}{"BC}
\reserved@a{"E0}{"FF}
```

The upper case characters need their `\uccode` and `\lccode` values set, and their `\sfcode` set to 999.

```
\def\reserved@c#1{%
   \count@=#1\advance\count@ by "20
   \uccode#1=#1
   \lccode#1=\count@
   \sfcode#1=999
}
\reserved@a{`\A}{`\Z}
\reserved@a{"80}{"9C}
\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.

```
\uccode`\^^Y=`\I     % dotless i
\lccode`\^^Y=`\^^Y   % dotless i
\uccode`\^^Z=`\J     % dotless j, ae in OT1
\lccode`\^^Z=`\^^Z   % dotless j, ae in OT1
\lccode`\^^9d=`\i    % dotted I
\uccode`\^^9d=`\^^9d % dotted I
\lccode`\^^9e=`\^^9e % d-bar
\uccode`\^^9e=`\^^d0 % d-bar
```

Finally here is one that helps hyphenation in the OT1 encoding.

```
\lccode`\^^[=`\^^[   % oe in OT1
\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.

```
\DeclareRobustCommand{\MakeUppercase}[1]{{%
      \def\i{I}\def\j{J}%
      \def\reserved@a##1##2{\let##1##2\reserved@a}%
      \expandafter\reserved@a\@uclclist\reserved@b{\reserved@b\@gobble}%
```

Tell UTF-8 processing to process chars even though we are in an `\protected@edef`.

```
      \let\UTF@two@octets@noexpand\@empty
      \let\UTF@three@octets@noexpand\@empty
      \let\UTF@four@octets@noexpand\@empty
      \protected@edef\reserved@a{\uppercase{#1}}%
      \reserved@a
   }}
\DeclareRobustCommand{\MakeLowercase}[1]{{%
      \def\reserved@a##1##2{\let##2##1\reserved@a}%
      \expandafter\reserved@a\@uclclist\reserved@b{\reserved@b\@gobble}%
      \let\UTF@two@octets@noexpand\@empty
      \let\UTF@three@octets@noexpand\@empty
      \let\UTF@four@octets@noexpand\@empty
      \protected@edef\reserved@a{\lowercase{#1}}%
      \reserved@a
   }}
\def\@uclclist{\oe\OE\o\O\ae\AE
      \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:

```
\protected@edef\MakeUppercase#1{\MakeUppercase{#1}}
\protected@edef\MakeLowercase#1{\MakeLowercase{#1}}
```

87.8 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.

```
%\IfFileExists{ltpatch.ltx}
%  {\typeout{=================================^^J%
%             Applying patch file ltpatch.ltx^^J%
%            =================================}
%   \def\fmtversion@topatch{unknown}
%   \input{ltpatch.ltx}
%   \ifx\fmtversion\fmtversion@topatch
%      \ifx\patch@level\@undefined
%        \typeout{^^J^^J^^J%
%         !!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!^^J%
%         !! Patch file ‘ltpatch.ltx’ not suitable for this^^J%
%         !! version of LaTeX.^^J^^J%
%         !! Please check if initex found an old patch file:^^J%
%         !! --- if so, rename it or delete it, and redo the^^J%
%         !! initex run.^^J%
%         !!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!^^J}%
%        \batchmode \@@end
%      \else
```

The code below adds the ‘patch level’ string to the first `\typeout` in the startup banner.

```
%        \def\fmtversion@topatch{0}%
%        \ifx\fmtversion@topatch\patch@level\else
%          \def\reserved@a\typeout##1##2\reserved@a{%
%                 \typeout{##1 patch level \patch@level}##2}
%          \everyjob\expandafter\expandafter\expandafter{%
%             \expandafter\reserved@a\the\everyjob\reserved@a}
%          \let\reserved@a\relax
%          \the\everyjob
%        \fi
%      \fi
%   \else
%      \typeout{^^J^^J^^J%
%    !!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!^^J%
%    !! Patch file ‘ltpatch.ltx’ (for version <\fmtversion@topatch>)^^J%
%    !! is not suitable for version <\fmtversion> of LaTeX.^^J^^J%
%    !! Please check if initex found an old patch file:^^J%
%    !! --- if so, rename it or delete it, and redo the^^J%
%    !!     initex run.^^J%
%    !!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!^^J}%
%      \batchmode \@@end
%   \fi
%   \let\fmtversion@topatch\relax
%  }{}
```

87.9 Loading further extensions

Simply load further code: it’s deliberately separate

```
\input{ltexpl.ltx}
```

87.10 Freeing Memory

\reserved@a
\reserved@b

And just to make sure nobody relies on those definitions of \reserved@b and friends. These macros are reserved for use in the kernel. *Do not use them as general scratch macros.*

```
\let\reserved@a\@filelist
\let\reserved@b=\@undefined
\let\reserved@c=\@undefined
\let\reserved@d=\@undefined
\let\reserved@e=\@undefined
\let\reserved@f=\@undefined
```

\toks

```
\toks0{}
\toks2{}
\toks4{}
\toks6{}
\toks8{}
```

\errhelp

Empty the error help message, which may have some rubbish:

```
\errhelp{}
```

87.11 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`.

```
\def\@providesfile#1[#2]{%
    \wlog{File: #1 #2}%
    \expandafter\xdef\csname ver@#1\endcsname{#2}%
  \endgroup}
```

\@filelist
\@addtofilelist

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 LaTeX 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.

```
\let\@filelist\@gobble
\def\@addtofilelist#1{\xdef\@filelist{\@filelist,#1}}%
```

87.12 Some last minute initializations ...

This initializes the 2020/02/02 extensions to NFSS after any changes in the preamble.

```
\AtBeginDocument{\reinstall@nfss@defs\init@series@setup}
```

87.13 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 ...

```
%\input ...
```

87.14 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.

```
\makeatother
\errorstopmode
\dump
⟨/2ekernel⟩
```

Change History

1985-11-04 ltmath.dtx LaTeX2.09

General: produce warning message if line extends into margin. Doesn't warn about formula overprinting equation number. 356

1989-04-10 ltfssbas.dtx v1.0a

General: Starting with version numbers! `\ifmmode` added in `\math@group` 154

1989-04-10 ltfssbas.dtx v1.0b

General: `\preload@sizes` added. 154

`\wrong@fontshape` changed to define substitution font/shape macro. 154

1989-04-10 ltfssini.dtx v1.0a

General: Starting with version numbers `\newif` for `\@tempswa` added since this switch is unknown at the time when this file is read in. (latex.tex is loaded later.) `\math@famname` changed to `\math@version`. . 247

1989-04-14 ltfssbas.dtx v1.0c

General: More documentation added. 154

1989-04-15 ltfssini.dtx v1.0b

General: `\mathfontset` renamed to `\mathversion`. 247

1989-04-19 ltfssbas.dtx v1.0d

General: Even more doc. 154

1989-04-21 ltfssbas.dtx v1.0e

General: Documentation is fun! Parameters of `\define@mathalphabet` changed. 154

1989-04-21 ltfssini.dtx v1.0c

General: Changed to conform to fam.tex. 247

1989-04-23 ltfssbas.dtx v1.0f

General: % in `\getanddefinefonts` added. 154

1989-04-26 ltfssini.dtx v1.0d

General: `\xpt` added. 247

1989-04-27 ltfssbas.dtx v1.0g

General: Documentation revised. 154

1989-04-27 ltfssini.dtx v1.0e

General: Definitions of LaTeX symbols corrected. 247

1989-04-29 ltfssbas.dtx v1.0h

General: Documented problem with `\halign`, and `\noalign` 154

`\mathversion`: Test if version defined added. 164

1989-04-29 ltfssbas.dtx v1.0i

General: Removed the `\halign` `\noalign` correction (wasn't bugfree) 154

1989-04-29 ltfssini.dtx v1.0f

General: Corrections to LaTeX tabular env. added. 247

1989-05-01 ltfssbas.dtx v1.0j

General: Default for `\baselinestretch` added. . . 154

1989-05-22 ltfssbas.dtx v1.0k

General: Lines longer than 72 characters folded. 154

1989-05-22 ltfssini.dtx v1.0g

General: Lines shortened to 72 characters 247

1989-09-14 ltfssbas.dtx v1.0m

General: Global replacement: `\group` to `\mathgroup` 154

`\mathversion`: Corrected typo: `\endscname` to `\endcsname`. . 164

1989-11-07 ltfssini.dtx v1.0i

General: All family, series, and shape names abbreviated. . . 247

1989-11-08 ltfssbas.dtx v1.0o

General: First parameter of `\define@mathalphabet` and `\define@mathgroup` changed from string to control sequence. 154

1989-11-14 ltfssbas.dtx v1.0p

`\math@version`: Math version prefix 'mv@' added. 164

1989-11-19 ltfssbas.dtx v1.0q

`\define@newfont`: Group added. 165

`\wrong@fontshape`: Instead of calling `\family\default@family`, etc. we directly set `\f@family`, etc. 170

1989-11-22 ltfssbas.dtx v1.0r

`\math@version`: `\def` → `\edef` for `\math@version`. 164

1989-11-25 ltfssbas.dtx v1.0s

General: All `\edef\font@name` changed to `\xdef\font@name`.

Necessary after introduction of \begingroup/\endgroup in v1.0q. 154
extra// → + in \extra@def. . 154

1989-11-26 ltfssbas.dtx v1.0t
\select@group: \bgroup/\egroup changed to \begingroup/\endgroup to avoid empty Ord atom on math list. 172

1989-12-02 ltfssini.dtx v1.1b
General: \rmmath renamed to \mathrm 247

1989-12-03 ltfssini.dtx v1.1c
General: Some internal macros renamed to make them inaccessible. 247

1989-12-05 ltfssbas.dtx v1.0u
\addto@hook: \addto@hook added. 177

1989-12-05 ltfsstrc.dtx v1.0u fam.dtx
\every@math@size: Hook \every@size added. 202

1989-12-13 ltfsstrc.dtx v1.0f
\use@mathgroup: \expandafter added before final \fi. 205

1989-12-16 ltfssbas.dtx v1.1a
\select@group: \relax in front added. 172
Now four arguments. 172
Redefinition of alphabet now simpler. 173
Usage of '=' macro added. . . . 173

1989-12-16 ltfsstrc.dtx v1.1a
\selectfont: Changed order of calls. 199
\use@mathgroup: Redefinition of alphabet now simpler. 205
Usage of '=' macro added. . . . 205

1990-01-18 ltfsstrc.dtx v1.0h
General: \tracingfonts meaning changed. 195

1990-01-20 ltfssbas.dtx v1.2a
\math@bgroup: Def. placed in this file. 174
\math@egroup: Def. placed in this file. 174
\select@group: Def for alph id changed. 173

1990-01-21 ltfssbas.dtx v1.2b
\select@group: Code moved to \use@mathgroup. 173

1990-01-21 ltfsstrc.dtx v1.2b
\use@mathgroup: Macro added to allow cleaner interface. 205

1990-01-23 ltfssbas.dtx v1.2c
General: \no@version@warning renamed to \no@alphabet@error. 154
Macro \no@alphabet@help added 154
\no@alphabet@error: Changed to error call 154

1990-01-25 ltfssini.dtx v1.1e
\nfss@text: Macro added. 261

1990-01-27 ltfssbas.dtx v1.2d
\DeclarePreloadSizes: Font identifier set to \relax. 160

1990-01-28 ltfssbas.dtx v1.2e
\mathgroup: \newfam let to \new@mathgroup. 154

1990-01-28 ltfssbas.dtx v1.2f
\define@newfont: Added call to \curr@fontshape macro to allow substitution. 166
\wrong@fontshape: Warning message slightly changed. . . 170

1990-01-28 ltfssini.dtx v1.2b
\em: Call to \@nomath added. . . . 259

1990-02-08 ltfssini.dtx v1.1g
General: Protected the commands \family, \series, \shape, \size, \selectfont, and \mathversion. 247

1990-02-16 ltfssbas.dtx v1.2g
General: Support for changes of \baselineskip without changing the size. 154
\math@version: \@nomath added. 164

1990-02-16 ltfsstrc.dtx v1.0i
\selectfont: Changed \f@size to \lcl@currsize (see fam file). 199

1990-02-18 ltfsstrc.dtx v1.0j
General: Redefine unprotected version \p@selectfont instead of \selectfont. 199

1990-03-14 ltfsstrc.dtx v1.0k
General: Added code for TeX3. . 195
\extract@font: Added code for TeX3. 198
\selectfont: Added code for TeX3. 199

1990-03-30 ltfssbas.dtx v1.2h
\math@egroup: Changed to have one arg. 174

\define@newfont: 165
\math@fonts: 172
\select@group: 172, 173
\split@name: Added splitting into \f@encoding. 165
\wrong@fontshape: 170

1992-07-26 ltfsstrc.dtx v2.0b
\s@fct@: 214
\selectfont: 199
\try@simple@size: 208, 209
\try@size@range: 212
\use@mathgroup: 205

1992-08-14 ltbibl.dtx LaTeX2.09
\@citex: added missing argument braces around \hbox, found by Ed Sznyter 472

1992-08-14 ltboxes.dtx LaTeX209
\endminipage: (RmS) replaced \vskip-\lastskip by \unskip (proposed by FMi) 385

1992-08-17 ltbibl.dtx LaTeX2.09
\@citex: simplified code for removing leading blanks in citation key (proposed by Frank Jensen and Kresten Krab Thorup) 472

1992-08-19 ltsect.dtx 0.0
\@xsect: (RmS) corrected bug: stretch and shrink in argument to \hskip previously not negated 443

1992-08-19 ltthm.dtx LaTeX2.09
\@othm: (RmS) Changed error message to complain about undefined counter 436

1992-08-20 ltfssini.dtx v1.4b
\@setsize: Added \@currsize. 260

1992-08-24 ltdefns.dtx LaTeX209
\@ifnextchar: (Rms) \@ifnextchar didn't work if its first argument was an equal sign. 50

1992-08-24 ltmiscen.dtx LaTeX2.09
\begin: Added code to \begin to remember line number. Used by \@badend to display position of non-matching \begin. 338
\verb: Changed \verb and \@sverb to work correctly in math mode 346

1992-08-25 ltsect.dtx LaTeX2.09
\@sect: (FMi) replaced explicit setting of \@svsec by call to \@seccntformat 442

1992-09-18 ltlists.dtx LaTeX2.09
\item: (RmS) Added warning if \item is used in math mode 371

1992-09-18 lttab.dtx LaTeX2.09
\@array: Changed \par to \@empty to avoid starting new row e.g. after \hline 401

1992-09-19 ltfsstrc.dtx v2.0c
\try@simple@size: 208

1992-09-21 ltfssini.dtx v1.4d
\not@math@alphabet: Macro defined. 259

1992-09-22 ltfssbas.dtx v1.91a
General: Introduced \tf@size for math size. 154

1992-09-22 ltfsstrc.dtx v2.1a
\getanddefine@fonts: Introduced \tf@size for math size. 206

1992-11-13 ltfssini.dtx v?
\hexnumber@: Made expandable. 260

1992-11-23 ltcounts.dtx LaTeX209
\stepcounter: Replaced {} in \stepcounter by \begingroup \endgroup to avoid adding an empty ord in math mode 147

1992-11-26 ltboxes.dtx LaTeX2.09
\@mpfootnotetext: (RmS) added protection for \edef 386

1992-11-26 ltfloat.dtx LaTeX2.09
\@footnotetext: (RmS) added protection for \edef 467
\footnote: (RmS) Changed all to ‘def‘protect‘noexpand‘protect‘noexpand 467

1992-12-03 ltfssini.dtx v?
\hexnumber@: Make it accept counters. 260

1993-03-08 preload.dtx v2.0b
General: Added 12pt preloads 284

1993-03-18 ltfssbas.dtx v2.0c
General: Changed all \@tempdima in \@tempdimb to avoid killing \numberline 154

1993-03-18 ltfsstrc.dtx v2.1b
General: Changed all \@tempdima in \@tempdimb to avoid killing \numberline 195
Changed all \@tempdimb in \@tempdimx to avoid killing \numberline 195

1993-03-18 ltfsstrc.dtx v2.1c
\DeclareSizeFunction: Added all args to avoid blanks problems 211

1993-04-09 lterror.dtx v1.0e
`\@latexerr`: Mention The Companion 65

1993-04-11 lterror.dtx v1.0f
`\@latexerr`: Remove setting of errorcontextlines 65

1993-05-05 ltfntcmd.dtx v2.0b
General: Removed all LaTeX related cmds 288

1993-05-16 ltfssbas.dtx v2.0e
`\showhyphens`: Use `\reset@font` 175

1993-07-16 ltfsstrc.dtx v2.1h
General: Changed layout of info messages 195

1993-07-17 ltoutenc.dtx 1.0d
General: changed `\catcoding` @ . 105

1993-08-03 ltmiscen.dtx LaTeX2.09
`\enddocument`: Changed redefinition of `\global` to redefinition of `\@setckpt`. . . 334

1993-08-05 ltpictur.dtx LaTeX2.09
`\circle`: (RMS) Added error message if `\circle` is used in math mode. 429

1993-08-05 ltsect.dtx LaTeX2.09
`\@sect`: (RmS) Made sure that `\protect` works correctly in expansion of `\the` counter . . 442

1993-08-05 ltspace.dtx LaTeX2e
`\@hspace`: (RmS) Removed superfluous `\leavevmode` in `\@hspace` and `\@hspacer`, as suggested by CAR. 85

1993-08-05 lttab.dtx latex2e
`\tabular*`: Replaced `\expandafter\def` by `\@namedef`. 400

1993-08-06 ltbibl.dtx LaTeX2.09
`\@citex`: Moved writing to .aux file in loop over citation keys so that leading blanks are removed there as well. 472

1993-08-13 ltoutenc.dtx 1.0f
General: Protected against active @ sign. 105

1993-08-13 preload.dtx v2.0c
General: Added `\relax` at end of font names. 285

1993-08-16 ltoutenc.dtx 1.0g
General: Needs space after `\string` 105

1993-08-18 ltfssdcl.dtx v2.0e
`\new@mathversion`: Exchanged names of encodings in warning message of `\SetSymbolFont`. 230

1993-09-02 ltfsstrc.dtx v2.1i
General: Corrected name of sgen size function. 195

1993-09-03 ltmiscen.dtx LaTeX2.09
`\verbatim@nolig@list`: Replaced `\@noligs` by extensible list . 347

1993-09-07 ltmiscen.dtx LaTeX2.09
`\verb@balance@group`: (RmS) Changed definition of `\verb` so that it detects a missing second delimiter. 346

1993-09-08 ltmiscen.dtx LaTeX2.09
`\enddocument`: Added warning in case of undefined references. 334

1993-09-15 ltfssbas.dtx v2.0g
`\DeclareFontEncoding`: Corrected: `\default@T` to `\default@M`. . 157

1993-09-15 ltfsstrc.dtx v2.1j
General: Corrected spelling of `\noxpand`. 195

1993-09-19 lterror.dtx LaTeX2.09
`\@invalidchar`: (RmS) Error message for invalid input characters. 67

1993-11-02 ltmath.dtx LaTeX2.09
General: RmS: Corrected description of `\@eqnsel`, moved `\@eqnsel` accordingly and removed extra `\tabskip` assignment. 356

1993-11-03 ltmath.dtx LaTeX2e
General: RmS: Initialized `\everycr` to empty 356

1993-11-03 ltpictur.dtx LaTeX2.09
General: (RmS) changed `\halign` to `\ialign` to initialize `\tabskip` and `\everycr` 414

1993-11-11 ltfssini.dtx v2.1a
`\normalfont`: Macro added 261

1993-11-11 ltfsstrc.dtx v2.2a
General: Option concept added for LaTeX2e 195

1993-11-14 ltclass.dtx v0.2a
`\@currext`: Name changed from `\@currextension` 555

`\@fileswithoptions`: Moved resetting of `\default@ds`, `\ds@` and `\@declaredoptions` here, from the end of `\ProcessOptions`. 565

`\@reset@ptions`: macro added . . 567

`\AtEndDocument`: Included extension in the generated macro name for package and class hooks. 567

\documentstyle: Added \RequirePackage \@unusedoptionlist stuff. . 562

\g@addto@macro: Made global . . 567

\NeedsTeXFormat: made more robust for alternative syntax for other formats. 563

\ProcessOptions*: Optimise ‘empty option’ code. 560

Stop adding the global option list inside class files. 560

1993-11-15 ltclass.dtx v0.2b

\documentstyle: Modified to match \ProcessOption* . . . 562

\ProcessOptions*: Star form added. 560

1993-11-17 ltclass.dtx v0.2c

\@@fileswith@pti@ns: Macro added 567

\@badrequireerror: Macro added 568

\@fileswithoptions: Added trap for two \LoadClass commands. 566

\@twoloadclasserror: Macro added 568

\CurrentOption: Name changed from \@curroption 555

\DeclareOption*: Error checking added 559

\NeedsTeXFormat: Name changed from \NeedsFormat 563

\ProcessOptions*: restoring \@fileswith@pti@ns added. 560

1993-11-18 ltclass.dtx v0.2d

\documentstyle: Modified \RequirePackage stuff. 562

\ExecuteOptions: Use \CurrentOption not \reserved@a 561

\NeedsTeXFormat: \fmtname \fmtversion not \@. 563

1993-11-21 ltfiles.dtx LaTeX2e

\@missingfileerror: Stop infinite looping on \@er@ext 99

1993-11-21 ltmiscen.dtx v0.9a

\@verbatim: use \verbatim@font instead of \tt 344

\verb: Use \verbatim@font instead of \tt. 346

\verbatim@font: Macro added . . 344

1993-11-22 ltclass.dtx v0.2f

\@fileswithoptions: Made the default [] not [\@unknownversion] 564

Made the initial version [] not [\@unknownversion] 565

\@ifclasslater: Added //00 so parsing never produces a runaway argument. 556

General: \@unknownversion removed 550

1993-11-22 ltdefns.dtx LaTeX2e

\@minus: Macro added 36

\@plus: Macro added 36

\CheckCommand: Macro added . . . 43

\providecommand: Macro added . 43

1993-11-22 lterror.dtx LaTeX2e

\c@errorcontextlines: Macro added 64

1993-11-22 ltfiles.dtx LaTeX2e

\listfiles: Removed checking for \@unknownversion 100

1993-11-22 ltlength.dtx LaTeX2e

\@settodim: Macro added 153

\@settopoint: Macro added 153

\settodepth: Macro added 153

\settoheight: Macro added 153

1993-11-22 ltlogos.dtx LaTeX2e

\LaTeXe: Macro added 87

1993-11-23 ltclass.dtx v0.2g

\@use@ption: Name changed from \@executeoption 561

General: Various macros now moved to latex.tex. 554

Warnings and errors now directly coded. 554

1993-11-23 ltdefns.dtx LaTeX2e

\@argdef: Macro added 39

\@ifundefined: Redefined to remove a trailing \fi 49

\@newcommand: Macro added 39

\@newenv: Macro interface changed 42

\@xargdef: Macro interface changed 39

\@yargd@f: Avoid \@?@? token . . 40

Macro interface changed 40

\newcommand: Macro reimplemented and extended . 39

\renewcommand: Macro reimplemented and extended . 41

\renewenvironment: Macro reimplemented and extended . 42

\two@digits: Macro added 36

1993-11-23 ltoutput.dtx v0.1a
- \paperheight: Register added . . 490
- \paperwidth: Register added . . . 490

1993-11-23 ltoutput.dtx v0.1c
- \@enlargepage: Command added 533
- \@kludgeins: Insert added 533
- \@makecol: Command changed . . 501
- \@specialoutput: Command changed 495
- \enlargethispage*: Commands added 533

1993-11-24 ltfntcmd.dtx v2.1a
- \maybe@ic@: Use \t@st@ic 293
- \t@st@ic: Macro added 293

1993-11-24 ltfssini.dtx v2.1a
- General: Removed \xpt stuff . . . 261

1993-11-24 ltlogos.dtx LaTeX2e
- \LaTeX: Macro changed 87

1993-11-28 ltclass.dtx v0.2h
- \@twoclasseserror: Macro added 568
- General: Assorted commands now in the kernel removed. 554
  - Directory syntax checing moved to dircheck.dtx 554
  - Primitive filenames now terminated by space not \relax. 554
- \endfilecontents: Don’t globally allocate a write stream (always use 15) 568

1993-11-28 ltfiles.dtx LaTeX2e
- \@missingfileerror: Use filename parser from dircheck 99

1993-11-29 ltoutput.dtx v1.0b
- \@makecol: \@makespecialcolbox added 501
- \@makespecialcolbox: Command added 503

1993-11-29 ltplain.dtx LaTeX2e
- General: All accents in decimals; suggested by Paul Taylor 29

1993-11-30 ltoutput.dtx v1.0c
- \fl@tracemessage: Commands added 535

1993-12-01 fontdef.dtx v2.1a
- General: Update for LaTeX2e . . . 264

1993-12-01 ltoutput.dtx v1.0e
- \@reinserts: Command added . 502

1993-12-03 ltboxes.dtx v0.1a
- \@argrsbox: macro removed 387
- \@begin@tempboxa: macro added 378
- \@end@tempboxa: macro added . . 378
- \@iirsbox: redefined to support \height 387
- \@imakebox: macro modified . . . 378
- \@irsbox: redefined to support \height 387
- \@isavebox: color support 380
  - extra group 380
- \@isavepicbox: extra group 380
- \@makebox: default changed from x to c 378
- \@makepicbox: macro modified . . 379
- \@savebox: default c not x 380
- \bm@b: macros added 378
- \endlrbox: macro added 380
- \fbox: extra group 381
- \lrbox: color support 380
  - macro added 380
- \makebox: modified 377
- \mbox: extra group 378
- \minipage: Redefined to support extra optional arguments . . . 385
- \newsavebox: Pass the whole of arg 1 to \@ifdefinable 379
- \parbox: Redefined to support extra optional arguments . . . 382
- \raisebox: redefined to support \height 387
- \sbox: color support 380
  - extra group 380
- \set@color: color support 379
  - macro added 379

1993-12-03 ltclass.dtx v0.2i
- \@cls@pkg: Name changed to avoid clash with output routine. . . 567
- General: \@onlypreamble: Many commands declared. 554
  - Removed obsolete \@documentclass 554

1993-12-03 lterror.dtx v1.0b
- \@latexerr: Set \c@errorcontextlines to -1 . 65

1993-12-03 ltfssini.dtx v2.1a
- General: update for LaTeX2e . . . 247

1993-12-04 ltfiles.dtx v0.9b
- \@iinput: Macro reimplemented . 99
- \@input: Macro reimplemented . 99
- \IfFileExists@: Macro added . . 97
- \input: Macro reimplemented . . 98
- \InputIfFileExists: Macro added 98

1993-12-05 ltfloat.dtx LaTeX2e
- \@dblfloatplacement: Command changed 459
- \@xfloat: Command changed . . 454

1993-12-05 ltoutput.dtx v1.0f
- \@addtobot: Command changed . 515

\@addtocurcol: Command changed . . . 516
\@addtodblcol: Command changed . . . 527
\@addtonextcol: Command changed . . . 523
\@addtotoporbot: Command changed . . . 515
\@boxfpsbit: Command added . 538
\@flcheckspace: Command added . . . 540
\@flsetnum: Command added . . 539
\@flsettextmin: Command added . . . 540
\@flstop: Commands added . . . 536
\@flupdates: Command added . 541
\@fpsadddefault: Command added . . . 537
\@getfpsbit: Command added . 538
\@opcol: Command changed . . . 501
Hook added . . . 501
\@outputpage: Command changed . . . 505
\@resethfps: Command added . 539
\@setfloattypecounts: Command added . . . 537
\@setfpsbit: Command added . 538
\@shipoutsetup: Command added . . . 505
\@startcolumn: Command changed . . . 510
\@startdblcolumn: Command changed . . . 510
\@testfp: Command added 538
\@textfloatsheight: Commands added . . . 537
\@topnewpage: Commands changed . . . 493
\@tryfcolumn: Command changed . . . 512
\@writesetup: \@startpagehook added . . . 505
\output: Command changed . . . 495

1993-12-06 ltclass.dtx v0.2k
\ExecuteOptions: Preserve \CurrentOption. . . . 561

1993-12-06 ltoutput.dtx v1.0f
\@specialoutput: Unboxing of 255 added to rescue writes 495

1993-12-06 ltoutput.dtx v1.0g
\@topnewpage: \@floatplacement placement bug fixed 493

1993-12-07 ltclass.dtx v0.2l
\ProvidesFile: Macro added . . . 559

1993-12-07 ltclass.dtx v0.2m
\@fileswithoptions: Reset \CurrentOption . . . 565

1993-12-07 ltoutenc.dtx 1.1
General: Protected all special characters with \string. . . . 105

1993-12-07 ltoutenc.dtx v1.1
General: Made all character numbers decimal. . . . 102
Removed a lot of equal signs and the like. . . . 102

1993-12-08 ltboxes.dtx v0.1b
\@begin@tempboxa: Extra braces for color support (braces removed from other macros) 378
\@irsbox: fix typo . . . 387
\@parboxto: \endgraf added due to extra group in \@begin@tempboxa . . . 383
\lrbox: move \@endpefalse out of the inner group . . . 380

1993-12-08 ltfntcmd.dtx v2.1b
General: Macros \rm, \bf and \sf moved to classes.dtx . . . 295

1993-12-08 ltlists.dtx LaTeX2e
\@item: use \sbox to support colour . . . 373

1993-12-08 ltspace.dtx LaTeX2e
\@bsphack: Command reimplemented . . . 77
Command reimplemented; late birthday present for Chris . . . 77
\@vbsphack: Command added . . 80

1993-12-09 ltboxes.dtx v0.1c
\@irsbox: fix another typo 387

1993-12-09 ltclass.dtx v0.2n
\documentstyle: input 209 compatibility file. . . . 562

1993-12-09 ltfiles.dtx v0.9e
\document: Hook added 90

1993-12-09 ltmiscen.dtx v0.9e
\enddocument: Hook added 334

1993-12-10 ltoutenc.dtx v1.2
General: Added source code for t1enc.sty. . . . 102

1993-12-11 ltfntcmd.dtx v3.0a
General: Complete reworking of all text commands, using just one creator function . . . 288
italic correction now put in front of penalty before glue 288
newcommands replaced by defs 288
newfontswitch command corrected and changed 288

- `\DeclareTextFontCommand`: Macro changed 290
- `\emph`: Macro changed 291
- `\fix@penalty`: Macro added 294
- `\maybe@ic`: Macro name changed 293
- `\maybe@ic@`: Macro and name changed 293
- `\sw@slant`: Macro changed 294
- `\textup`: Macros changed 291

1993-12-11 ltmath.dtx v0.9g

- General: Added a group around the first argument of `\frac` to prevent changes (for example font changes) from modifying the contents of the second argument. 356

1993-12-11 ltoutenc.dtx v1.2a

- General: Corrected for t1enc, math. 102

1993-12-11 ltsect.dtx LaTeX2e

- `\author`: Added default 438
- `\title`: Added default 438

1993-12-11 ltxref.dtx LaTeX2e

- `\@setref`: Macro added 331
- `\pageref`: Macro reimplemented . 331
- `\ref`: Macro reimplemented 331

1993-12-12 ltoutput.dtx v1.0h

- `\@cflb`: boxmaxdepth setting moved 509
  - defs changed to lets 509
- `\@cflt`: name changed 509
- `\@doclearpage`: defs changed to lets 500
- `\@makecol`: defs changed to lets . 502
- `\@resethfps`: Warnings added: minimal 539
- `\@startdblcolumn`: defs changed to lets 511
- `\@topnewpage`: braces removed . . 493
- `\@tryfcolumn`: defs changed to lets 512
- `\fl@tracemessage`: Commands changed 535

1993-12-13 ltclass.dtx v0.2o

- General: Removed setting `\errorcontextlines` (now in latex.tex) 554
- `\documentstyle`: compatibility file now latex209.sty. 562
- `\usepackage`: Fixed error handling 563

1993-12-13 ltdirchk.dtx v0.2a

- General: on the 'docstrip' pass, do not check openin path 10
- `\IfFileExists`: Removed interactive prompting for current directory syntax 10
- `\strip@prefix`: modified, name changed from `\stripmeaning`. . 5

1993-12-13 ltlists.dtx latex2e

- `\trivlist`: Initialised `\@itemlabel` 370

1993-12-13 ltmiscen.dtx v0.9h

- `\@noligs`: Readded `\@noligs` . . . 347
- `\@verbatim`: Readded `\@noligs` . 344
  - Removed optional argument of `\item` 343
- `center`: Removed optional argument of `\item` 342
- `flushleft`: Removed optional argument of `\item` 342
- `flushright`: Removed optional argument of `\item` 342

1993-12-13 ltoutenc.dtx v1.2b

- General: Corrected file name in driver code. 102

1993-12-13 lttab.dtx latex2e

- `\tabbing`: Removed optional argument of `\item` 395

1993-12-14 ltoutput.dtx v1.0i

- General: Section added to declare all parameters 546

1993-12-15 ltboxes.dtx v0.1d

- `\@iminipage`: Changed default from 'c' to 's' 385
- `\@iparbox`: Changed default from 'c' to 's' 383
- `\minipage`: Changed default from 'c' to 's' 385
  - extra space removed. 385
- `\parbox`: Changed default from 'c' to 's' 382

1993-12-15 ltclass.dtx v0.2p

- General: Removed extra '.'s from `\@@warnings` 554

1993-12-16 ltlogos.dtx LaTeX2e

- `\LaTeXe`: Extended logo by DPC 87

1993-12-16 ltmath.dtx v0.9i

- `\@@eqncr`: use `\refstepcounter` instead of shortcut 357
- General: use `\refstepcounter` instead of shortcut 356

1993-12-16 ltmiscen.dtx v0.9i

- General: `\literal` added 347

1993-12-16 ltpage.dtx LaTeX2e

- `\mark`: Init `\mark` at begin document 477

1993-12-16 ltspace.dtx LaTeX2e

- `\@bsphack`: Corrected optimisation :-) 77

1993-12-16 lttab.dtx latex2e

- `\@xhline`: Measure from middle of vertical rules 409

1993-12-17 ltclass.dtx v0.2q

- `\@documentclasshook`: Macro added 554
- `\@fileswithoptions`: Add `\@compatibility` hook 564
- `\documentstyle`: Match Alan's new code. 562

1993-12-17 ltoutenc.dtx 1.3

- General: Added this section 106
  - Removed all the hackery for use in `\DeclareFontEncoding`, and redid everything using `\DeclareTextFoo`. 118, 120
  - 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 `\ProvidesPackage` commands. Added XXXenc. 105

1993-12-17 ltoutenc.dtx v1.3

- General: Added `\EncodingSpecificAccent`, `\EncodingSpecificAccentedLetter` and `\EncodingSpecificCommand`. 102
  - Made Rokicki's encoding a proper encoding scheme rather than a variant of OT1. 102

1993-12-17 ltoutput.dtx v1.0j

- `\@opcol`: Hook removed 501
- `\@specialoutput`: Page room test added 496
- `\@topnewpage`: check for vsize too small added 493
  - Page room test added 495
- `\@writesetup`: —and then removed 505
- `\fl@tracemessage`: tracefloatvals made a document command 535

1993-12-17 ltpage.dtx LaTeX2e

- `\mark`: Removed init `\mark` at begin document, since it doesn't work. 477
- `\rightmark`: Stopgap solution to mark `\leftmark` and `\rightmark` work without initializing mark until the problem is solved. 477

1993-12-18 ltoutenc.dtx 1.3b

- General: Fixed typos with `\ProvidesPackage` lines. Added the `\NeedsTeXFormat` line. Added the last argument to `\DeclareEncoding`. Moved the use of the encodings to after their declaration. 105
  - Replaced the missing last argument to `\DeclareFontEncoding`. 118, 120

1993-12-18 ltoutenc.dtx 1.3c

- General: Rewrote for the new syntax of `\EncodingSpecific`. 118, 120
  - Split `\EncodingSpecificAccent` up into `\EncodingSpecific` and `\DeclareAccent`. 106

1993-12-18 ltoutenc.dtx v1.3a

- General: Replaced OT3 by XXX 102

1993-12-18 ltoutenc.dtx v1.3b

- General: Corrected typos. 102
  - Replaced the missing last argument to `\DeclareFontEncoding`. 102

1993-12-18 ltoutenc.dtx v1.3c

- General: A new syntax, separating accent-definitions from encoding-specific definitions, and allowing encoding-specific `\chardef`, `\let`, etc. 102
  - Rewrote for the new syntax of `\EncodingSpecific`. 102

1993-12-18 ltoutenc.dtx v1.3d

- General: Some T1 stuff had drifted into the OT1 file. 102

1993-12-18 ltpage.dtx LaTeX2e

- `\sloppy`: Added `\emergencystretch` 477

1993-12-19 ltclass.dtx v0.2r

- `\endfilecontents`: Different message when ignoring a file 568

1993-12-19 ltfntcmd.dtx v3.0b

- General: `\@pdef` command added 288
  - Added by ASAJ. 296
  - Made `\@newfontswitch` produce an error if command already exists, and added `\@renewfontswitch`, ASAJ 288
  - Other tidying 288
  - Some more tidying done 288

    - Untidying added, so this is now a TEMPORARY version. 288
    - Wording changes by CAR. 296
- \DeclareOldFontCommand: Corrected and tidied 295
- \DeclareTextFontCommand: Corrected and tidied 290

1993-12-19 ltspace.dtx LaTeX2e
- \@bsphack: There seem to be problems with selfmade birthday presents 78

1993-12-20 ltdefns.dtx LaTeX2e
- \@reargdef: Kept old version of \@reargdef, for array.sty 41

1993-12-20 ltfiles.dtx v0.9m
- \@obsoletefile: Added this command, removed @oldfilewarning 100

1994-01-05 fontdef.dtx v2.1d
- General: Removed nf prefix from file names. 266

1994-01-13 ltmath.dtx v0.9o
- \@@eqncr: correcting 0.9i 357
- General: correcting 0.9i 356

1994-01-14 ltdirchk.dtx v0.2d
- \IfFileExists: Close the texsys.aux output stream 10

1994-01-15 ltfiles.dtx v0.9o
- \document: move \@preamblecmds after document hook 92

1994-01-17 ltclass.dtx v0.2s
- \@fileswithoptions: Modify to reduce parameter stack usage 564, 566
- General: Added many more \@onlypreamble commands 554
    - Wrapped long lines to column 72 554

1994-01-17 ltfiles.dtx LaTeX2e
- \listfiles: New Version, adds '.tex' if needed, and lines up columns 100

1994-01-17 ltfssbas.dtx v2.1a
- General: New math font setup 154
- \curr@math@size: New math font setup 165
- \everydisplay: New math font setup 164
- \everymath: New math font setup 164
- \frozen@everydisplay: New math font setup 164
- \frozen@everymath: New math font setup 164
- \math@version: New math font setup 164

1994-01-17 ltfssini.dtx v2.1e
- \not@math@alphabet: Message changed 259

1994-01-17 ltfsstrc.dtx v2.3a
- General: New math font setup 195
- \check@mathfonts: New math font setup 204
- \glb@currsize: New math font setup 201
- \restglb@settings: New math font setup 204

1994-01-18 ltbibl.dtx LaTeX2e
- \bibliography: Use \@input@ so include files are listed. 473

1994-01-18 ltclass.dtx v0.2t
- \@ifclassloaded: Fix typo \@pkgetension 555

1994-01-18 ltfiles.dtx v0.9p
- \@iffileonpath: Macro added 97
- \@input: do not use a different definition for \input@path 99
- \@input@: Macro added 99
- \IfFileExists@: New Definition 97
- \include: Use \@input@ so include files are listed. 94
- \InputIfFileExists: New Definition 98

1994-01-18 ltfssini.dtx v2.1f
- \not@math@alphabet: Message corrected 259

1994-01-18 ltmiscen.dtx v0.9p
- \@verbatim: Add \global\@inlabelfalse 343
    - Only add \penalty if in hmode 343

1994-01-19 fontdef.dtx v2.1e
- General: Added missing setting for symbols in bold version. 271

1994-01-19 ltdirchk.dtx v0.2e
- \IfFileExists: name changed from \test 9
- \input@path: No longer check that an empty group is in the path 11
- \strip@prefix: name changed from \strip@meaning, to match NFSS. 5

1994-01-19 ltmath.dtx v1.0n classes
- \mathindent: Deferred setting of \mathindent 358

1994-01-20 ltdirchk.dtx v0.2f
- General: \@copytexsys and the texsys.new file removed 9
    - Modify all of ltxcheck 14

- \IfFileExists: \@copytexsys removed 10

1994-01-21 ltclass.dtx v0.2u

- \documentstyle: compatibility file now latex209.def. 562

1994-01-21 ltdirchk.dtx v0.2g

- General: Improve documentation, reorganise docstrip module . . . 1
- \filename@parse: Minor changes, and add Mac version (:) 11
- \today: Name changed from \stamp, to save memory 9

1994-01-21 ltfloat.dtx LaTeX2e

- \@xfloat: Added missing percent characters. 454

1994-01-21 ltmiscen.dtx v0.9s

- \verbatim@font: Removed unnecessary category code hackery. 344

1994-01-24 ltdirchk.dtx v0.2h

- \IfFileExists: Stop testing once texsys.aux has been found . . . 10

1994-01-24 ltpage.dtx LaTeX2e

- \pagestyle: (DPC) Complain if pagestyle is undefined. 475

1994-01-25 ltdirchk.dtx v0.2i

- General: Protect against looping on \@@input and \@@end. 2

1994-01-25 ltfssbas.dtx v2.1b

- \math@version: Corrections for math setup 164

1994-01-25 ltmath.dtx LaTeX2e

- \bordermatrix: Removed \p@renwd. 352

1994-01-26 ltfsstrc.dtx v2.3c

- \check@mathfonts: Correct trace info placement 204
- \restglb@settings: Correct trace info placement 204

1994-01-27 ltfntcmd.dtx v3.1a

- \nocorrlist: Only ., used as default for cm fonts 294

1994-01-29 ltclass.dtx v0.2v

- \@@unprocessedoptions: Macro added. 568
- \@fileswithoptions: All options raise error if no \ProcessOptions appears . . 566

1994-01-31 ltclass.dtx v0.2w

- \g@addto@macro: Use toks register to avoid 'hash' problems . . . 567

1994-01-31 ltfiles.dtx v0.9t

- \document: set \@normalsize or \normalsize if necessary 91

1994-01-31 ltfntcmd.dtx v3.1b

- General: \@normalsize no longer defined 288

1994-02-01 ltpage.dtx LaTeX2e

- \pagestyle: (DPC) Modify to get nicer error message 475
- \thispagestyle: (DPC) Modify to get nicer error message 476

1994-02-02 ltclass.dtx v0.2x

- \@fileswithoptions: Only run the hook and options check if the file was loaded. 566

1994-02-03 ltoutput.dtx v1.0k

- \@makespecialcolbox: correct mistakes in the documentation 504

1994-02-07 ltclass.dtx v0.2y

- \@fileswithoptions: Run \@compatibility on the first class to start (not the first to finish) 564
- \@ifclasswith: Add extra ,s so 'two' is not matched with 'twocolumn' 557
- \ProcessOptions*: Add extra ,s so 'two' is not matched with 'twocolumn' 560

1994-02-07 ltfssbas.dtx v2.1c

- \DeclareFontEncoding: revert catcode settings earlier 157
- \DeclareFontShape@: revert catcode settings earlier 155

1994-02-08 ltoutput.dtx v1.0k

- \@makespecialcolbox:
  - boxmaxdepth setting added . 503
  - boxmaxdepth setting removed 503
- General: Documentation and tasks tidied. 478

1994-02-10 ltclass.dtx v0.2z

- \@documentclasshook: Changed the name from \@compatibility to \@documentclasshook, and added the check for whether \@normalsize has been defined. ASAJ. 554
- \@fileswithoptions: Renamed \@compatibility to \@documentclasshook. ASAJ. 564

1994-02-10 ltfssbas.dtx v2.1d

- \addto@hook: Made \addto@hook long. 177

1994-02-10 ltfsscmp.dtx v2.1d
- `\scan@@fontshape`: scan away stuff after pt 219

1994-02-22 ltfssini.dtx v2.1g
- General: Correct error message . . 262

1994-02-24 ltfssbas.dtx v2.1e
- `\DeclareFontShape`: Separate restoration of catcodes for fd cmds 155
- `\define@newfont`: Separate restoration of catcodes for fd cmds 166
- `\nfss@catcodes`: Separate restoration of catcodes for fd cmds 166

1994-02-25 ltdirchk.dtx v0.2j
- General: Remove need for drv file . 1

1994-03-01 ltdirchk.dtx v0.2k
- General: Add unstripped module, so that dircheck.dtx may be used with initex 1

1994-03-02 ltboxes.dtx v0.1e
- General: Add 2ekernel module . . 377
- Remove need for drv file 377

1994-03-02 ltclass.dtx v0.3a
- General: Remove need for driver file 554

1994-03-03 ltboxes.dtx v0.1f
- `\@irsbox`: Replaced a missing `\else` 387

1994-03-04 ltfloat.dtx v1.0a
- General: Initial version, split from latex.dtx 450

1994-03-04 ltsect.dtx v1.0a
- General: Initial version, split from latex.dtx 438

1994-03-04 lttab.dtx v1.0a
- General: Initial version, split from latex.dtx 389

1994-03-04 ltvers.dtx v1.0a
- General: Initial version, split from latex.dtx 33

1994-03-07 ltboxes.dtx v0.1a
- `\@mpfootnotetext`: Extra group for colour 386

1994-03-07 ltboxes.dtx v1.0a
- General: Unify format with other Kernel files 377

1994-03-07 ltdefns.dtx v1.0a
- `\@@italiccorr`: Macro added . . . 36

1994-03-07 ltfiles.dtx v1.0a
- General: Initial version, split from latex.dtx 88
- Long lines wrapped to 72 columns 88

1994-03-07 ltfinal.dtx v0.1a
- General: Add code from the old dump.dtx 624
- Initial version, split from latex.dtx 612
- move code here from lhyphen.dtx 617
- Remove oldcomments environment 612
- use `\InputIfFileExists` not `\IfFileExists` 617

1994-03-07 ltfloat.dtx v1.0a
- `\@endfloatbox`: (DPC) Extra group for colour 459
- `\@footnotetext`: (DPC) Extra group for colour 467
- `\@xfloat`: (DPC) Extra group for colour 455

1994-03-07 lthyphen.dtx v0.1c
- General: move the 2ekernel code to ltfinal.dtx 583

1994-03-07 ltlength.dtx v1.0a
- `\@settodim`: (DPC) Extra group for colour 153

1994-03-07 ltlists.dtx v1.0a
- General: Initial version, split from latex.dtx 362
- Long lines wrapped to 72 columns 362

1994-03-07 ltpage.dtx v1.0a
- General: Initial version, split from ltherest.dtx 475

1994-03-07 ltpictur.dtx v0.1a
- General: Initial version, split from latex.dtx 411
- Long lines wrapped to 72 columns 411

1994-03-07 ltsect.dtx v1.0a
- `\@hangfrom`: (DPC)Extra groups for colour 445

1994-03-07 lttab.dtx v1.0a
- General: Long lines wrapped to 72 columns 389

1994-03-08 ltclass.dtx v0.3b
- General: Modify driver code into ‘new style’ 554

1994-03-08 ltdirchk.dtx v1.0a
- General: Reorganise driver module into ‘new style’ 1

1994-03-08 ltplain.dtx v1.0a
- General: Remove need for a driver file. 15

\@topnewpage: Corrected and amended warning message 494
Warning added: it should be improved 495
General: Added some warnings when page gets full of top floats. 478
Driver added and further tidying. 478
Removed duplicated code and corrected docstrip options. 478
Some boxmaxdepth settings removed. 478

1994-03-16 ltclass.dtx v0.3f
General: Add pkgindoc package 582

1994-03-16 ltfiles.dtx LaTeX2e
\listfiles: Move this code directly into \document 101

1994-03-16 ltfiles.dtx v1.0c
\document: (DPC) directly add file list settings 91

1994-03-16 ltmiscen.dtx v1.0b
\@verbatim: Remove \global\@inlabelfalse again. 343

1994-03-28 ltalloc.dtx v1.0d
General: Redefinition of 'new' allocations removed. 54

1994-03-28 ltdirchk.dtx v1.0d
General: Improve documentation 1

1994-03-28 lterror.dtx v1.0d
\@invalidchar: (DPC) Comment out (use catcode15 instead) 67
General: Remove test for \inputlineno undefined. 64

1994-03-28 ltfiles.dtx v1.0d
\document: (DPC) Use \normalsize not \@normalsize 91
(DPC) remove \@normalsize check 91

1994-03-28 ltfloat.dtx v1.0b
\@caption: Use \normalsize not \@normalsize 453
General: Split further from ltherest.dtx 450

1994-03-28 ltlists.dtx v1.0b
General: Improve documentation 361

1994-03-28 ltmiscen.dtx v1.0c
General: Improve Documentation 333

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 438

1994-03-28 lttab.dtx v1.0b
General: Improve documentation 389

1994-03-28 ltthm.dtx v1.0a
General: Initial version, split from latex.dtx 434

1994-03-29 ltcounts.dtx v1.0c
General: Create file from parts of ltmiscen and ltherest. 146

1994-03-29 ltlength.dtx v1.0c
General: Create file ltcntlen from parts of ltmiscen and ltherest. 153

1994-03-29 ltmiscen.dtx v1.0d
General: Remove counter macros to ltcntlen 333

1994-03-29 ltpageno.dtx v1.0c
General: Create file ltcntlen from parts of ltmiscen and ltherest. 328

1994-03-29 ltxref.dtx v1.0c
General: Create file ltcntlen from parts of ltmiscen and ltherest. 329

1994-03-31 ltbibl.dtx v1.0a
General: Initial version of ltidxbib.dtx, split from ltherest.dtx 471

1994-03-31 ltidxglo.dtx v1.0a
General: Initial version of ltidxbib.dtx, split from ltherest.dtx 469

1994-04-09 ltcounts.dtx v1.0d
\@newctr: \@nocnterr now has counter name argument 147
\addtocounter: \@nocnterr now has counter name argument 147
\setcounter: \@nocnterr now has counter name argument 147
\stepcounter: Use \addtocounter to have name checked 147

1994-04-09 ltthm.dtx v1.0b
\@othm: Use standard counter error message (FMi) 436

1994-04-11 ltclass.dtx v0.3g
\endfilecontents: Add star form, dont write \endinput at the end of the file. 568
\ProvidesFile: Protect against weird catcodes. 559

1994-04-11 ltfssbas.dtx v2.1h
General: Added \defaultscriptratio and \defaultscriptscriptratio. ASAJ. 154

\defaultscriptratio: Macro added 175
\defaultscriptscriptratio: Macro added 175
1994-04-12 ltboxes.dtx v1.0c
General: Remove \@acci, now defined in ltplain.dtx 383
Remove \@dischyph, now defined in ltinit.dtx 383
1994-04-12 ltdefns.dtx v1.0g
\@dischyph: Define \@dischyph, was previously in ltboxes.dtx . 51
1994-04-12 ltplain.dtx v1.0d
General: Define \@acci 29
1994-04-12 ltvers.dtx v1.0b
General: Have version info generated automatically. 33
1994-04-14 ltfntcmd.dtx v3.2b
General: Macros renamed to non-private forms, JB 288
\DeclareOldFontCommand: Renamed from \@newfontswitch 295
1994-04-15 ltboxes.dtx v1.0d
\@isavebox: Added missing procent character. 380
1994-04-17 ltcounts.dtx v1.0e
\@newctr: Use \@nocounterr instead of \@nocnterr 147
\addtocounter: Use \@nocounterr instead of \@nocnterr 147
\setcounter: Use \@nocounterr instead of \@nocnterr 147
1994-04-17 lterror.dtx v1.0h
\@nocounterr: New name for error message, old error message (without arg) kept 65
1994-04-17 ltthm.dtx v1.0c
\@othm: Use new std counter error message (FMi) 436
1994-04-18 ltfinal.dtx v0.1b
General: Initialise \textheight, \textwidth and page style . 614
1994-04-18 ltfloat.dtx v1.0d
\@footnotetext: (DPC) Remove Colour support 467
\@savemarbox: (DPC) Remove Colour support 462
1994-04-18 ltfssbas.dtx v2.1i
General: Macro \no@alphabet@help removed again 154
\calculate@math@sizes: Changed message to log only 174

\no@alphabet@error: Use std LaTeX error macro 154
1994-04-18 ltfssdcl.dtx ???
\DeclareMathAlphabet: Pass correct arg (2 not 3) 233
1994-04-18 ltfssdcl.dtx v2.1d
General: Removed surplus \no@alphabet@error (see fam.dtx) 222
1994-04-18 ltfsstrc.dtx v2.3d
General: Changed to new error/warning scheme 195
\font@submax: Changed dimen to macro 212
\fontsubfuzz: Changed dimen to macro 212
\subst@size: \font@submax and \fontsubfuzz now macros . . 213
1994-04-19 ltpage.dtx v1.0b
General: Improve documentation 475
1994-04-20 ltfntcmd.dtx v3.3a
General: Documentation up-dated 288
New implementation of \nocorr 288
\check@nocorr@: Macros added . 292
\maybe@ic@: \nocorr etc removed from list of tokens to check, leaving only punctuation characters 293
1994-04-20 ltmiscen.dtx v1.0e
\enddocument: Changed logic for producing warning messages 335
1994-04-21 ltboxes.dtx v1.0e
\@iiiminipage: Extra \bgroup for colour 385
\@mpfootnotetext: Extra \endgraf for colour 386
\endminipage: Extra \egroup for colour 385
1994-04-21 ltfinal.dtx v0.1c
General: Added comments, set the catcodes of 128–255. 612
1994-04-22 ltfssini.dtx v2.1g
\not@math@alphabet: Message changed again 259
1994-04-23 ltfinal.dtx v0.1d
General: Check that \font@submax is still zero 612
1994-04-24 ltoutput.dtx v1.0m
\@resethfps: Number 2 changed to \tw@ 539
Warning changed 539

- `\@specialoutput`: Message changed to give more info and ‘top’ removed 496
- `\@topnewpage`: Message changed to give more info 495
  - Warning message removed as it will be generated later 494
- General: Changed `\@normalsize` to `\normalsize`. 478
  - Corrected unverbed commands in documentation. 478
  - Removed some long lines and other aesthetic changes. 478
  - Warning messages changed/corrected. 478

1994-04-24 ltpictur.dtx v0.1b

- General: Removed surplus spaces after `\hbox to` in several cases 411

1994-04-25 ltclass.dtx v0.3h

- General: Removed spurious extra ‘.’s at the end of error messages 554

1994-04-25 ltfloat.dtx v1.0e

- `\@largefloatcheck`: Changed warning message to give more info 459
  - Command added 459
- General: Changed warning messages 450
  - Removed obsolete tracing code 450

1994-04-27 ltfsstrc.dtx v2.3e

- General: Corrected item that was forgotten in last change. 195

1994-04-28 lterror.dtx v1.0j

- `\@inmatherr`: Macro added 67

1994-04-28 lterror.dtx v1.1c

- `\@inmatherr`: Replaced `\noexpand` with `\protect`. 67

1994-04-28 ltfssdcl.dtx v2.1e

- General: Removed all `\uppercase` in hex num parsing macros 222

1994-04-28 ltlists.dtx v1.0c

- `\item`: Replaced `\@ltxnomath` by `\@inmatherr` 371

1994-04-28 ltpictur.dtx v0.1c

- General: bezier curves added 431
- `\multiput`: (DPC) Ignore spaces between)(413
  - (DPC) Macro added 413
- `\picture`: (DPC) Ignore spaces before (412

1994-04-28 ltplain.dtx v1.0g

- General: Turn off overfull box tracing in log 25

1994-04-29 ltclass.dtx v1.0a

- General: Change version number to 1 (no other change) 554

1994-04-29 ltmiscen.dtx v1.0f

- `\@verbatim`: `\leavevmode` added 343
  - Change to `\everypar` added 344

1994-04-29 ltoutenc.dtx 1.4a

- General: Removed `\EncodingSpecific`. Renamed all the commands. Added `\DeclareTextGlyph` and `\UndeclareTextCommand`. 106
  - Removed Rokicki’s OT1 variant encoding. Moved the driver to the top. 105

1994-04-30 ltfntcmd.dtx v3.3b

- General: Documentation up-dated and tidied 288
  - Prefix frag@ changed to frag in `\@protecteddef` 288
  - Title changed 288
  - Warning changed to info message in `\@protecteddef` 288

1994-04-30 ltoutput.dtx v1.0n

- `\@activechar@info`: `\@activechar@warning` changed to `\@activechar@info` 504
- `\@combinedblfloats`: Removed rule in topnewpage case 510
- `\@emptycol`: Empty column action added: `\@emptycol` 493
- `\@flsetnum`: Rogue space removed 539
- `\@specialoutput`: Cut-off point changed to 2`\baselineskip` 496
  - Empty column action added: `\@emptycol` 496
  - Extra empty column added for twocolumn case 496
  - Extra empty column added for twocolumn case (wrong, see below) 496
- `\@topnewpage`: Added setting of `\col@number` 493
  - Cut-off point changed to 3`\baselineskip` 495
  - Empty column action added: `\@emptycol` 495
  - Message changed for Frank 495

General: \@activechar@warning changed to an info message. . 478
Added \col@number. 478
Documentation tidied. 478
Empty column action added. . 478
Fixed bug from \dblfigrule with \@topnewpage. 478
Full of floats action improved. . 478
\col@number: Added \col@number 490
\onecolumn: Added setting of \col@number 492
1994-05-01 lterror.dtx v1.0k
\@latexerr: (CAR) Added draft \@latexinfo. 65
1994-05-01 ltoutenc.dtx 1.4a
General: Added the \a command. 114
Added the \SaveAtCatcode and \RestoreAtCatcode commands. 118
Removed the uc/lc table settings, since the T1 uc/lc table is now the default. . . . 125
Rewrote for the new syntax. 118, 120
1994-05-01 ltoutenc.dtx v1.4a
General: Removed Rokicki's encoding. 102
Renamed the commands, removed the \EncodingSpecific command. Turned all slots into decimal. Added \a. 102
1994-05-02 ltcntrl.dtx v1.0l
\@break@tfor: Macro added (from ltfiles.dtx) 59
1994-05-02 ltdefns.dtx v1.1f
\renewcommand: Removed surplus \space in error 41
\renewenvironment: Removed surplus \space in error 42
1994-05-02 ltfiles.dtx v1.0f
\@iffileonpath: \@break@loop renamed to \@break@tfor . . . 97
\@obsoletefile: Make \@onlypreamble 100
1994-05-02 ltfinal.dtx v0.1e
General: Added setting the 'letter' catcodes. 622
Added setting the 'other' catcodes. 622
Added setting the special catcodes. 621
Made slot 127 illegal 622
Set all the catcodes 612
1994-05-02 ltfinal.dtx v0.1f
General: Set the catcode of control-J. 621
1994-05-02 ltmiscen.dtx v1.0g
General: Changed 91 to 1991 and moved some bits 333
1994-05-02 ltoutput.dtx v1.0o
\@resethfps: Code shortened . . . 539
General: Code of \@resethfps shortened. 478
1994-05-03 ltbibl.dtx v1.0b
\nocite: Make \nocite issue a warning for an undefined citation key. 473
1994-05-03 ltfinal.dtx v0.1f
General: Set the catcode of control-J to be 'other', for use in messages. 612
1994-05-03 ltfloat.dtx v1.0f
General: (CAR) Added \@largefloatcheck 450
Removed unnecessary braces from arguments of \@ifnextchar 450
\end@dblfloat: \@largefloatcheck added . . 458
\end@float: (CAR) Added \@largefloatcheck 457
1994-05-03 ltfssdcl.dtx v2.1f
General: Renamed \@@DeclareMathDelimiter to \@DeclareMathDelimiter . . 222
1994-05-03 ltlists.dtx v1.0d
\@item: \hskip changed to \kern 372
\item: Removed superfluous braces 371
1994-05-03 ltmiscen.dtx v1.0h
\@centercr: \@badcrerr replaced by \@nolnerr 341
1994-05-03 lttab.dtx v1.0d
\@endpbox: Use \@finalstrut based on depth of \@arstrutbox 410
1994-05-04 ltclass.dtx v1.0b
\NeedsTeXFormat: Changed wording of the warning 563
1994-05-04 lterror.dtx v1.0m
\@badcrerr: Error message removed 67
1994-05-05 ltbibl.dtx v1.0c
\@citex: Set switch for warning and end of run. 472

1994-05-13 ltfiles.dtx v1.0g
- \document: Added execution of \every@size 91

1994-05-13 ltfinal.dtx v0.1h
- General: Added package ot1enc, and defined \@acci, \@accii and \@acciii. 612

1994-05-13 ltfinal.dtx v1.0h
- General: Added output enc stuff 624

1994-05-13 ltfloat.dtx v1.0g
- \@footnotetext: (DPC) Add new style colour support: \normalcolor 467
  - (DPC) Use \@finalstrut 467
- \@xfloat: (DPC) Use \normalcolor 455

1994-05-13 ltfntcmd.dtx v3.3g
- General: Replaced \@protecteddef by \DeclareRobustCommand 288

1994-05-13 ltfssbas.dtx v2.1k
- General: Remove File identification ‘typeout’ 154

1994-05-13 ltfssbas.dtx v2.1l
- \DeclareFontEncoding: Init encoding change command 158
- \define@newfont: Use \@input@ for fd files 166

1994-05-13 ltfssdcl.dtx v2.1h
- General: Removed file identification typeout 222

1994-05-13 ltfssini.dtx v2.1j
- General: Removed file identification typeout 247

1994-05-13 ltfsstrc.dtx v2.3g
- General: Removed typeouts as \ProvidesPackage writes to log. 195

1994-05-13 ltoutenc.dtx v1.5b
- General: Added \{, \} and \$. 102
  - Renamed \DeclareProtectedCommand to \DeclareRobustCommand. 102
  - Replaces \space by ‘ ’ in \csname. 102

1994-05-13 ltpictur.dtx v0.1d
- General: Removed surplus braces from \@if.. constructions 411

1994-05-13 lttab.dtx v1.0d
- \@contfield: Colour support 395
- \@startfield: Colour support 394
- \@stopfield: Colour support 395
- \a: moved to ltoutenc 393

1994-05-14 fontdef.dtx v2.1f
- General: Removed .def files. 266

1994-05-14 ltfssbas.dtx v2.1m
- \enc@update: Macro added 162

1994-05-14 ltfssbas.dtx v2.1n
- General: Set defaults for all \f@... 163
- \DeclareErrorFont: Don’t set \f@encoding 169
- \DeclareFontEncoding: Log if encoding is redeclared 158
  - Only init enc change cmd when new encoding 158

1994-05-14 ltfssini.dtx v2.1k
- General: Init error font just before checking for fontdef.cfg 262
- \reset@font: Remove surplus braces 261

1994-05-14 ltfsstrc.dtx v2.3h
- \selectfont: Added \enc@update 200

1994-05-14 ltoutenc.dtx 1.5d
- General: Moved the driver to the top. 105

1994-05-14 ltoutenc.dtx v1.5c
- General: Added the fontenc package 143
  - Added the fontenc package. 102
  - Fixed a bug which caused an infinite loop if \f@encoding was incorrectly set. 102, 106
  - Moved fontsmpl to its own dtx file. 102

1994-05-14 ltoutenc.dtx v1.5d
- General: Rewrote \DeclareTextCommand to define its argument to use the current encoding by default, rather than the encoding provided to \DeclareTextCommand. 102, 106
  - Tidied up the documentation. 102

1994-05-14 ltoutenc.dtx v1.5e
- General: Replaced \ENC@cmd by \ENC-cmd. 102

1994-05-15 ltfssbas.dtx v2.1o
- General: encoding cmds changed to enc-cmd 154

1994-05-16 fontdef.dtx v2.1g
- General: Removed \DeclareFontEncoding for ot1 and t1 and input .def files instead 266

1994-05-16 ltalloc.dtx v1.1a
- General: (ASAJ) Split from ltinit.dtx. 54

1994-05-16 ltcntrl.dtx v1.0a
General: (ASAJ) Split from ltinit.dtx. 56

1994-05-16 ltdefns.dtx v1.1a
General: (ASAJ) Split from ltinit.dtx. 36

1994-05-16 lterror.dtx v1.1a
General: (ASAJ) Completely new error interface. 60
(ASAJ) Split from ltinit.dtx. . 60

1994-05-16 ltfinal.dtx v1.0i
General: moved output enc stuff to lfonts 624

1994-05-16 ltfssbas.dtx v2.1p
`\fontsize`: Pass `\baselinstretch` not `\f@linespread` 163
`\linespread`: Remove surplus braces 163

1994-05-16 ltfssini.dtx v2.1m
`\@acciii`: Define saved versions of accents 263

1994-05-16 ltlogos.dtx v1.1a
General: (ASAJ) Split from ltinit.dtx. 87

1994-05-16 ltmath.dtx v1.0k
`\ensuremath`: Use `\DeclareRobustCommand` and add extra braces in math mode 357

1994-05-16 ltoutenc.dtx 1.5h
General: `\pounds` was still using u rather than ui shape. 118

1994-05-16 ltoutenc.dtx v1.5f
General: enc files now have uc encoding name parts (FMi) . 102
Revert code so that the encoding given is used in `\DeclareTextCommand` (FMi) 102

1994-05-16 ltoutenc.dtx v1.5g
General: Made fontenc.sty use the new mixed-case encoding files. 102
Removed the lowercasing of the filename. 143

1994-05-16 ltoutenc.dtx v1.5h
General: Added `\NG`, `\ng`, `\TH`, `\th`, `\DH`, `\dh`, `\DJ` and `\dj`. 102
Added `\r` (ring accent) and `\k` (ogonek) accents. 102
Fixed a bug with `\pounds`. . . . 102
Removed `\P` from the OT1 definitions file. 102

1994-05-16 ltoutenc.dtx v1.5i
General: Fixed a bug with `\d`. . . 102

1994-05-16 ltoutput.dtx v1.0q
`\@writesetup`: Changed setting of accents (FMi): with the new encoding setup they can use `\let`. It could also use the new internal commands? 506
General: Changed setting of accents (FMi). 478

1994-05-16 ltpar.dtx v1.1a
General: (ASAJ) Split from ltinit.dtx. 69

1994-05-16 ltplain.dtx v1.0h
General: Comment out encoding specific commands 29
Remove `\@acci` and friends again 29
Remove unnecessary def for `\item` 28
`\loop`: Use Kabelschacht method 27
`\m@th`: Remove unnecessary space 28

1994-05-16 ltspace.dtx v1.1a
General: (ASAJ) Split from ltinit.dtx. 71

1994-05-17 ltclass.dtx v1.0e
`\@use@ption`: Execute option after removing from list, not before 561

1994-05-17 ltdefns.dtx 1.1b
General: (ASAJ) Added the `\@protect@...` commands. . . 45

1994-05-17 ltdefns.dtx v1.1b
General: (ASAJ) Added definitions for protect. 36
(ASAJ) Removed warnings and logging to lterror.dtx. 36
Added the discussion of protected commands, defined the values that `\protect` should have. 44

1994-05-17 ltdefns.dtx v1.1c
General: (ASAJ) Redid definitions for protect. 36

1994-05-17 lterror.dtx v1.1b
General: (ASAJ) Moved error stuff from ltdefns.dtx. 60

1994-05-17 ltfssini.dtx v2.1n
`\copyright`: Really add extra braces 261
`\nfss@text`: Added braces to allow use in subscripts 261

1994-05-17 ltmath.dtx v1.0i
General: Replaced `\let` by `\gdef`, for indirect definition. 353

1994-05-17 ltoutenc.dtx v1.5j
General: Added braces to `\pounds` so it works as a subscript. 102

1994-05-18 ltdefns.dtx 1.1c
General: (ASAJ) Renamed the commands, and removed one which is no longer needed. 45

1994-05-18 ltdefns.dtx v1.1c
General: Redid the discussion and definitions, in line with the proposed new setting of `\protect` in the output routine. 44

1994-05-18 ltfinal.dtx v0.1j
General: Corrected the lccode for d-bar. 612

1994-05-18 ltlogos.dtx v1.1b
General: (ASAJ) Added the TeX logo. 87
(ASAJ) Made the LaTeX 2$_\varepsilon$ logo use the text font '2' rather than the math font '2'. 87

1994-05-18 ltoutenc.dtx v1.5k
General: Made dotted-i produce 'i'. 102
Removed braces from `\pounds` and `\dollar`. 102
Replaced `\defaultencoding` with `\encodingdefault`. 102

1994-05-19 ltbibl.dtx v1.1a
General: Initial version of ltbibl.dtx, split from ltidxbib.dtx 471

1994-05-19 ltcounts.dtx v1.1a
General: Extracted file from ltcntlen. 146

1994-05-19 ltdefns.dtx v1.1d
General: (RmS) Added definitions for `\@namedef` and `\@nameuse` again. 36

1994-05-19 ltfinal.dtx v0.1k
General: Removed `\makeat`... 612

1994-05-19 ltidxglo.dtx v1.1a
General: Initial version of ltidxglo.dtx, split from ltidxbib.dtx 469

1994-05-19 ltlength.dtx v1.1a
General: Extract file ltlength from ltcntlen. 153

1994-05-19 ltpageno.dtx v1.1a
General: Extract file ltpageno from ltcntlen. 328

1994-05-19 ltplain.dtx v0.1k ltfinal
`\showoutput`: used `\maxdimen` not 99999 30
`\showoverfull`: used `\@ne` not 1 . 30

1994-05-19 ltxref.dtx v1.1a
General: Extract file ltxref from ltcntlen. 329

1994-05-20 ltdefns.dtx v1.1e
General: Changed command name from `\@checkcommand` to `\CheckCommand`. 36
`\CheckCommand`: Changed name from `\@checkcommand` to `\CheckCommand`. 43

1994-05-20 lterror.dtx v1.1c
General: (ASAJ) Added `\@latex@info@no@line`. 60
(ASAJ) Added missing full stops. 60
(ASAJ) Fixed a bug with `\@inmatherr`. 60

1994-05-20 ltfinal.dtx v0.1l
General: Use new font warning commands 618

1994-05-20 ltfloat.dtx v1.0h
`\@endfloatbox`: Restore outer value of @nobreak switch. 459
`\outer@nobreak`: Macro added: default is to do nothing. 459

1994-05-20 ltfntcmd.dtx v3.3h
General: Use new error commands 288

1994-05-20 ltfssbas.dtx v2.1q
General: Use new error commands 154

1994-05-20 ltfsstrc.dtx v2.3i
General: Use new error command names 195

1994-05-20 ltmiscen.dtx v1.0l
`\@writefile`: Added correct setting of `\protect`. 337

1994-05-20 ltmiscen.dtx v1.0m
General: Use new warning commands 333

1994-05-20 ltoutput.dtx v1.0s
`\@writesetup`: Added setting of `\protect` during `\shipout`. 505
General: Added setting of `\protect` during `\shipout`. 478

1994-05-20 ltpage.dtx v1.0d
`\markright`: Changed setting for `\protect`. 476

1994-05-20 ltsect.dtx v1.0c
General: Correct setting of `\protect`. 447

\addcontentsline: Correct setting of \protect. 447

1994-05-21 ltbibl.dtx v1.1b

General: Use new warning commands 471

1994-05-21 lterror.dtx v1.1d

General: (ASAJ) Made the error commands robust. 60

1994-05-21 ltfiles.dtx v1.0h

General: Use new error commands 88

1994-05-21 ltlists.dtx v1.0f

General: Use new error commands 361

1994-05-21 ltmiscen.dtx v1.0n

General: Use new error commands 333

1994-05-21 ltsect.dtx v1.0d

General: Use new error commands 438

1994-05-21 lttab.dtx v1.0f

General: Use new error commands 389

1994-05-21 ltxref.dtx v1.1b

General: Use new warning commands 329

\newlabel: Use new warning commands 331

1994-05-22 ltclass.dtx v1.0f

General: Use new warning and error commands 550

1994-05-22 ltdefns.dtx v1.1f

General: Use new warning and error cmds 36

1994-05-22 lterror.dtx v1.1e

General: (ASAJ) Replaced bgroup by begingroup in error messages, to stop extra mathords creeping into math mode. 60

1994-05-22 lterror.dtx v1.2a

General: (ASAJ) Made \GenericError, \GenericWarning and \GenericInfo robust. 60

(ASAJ) Replaced \@generic@message and \@generic@error by \GenericError, \GenericWarning and \GenericInfo. 60

(ASAJ) Replaced \\ and tilde by \MessageBreak and \space. 60

(ASAJ) Replaces \string by \protect in some messages. . 60

1994-05-22 lterror.dtx v1.2d

\GenericError: (DPC) Alternative version added for old TeXs . . 60

(DPC) New version using long command name. 60

1994-05-22 ltfloat.dtx v1.0i

General: Use new warning commands 450

1994-05-22 ltoutput.dtx v1.0t

General: Changed warnings and infos to new commands. . . . 478

1994-05-22 ltpictur.dtx v0.1e

General: Use new warning cmds . 411

1994-05-23 ltclass.dtx v1.0h

\NeedsTeXFormat: Don't stop completely when format is wrong 563

\usepackage: Remove argument if possible 563

1994-05-23 ltdirchk.dtx v1.0f

General: Document \@TeXversion 1

1994-05-23 ltfsstrc.dtx v2.3j

General: Removed def of \f@warn@break 212

1994-05-23 ltoutput.dtx v1.0u

\@activechar@info: Added \MessageBreak 504

\@writesetup: Changed resetting of \protect after shipout to use \aftergroup 505

General: Added \MessageBreak. . 478

Changed resetting of \protect after shipout. 478

1994-05-24 lterror.dtx v1.2e

\@latex@info@no@line: Macro added 63

1994-05-24 lterror.dtx v1.2f

General: (DPC) wrap long lines . 60

1994-05-24 ltfntcmd.dtx v3.3i

General: Tidying and typos fixed 288

1994-05-24 ltmiscen.dtx v1.0q

\@currenvline: Use \@empty as outer default 340

1994-05-25 ltdirchk.dtx v1.0g

\filename@parse: Mac parser had " typo for : 12

1994-05-25 ltfntcmd.dtx v3.3j

General: Insertion of \aftergroups to implement \nocorr moved to the end of the group 288

\check@icr: Macros added 292

\check@nocorr@: Insertion of \aftergroups moved and defaults set up for efficiency . 292

\DeclareTextFontCommand: \expandafter inserted 290

Insertion of \aftergroups moved 290

1994-05-25 ltoutput.dtx v1.0v
General: Extra documentation. . . 478

1994-05-25 ltsect.dtx v1.0e
\@dottedtocline: Put braces around argument 4 (the actual toc entry) to avoid font (and possibly other) changes leaking out to the leaders. 448

1994-05-25 ltthm.dtx v1.0c
General: Modify documentation . 434

1994-05-25 ltvers.dtx v1.0d
General: Remove PRELIMINARY TEST RELEASE from startup banner (spring is here) 33

1994-05-25 ltxref.dtx v1.1c
General: Modify documentation . 329

1994-05-26 ltfiles.dtx LaTeX2e
\@missingfileerror: Modify message format 99

1994-05-26 ltlogos.dtx v1.1c
General: Remove \SLiTeX logo . . 87

1994-05-26 ltmiscen.dtx v1.0r
General: \literal removed 347

1994-05-26 ltplain.dtx v1.1m
\iterate: (CAR) added \long . . 27
\underbar: (CAR/FMi) changed to use box \tw@ 28

1994-05-26 ltplain.dtx v1.1p
\underbar: (DPC) changed to use \sbox 28

1994-05-29 ltfssdcl.dtx v2.1j
General: Use new error commands 222

1994-05-31 ltfinal.dtx v1.0n
General: Renamed lthyphen.* to lthyphen.*. 612

1994-06-01 ltboxes.dtx v1.0i
\@frameb@x: Macro added. 382
\@iframebox: New version, so \width is correct in \framebox 382
\fbox: New version, using \@frameb@x 381
\framebox: New version, so \width is correct in \framebox 381

1994-06-01 ltlogos.dtx v1.1d
\LaTeX: Add \m@th to force math size calculations 87

1994-06-01 ltoutput.dtx v1.0w
General: Tidied up typesetting. . 478

1994-06-08 ltfinal.dtx v1.0m
General: Add patch file system . . 624

1994-06-09 ltfinal.dtx v1.0n
General: For TEX2, do not set codes for higher half of character table. 616, 622

1994-06-09 ltfntcmd.dtx v3.3k
General: Tidying and typos fixed in documentation 288

1994-06-18 ltfntcmd.dtx v3.3l
General: Added check for empty text 288
\check@nocorr@: Added check for empty text 292

1994-06-22 ltfntcmd.dtx v3.3m
General: Removed space from \nfss@text 288
Renamed \check@nocorr 288
\check@nocorr@: Renamed \check@nocorr to \text@command to improve \long error message 292
\DeclareTextFontCommand: Removed space from \nfss@text 290

1994-06-22 ltmath.dtx v1.2t classes
\mathindent: Set \mathindent at the end of the class instead of at begin document 358

1994-07-20 ltlogos.dtx v1.1e
\LaTeX: Save a few tokens 87
\LaTeXe: Save a few tokens 87

1994-07-20 ltpage.dtx v1.0h
\sloppy: Save a few tokens 477

1994-09-16 ltfssbas.dtx v2.1s
\nfss@catcodes: Reset [and] as well, just in case 167

1994-10-07 ltoutenc.dtx v1.5l
General: Moved the ogonek accent. 102

1994-10-11 ltdirchk.dtx v1.0h
\@TeXversion: Check for TeX3.14 13
General: Modify all of ltxcheck again 14

1994-10-12 ltsect.dtx v1.0f
General: Doc. typos 438

1994-10-14 fontdef.dtx v2.2a
General: New coding 264

1994-10-14 ltfssini.dtx v2.2a
General: New coding for cfg files . 247

1994-10-14 ltmiscen.dtx v1.0s
General: Move math to other file 333

1994-10-14 ltplain.dtx v1.1a
General: Moved code to other files. 15

- 1994-10-15 ltfssbas.dtx v2.1t
  - \extract@alph@from@version: Warn if math alpha is used outside math 174
- 1994-10-18 ltboxes.dtx v1.0j
  - \@frameb@x: \leavevmode added 382
  - \@iframebox: \leavevmode moved to \@frameb@x 382
  - \@parboxto: Macro added to remove misuse of \@empty 383
  - General: stuff from ltpatch done 377
  - \fbox: \long added 381
  - \mbox: \long added 378
  - \sbox: \long added 380
- 1994-10-18 ltclass.dtx v1.0j
  - General: Move \listfiles to ltfiles.dtx 550
- 1994-10-18 ltdefns.dtx v1.2a
  - \@star@or@long: macro added 38
  - General: Add extra test for \endgraf 36
    - Add star-forms for all commands 36
  - \renew@environment: reset end command 42
- 1994-10-18 ltfiles.dtx v1.0i
  - \listfiles: code moved here from ltclass 100
- 1994-10-18 ltoutenc.dtx v1.5l
  - General: Added new definitions of \patterns and \hyphenation. 113
- 1994-10-18 ltoutenc.dtx v1.5m
  - General: Added new definitions of \patterns and \hyphenation. 102
- 1994-10-18 ltsect.dtx v1.0g
  - \@dottedtocline: Added \normalcolor for page number 448
  - General: Added \normalcolor 438
- 1994-10-19 ltfssbas.dtx v2.1t
  - \DeclareFontEncoding: Add missing \relax. 157
- 1994-10-23 ltfsstrc.dtx v23.k
  - \every@math@size: Renamed to \every@math@size 202
- 1994-10-23 ltmath.dtx v1.0l
  - \@eqnnum: Added \normalcolor since \eqno introduces a subgroup of the displayed math group 355
  - \ensuremath: Remove extra braces: but see p 168 of Leslie's book 357
- 1994-10-24 ltboxes.dtx v1.0k
  - \fbox: Inner braces added (to fix latex/1061) 381
- 1994-10-25 fontdef.dtx v2.2c
  - General: Added OMSenc.def 266
- 1994-10-25 ltboxes.dtx v1.0l
  - \@isavepicbox: missing percent (moved from ltpatch) 380
- 1994-10-25 ltdefns.dtx v1.2b
  - General: Documentation improvements 36
- 1994-10-25 ltoutenc.dtx 1.6a
  - General: Added \textdollar, \textlbrace, \textrbrace, \textsterling, \textunderline. 120
    - Removed \textlbrace, \textrbrace, \textunderline to give them their proper names. 120
- 1994-10-25 ltoutenc.dtx v1.6a
  - General: Added \ProvideTextCommand, \UseTextSymbol, \UseTextAccent, \DeclareTextSymbolDefault, \DeclareTextAccentDefault, \DeclareTextCommandDefault, and \ProvideTextCommandDefault. 102
    - Added the \Provide commands, and the default definitions. 106
    - Added the defaults. 114
    - Added the files OT1enc.def, T1enc.def and OMSenc.def. 114
    - Added the OMS encoding. 125
- 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

- to endminipage: as for main page. 386
- \color@endbox: macro added for colour support 379
- \color@hbox: macro added for colour support 379
- \endminipage: Colour resetting for footnotes moved to here: as for main page. 385

1994-11-05 ltboxes.dtx v1.0o

- \@mpfootnotetext: Colour groups restored here. 386

1994-11-05 ltfloat.dtx v1.1c

- \@dblflset: Add compatibility with old version of \@xfloat. 453
- \@endfloatbox: Use new \color@hbox concept. 459
- \@footnotetext: Removed \normalcolor (again) 467
- \@savemarbox: Use new \color@hbox concept. 462
- \@setfps: Add compatibility with old version of \@xfloat. 454
- \@xfloat: Add compatibility with old version of \@xfloat: but the arguments, provided at exorbitant cost, are now completely ignored 454
  - Use new \color@hbox concept. 455
- \@xympar: Use new \color@hbox concept. 463

1994-11-05 ltoutenc.dtx v1.6g

- General: Added setting of \@typeset@protect to \patterns and \hyphenation. 113

1994-11-05 ltoutput.dtx v1.1b

- \@topnewpage: Use new \color@hbox concept. 493
- \@writesetup: Change protect settings for new-style, protect-free aux-files. 505
  - Use new \color@hbox concept. 505

1994-11-05 ltoutput.dtx v1.1c

- \@begindvi: Added macro 509
- \@begindvibox: Added macro 490
- \@writesetup: Add new \AtBeginDvi concept 505
- \AtBeginDvi: Added macro 490

1994-11-06 ltfssbas.dtx v2.1u

- \cf@encoding: New macro 163
- \DeclareFixedFont: Renamed \every@size to \every@math@size. 156

1994-11-06 ltfssini.dtx v2.2b

- \@setsize: Use \@typeset@protect 260

1994-11-06 ltfsstrc.dtx v2.3k

- \glb@currsize: New implementation 201
- \try@simples: New implementation 212
- \try@size@substitution: New implementation 212
- \tryis@simple: New implementation 212

1994-11-07 fontdef.dtx v2.2f

- General: (DPC) Add \DeclareMathSizes declarations 271
  - (DPC) Updated to use \ProvidesFile 266

1994-11-07 ltfiles.dtx v1.0l

- \@unused: move here from ltdefns, remove duplicate \@mainaux 90

1994-11-07 ltfiles.dtx v1.0m

- \document: Renamed \every@size to \every@math@size. 91

1994-11-07 preload.dtx v2.1e

- General: (DPC) Updated to use \ProvidesFile 284

1994-11-09 ltboxes.dtx v1.0p

- \@finalstrut: Revert \finalstrut to 2.09 equivalent (from ltpatch) 388
- General: more colour changes... 377

1994-11-09 ltfssbas.dtx v2.1v

- \@vpt: (DPC) macros added, from setsize.dtx 177
  - (DPC) reduce save stack usage latex/1742 177

1994-11-10 ltbibl.dtx v1.1c

- General: Fix \nocite{*} 471
- \nocite: Fix \nocite{*} 473

1994-11-10 ltmath.dtx v1.2v classes

- eqnarray: Added value of \parskip to \abovedisplayskip to compensate for negative \topsep 359

1994-11-10 ltoutput.dtx v1.1e

- \@writesetup: Modify \protect setting 505

1994-11-10 ltplain.dtx v1.1b

- General: (CAR) added patch to \loop. 15
- \iterate: (CAR) added extra \relax 27

1994-11-11 ltspace.dtx v1.2a
- \\: (DPC) Make robust 75

1994-11-12 ltfntcmd.dtx v3.3o
- \normalsize: Added \MessageBreak 296

1994-11-12 ltlists.dtx v1.2b ltspace
- \endtrivlist: Changed order of tests to make \@noitemerror correct: end of an era. 370

1994-11-12 ltmiscen.dtx v1.0u
- center: Changed end macro to \def: safer and consistent 342
- flushleft: Changed end macro to \def: safer and consistent 342
- flushright: Changed end macro to \def: safer and consistent 342

1994-11-12 ltplain.dtx v1.1c
- General: Comment out more encoding specific commands 29

1994-11-12 ltspace.dtx v1.2b
- \addpenalty: Corrected error message 81
- \addvspace: Corrected error message 81

1994-11-13 ltspace.dtx v1.2c
- \addpenalty: Recorrected error message 81
- \addvspace: Recorrected error message 81

1994-11-14 ltoutput.dtx v1.1f
- \@begindvi: Use normal box register: why a box? 509
- \@begindvibox: Use normal box register: why a box? 490
- \@writesetup: Modify new \AtBeginDvi concept 505
- General: Removed old definition of \@testfp. 478

1994-11-14 ltspace.dtx v1.2d
- \\: (DPC) Macro modified 75

1994-11-14 lttab.dtx v1.0i
- \tabularnewline: (DPC) Macro added 402

1994-11-16 fontdef.dtx v2.2h
- General: (DPC) Removed \{ and \} 266

1994-11-17 ltboxes.dtx v1.0q
- General: \@tempa to \reserved@a 377

1994-11-17 ltclass.dtx v1.0l
- General: \@tempa to \reserved@a 550

1994-11-17 ltcntrl.dtx v1.0b
- General: \@tempa to \reserved@a 56

1994-11-17 ltdefns.dtx v1.0g
- General: \@tempa to \reserved@a 36

1994-11-17 ltdirchk.dtx v1.0j
- General: \@tempa to \reserved@a 1

1994-11-17 lterror.dtx v1.2h
- General: \@tempa to \reserved@a 60

1994-11-17 ltfiles.dtx v1.0n
- General: \@tempa to \reserved@a 88

1994-11-17 ltfinal.dtx v1.0o
- General: \@tempa to \reserved@a 612

1994-11-17 ltfloat.dtx v1.1e
- General: \@tempa to \reserved@a 450

1994-11-17 ltfntcmd.dtx v3.3p
- General: \@tempa to \reserved@a 288

1994-11-17 ltfssbas.dtx v2.1w
- General: \@tempa to \reserved@a 154

1994-11-17 ltfssdcl.dtx v2.1m
- General: \@tempa to \reserved@a 222

1994-11-17 ltfsstrc.dtx v2.3l
- General: \@tempa to \reserved@a 195

1994-11-17 ltmath.dtx v1.0o
- General: \@tempa to \reserved@a 348

1994-11-17 ltmiscen.dtx v1.0v
- General: \@tempa to \reserved@a 333

1994-11-17 ltoutenc.dtx v1.6h
- General: (DPC) \@tempa to \reserved@a 102

1994-11-17 ltoutput.dtx v1.1h
- General: \@tempa to \reserved@a. 478

1994-11-17 ltpictur.dtx v1.0f
- General: \@tempa to \reserved@a 411

1994-11-17 ltsect.dtx v1.0i
- General: \@tempa to \reserved@a 438

1994-11-17 lttab.dtx v1.0j
- General: \@tempa to \reserved@a 389

1994-11-18 ltboxes.dtx v1.0r
- \color@vbox: macro added for colour support 379

1994-11-18 ltfinal.dtx v1.0n
- General: re-allow slots 127–255 622

1994-11-18 ltfssbas.dtx v2.1x
- General: (DPC) use \reserved@f not \next 154

1994-11-18 ltfssdcl.dtx v2.1m
- \DeclareMathDelimiter: (DPC) \expandafter instead of \next 239

1994-11-18 ltfsstrc.dtx v2.3m
- General: \next to \reserved@f 195

1994-11-18 ltmath.dtx v1.0p
- \phantom: (DPC) colour support 350
  - (DPC) use \expandafter instead of \next 350
- \prime@s: (DPC) use \@let@token instead of \next and \expandafter instead of \nxt 353

1994-12-02 fontdef.dtx v2.2i
  General: Commented out \ldots. ASAJ. 264

1994-12-02 ltfssini.dtx v2.2c
  \copyright: \copyright is now in ltoutenc. ASAJ 261

1994-12-02 ltlists.dtx v1.0e
  \@trivlist: RmS: Added check for looping 369

1994-12-02 ltoutenc.dtx 1.7b
  General: Redefined \a properly. 114

1994-12-02 ltoutenc.dtx v1.7b
  General: Fixed a bug with \a. 102

1994-12-04 lthyphen.dtx v1.0h
  General: Documentation edits for /1989 583

1994-12-05 ltoutenc.dtx v1.7c
  General: Added braces to \textcircled. 102

1994-12-06 ltfssbas.dtx v2.1z
  \DeclareFontEncoding: use \nfss@catcodes 157
  \nfss@catcodes: Added tab char as well 167

1994-12-08 ltoutenc.dtx v1.7d
  General: Added \null and \sh@ft to \b and \d. 102

1994-12-08 lttab.dtx v1.0k
  \@array: Add \tabularnewline 401
  \tabularnewline: (DPC) Made it \relax 402

1994-12-09 ltbibl.dtx v1.1d
  \bibliographystyle: (DPC) Allow use in preamble. 473

1994-12-10 ltfloat.dtx v1.1g
  \@dblfloat: Old version reinstated temporarily 454
  \@dblflset: Macro removed temporarily 453
    Old version reinstated temporarily 453
  \@setfps: Macro removed temporarily 454
  \@xdblfloat: Macros reinserted temporarily 459
  \@xfloat: Old version reinstated temporarily 454
    Sanitisation added temporarily 454
  General: Some temps reinserted temporarily 450
  \fps@dbl: Macro removed temporarily 454

1994-12-10 ltfntcmd.dtx v3.3q
  \@@math@egroup: Don't read arguments 295
  \check@nocorr@: Use \space command for comparison 292

1994-12-10 ltfssdcl.dtx v2.1p
  \document@select@group: Surround with braces (add fourth arg) 226
  \select@group: Surround with braces (add fourth arg) 224

1994-12-10 ltoutenc.dtx v1.7e
  General: Added documentation for the OML encoding. 102
    Replaced width with \@width and ditto height in vrules. 102

1994-12-14 ltoutenc.dtx v1.7f
  General: Added braces to \copyright so it works unbraced in subscripts. 102
    Added check for math mode in \@changed@cmd. 102
    Commented out \textasciicircum, \textasciitilde, \textbackslash, \textbar, \textgreater, \texthyphenchar, \texthyphen and \textless to save memory. 102

1995-01-12 ltmath.dtx v1.2y classes
  \@eqnnum: Added \normalcolor 358

1995-03-03 ltoutenc.dtx 1.7g
  General: Corrected an error in documentation referring to the tabular rather than the tabbing environment. 114

1995-04-02 ltfntcmd.dtx v3.3r
  \@@math@egroup: Read them again to be able to add \relax. 295

1995-04-02 ltfssdcl.dtx v2.1q
  \document@select@group: fix problem for pr/1275 226
  \select@group: fix problem for pr/1275 224
  \set@mathdelimiter: fix pr/1329 242

1995-04-02 ltfssini.dtx v2.2d
  \not@math@alphabet: add \noexpand to second part of message 259

1995-04-21 ltclass.dtx v1.0m
  \DeclareOption*: Made long /1498 559

\endfilecontents: Close input check stream: latex/1487 . . . 568

1995-04-21 ltfinal.dtx v1.0q

General: Allow initial patch level 0 624

1995-04-21 ltoutenc.dtx v1.7h

General: Added \null \k latex/1274 102

1995-04-22 ltfiles.dtx v1.0p

\includeonly: Allow blanks in argument 94

1995-04-22 ltmiscen.dtx v1.0x

General: Removed extra def of \@gobble 333

1995-04-23 ltsect.dtx v1.0j

\addcontentsline: Use \contentsline internally. . . 447

1995-04-24 ltbibl.dtx v1.1e

\@citex: Add \mbox to undefined case: latex/1239. 472

1995-04-24 ltbibl.dtx v1.1f

\bibcite: Make \@onlypreamble /1388. 472

1995-04-24 ltcntrl.dtx v1.0d

\@for: Dont expand second argument with \edef: /1317 (DPC) 58

1995-04-24 ltoutput.dtx v1.1j

\fl@tracemessage: Do not add to kernel unless 'trace' specified 535

1995-04-24 ltoutput.dtx v1.1l

\@begindvibox: Add \vbox latex/1392 490

\@writesetup: Reset \\ latex/1451 (DPC) 506

1995-04-24 ltpage.dtx v1.0f

\fussy: reset \emergencystretch latex/1344 477

1995-04-24 ltplain.dtx v1.1h

\newlanguage: Remove remaining \outer declarations. 17

1995-04-24 ltxref.dtx v1.1e

\newlabel: Make \@onlypreamble for /1388. 331

1995-04-25 ltdefns.dtx v1.2i

\@check@c: Make \long for latex/1346 43

\new@environment: Parse arguments slowly but safely /1507 41

1995-04-25 ltfiles.dtx v1.0q

\document: Removed execution of \every@size latex/1407 91

1995-04-25 ltsect.dtx v1.0k

\@dottedtocline: Added \hbox around dots. 448

1995-04-27 ltboxes.dtx v1.0s

\@frameb@x: Move \leavevmode for graphics/1512 382

\@iframebox: Move \leavevmode for graphics/1512 382

\@iirsbox: Move \leavevmode for graphics/1512 387

\@irsbox: Move \leavevmode for graphics/1512 387

\fbox: Move \leavevmode for graphics/1512 381

\raisebox: Move \leavevmode for graphics/1512 387

1995-04-27 ltfiles.dtx v1.0r

\document: Added \global to support groups in hook 91

1995-04-27 ltmiscen.dtx v1.0y

\enddocument: \@checkend moved after hook 334

1995-04-27 ltplain.dtx v1.1i

General: Move \hang and \textindent to latex209.def . 28

1995-04-29 ltcntrl.dtx v1.0e

General: Moved init of \protect to ltdefns.dtx 59

Removed unused defs for \@setprotect and \@resetprotect 59

1995-04-29 ltdefns.dtx v1.2j

\protect: Init \protect here . . . 47

1995-04-29 ltpar.dtx v1.1b

General: (TO) Comments clean-up. 69

1995-05-02 ltsect.dtx v1.0l

\@dottedtocline: Don't reset to \rmfamily 448

1995-05-03 ltsect.dtx v1.0m

General: TO: Promoted documentation to doc.sty standard 438

1995-05-06 ltsect.dtx 1.0n

\@seccntformat: Use \quad instead of \hskip 444

\@sect: Added \relax after \@seccntformat just in case 442

1995-05-07 ltboxes.dtx v1.0t

General: Use \hb@xt@ 377

1995-05-07 ltdefns.dtx v1.2k

\hb@xt@: Macro added 36

1995-05-07 ltmath.dtx v1.0r

General: Use \hb@xt@ 348

1995-05-21 ltplain.dtx v1.1j
General: Moved some code to other files 15

1995-05-22 ltplain.dtx v1.1k
General: Definitions of `\footins` and `\footnoterule` moved to ltfloat. 30

1995-05-22 lttab.dtx v1.1a
General: Support autoloading feature 389

1995-05-23 ltfssini.dtx v2.2e
`\newfont`: Font assignment made local again. 260

1995-05-24 ltdefns.dtx v1.1l
`\newif`: (DPC) New implementation 42

1995-05-24 ltdefns.dtx v1.2m
`\typein`: (DPC) New implementation 37

1995-05-24 ltfloat.dtx v1.1l
`\@textsuperscript`: Command added. 467
General: Moved definition of `\footins` and `\footnoterule` from ltplain. 466
`\textsuperscript`: Use `\@textsuperscript` 467

1995-05-24 ltfssbas.dtx v3.0a
General: (DPC) Make file from previous file, fam.dtx 1995/05/20 v2.2d 154
`\mathgroup`: (DPC) No need to redefine `\newfam` as not outer 154

1995-05-24 ltfsscmp.dtx v3.0a
General: (DPC) Make file from previous file, fam.dtx 1995/05/20 v2.2d 218

1995-05-24 ltfssdcl.dtx v3.0a
General: (DPC) Make file from previous file, latint.dtx 1995/05/21 v2.1t 222

1995-05-24 ltfssini.dtx v3.0a
General: (DPC) Make file from previous file, lfonts.dtx 1995/05/23 v2.2e 247
`\cal`: (DPC) Remove definition . 263
`\mit`: (DPC) Remove definition . 263

1995-05-24 ltfsstrc.dtx v3.0a
General: (DPC) Make file from previous file, tracefnt 1995/05/16 v2.3o 195

1995-05-24 ltfsstrc.dtx v3.0b
General: (DPC) Fix `\ProvidesFile` usage 195

1995-05-25 ltclass.dtx v1.0p
`\endfilecontents`: Delete `\filec@ntents` after preamble 568

1995-05-25 ltfiles.dtx v1.0s
`\document`: Added check for `\topskip` zero 91

1995-05-25 ltfiles.dtx v1.0t
`\@iffileonpath`: (CAR) added `\long` 97
`\document`: Corrected typo 91
`\IfFileExists@`: (CAR) added `\long` 97
`\InputIfFileExists`: (CAR) added `\long` 98
`\nofiles`: (CAR) added `\long` . . 93
`\protected@write`: (CAR) added `\long` 94

1995-05-25 ltfloat.dtx v1.1m
`\@savemarbox`: (CAR) Resettings moved to hook 462
`\@xfloat`: (CAR) Resettings moved to hook 455

1995-05-25 ltlists.dtx v1.0i
`\endtrivlist`: Macros moved from ltspace.dtx 370

1995-05-25 ltmath.dtx v1.3c classes
`\@eqnnum`: replace `\reset@font\rmfamily` with `\normalfont` (PR 1578) . . . 358

1995-05-25 ltspace.dtx v1.2f
`\@vbsphack`: (CAR) not used so 'removed'. 80
`\@vspacer`: (CAR) `\@restorepar` added to avoid possible infinite tail recursion caused by a typo in the argument. 83
(CAR) macros modified to be more efficient 83
General: Macros moved to ltlists.dtx 71

1995-05-26 ltdefns.dtx v1.2n
`\@gobblefour`: (CAR) Added `\longs` 44

1995-05-26 ltmath.dtx v1.0s
`\@eqnnum`: Removed `\rmfamily` (PR 1578), replaced `\reset@font` with `\normalfont` 355

1995-05-26 ltpage.dtx v1.0g
`\ps@plain`: removed `\rmfamily` (PR 1578) 476

1995-05-27 ltfssbas.dtx v3.0b
`\mathgroup`: (FMi) But a need to define `\new@mathgroup` 154

1995-06-05 fontdef.dtx v2.2k
- General: Moved math commands from ltoutenc.dtx. 282

1995-06-05 ltfinal.dtx v1.0r
- General: Added `\MakeUppercase` and `\MakeLowercase`. 612

1995-06-05 ltoutenc.dtx v1.7k
- `\@inmathwarn`: Removed `\protected@cmd` and replaced with explicit `\noexpand`. . . . 106
- General: Allowed `\ProvideTextCommandDefault` after the preamble. 108
  - Commented out `\textless` and `\textgreater`. 115
  - Moved math commands to fontdef.dtx. 118
  - Save some tokens in `\textvisiblespace` and `\textunderscore`. 115

1995-06-06 ltfinal.dtx v1.0s
- General: Made `\MakeUppercase` and `\MakeLowercase` brace their argument. 612

1995-06-09 ltoutenc.dtx v1.7l
- `\DeclareTextComposite`: Rewrote `\DeclareTextComposite` to define the composite as a no-argument command rather than a two-argument command. 109

1995-06-11 ltspace.dtx v1.2g
- `\restorecr`: (CAR) `\relax` added to stop silent eating of *. 86

1995-06-13 ltfinal.dtx v1.0t
- General: Add patch level string more carefully 624
  - Call `\errorstopmode` 626

1995-06-13 ltpictur.dtx v1.1b
- General: Use `\ProvidesFile` in autoload 411

1995-06-14 lttab.dtx v1.1b
- General: Use `\ProvidesFile` in autoload 389

1995-06-15 ltfssbas.dtx v3.0c
- General: (DPC) minor documentation changes 154

1995-06-15 ltfsscmp.dtx v3.0b
- General: (DPC) minor documentation edits 218

1995-06-15 ltfssdcl.dtx v3.0b
- General: (DPC) minor documentation changes 222

1995-06-19 ltbibl.dtx v1.1h
- `\bibcite`: Call `\@newl@bel` so repeated keys produce better warning. 472

1995-06-19 ltclass.dtx v1.0q
- `\documentclass`: Dont redefine `\usepackage` in compat mode for /1634 562

1995-06-19 ltxref.dtx v1.1e
- `\newlabel`: Use `\@newl@bel` to share code with `\bibcite` . . 331

1995-06-28 ltfssini.dtx v3.0b
- General: (DPC) Fix documentation typos 247

1995-06-28 ltmath.dtx v1.0t
- General: minor doc edits 348

1995-07-02 ltplain.dtx v1.1n
- General: Removed surplus ‘by’ and ‘=’ in various places 15
- `\offinterlineskip`: Replaced 1000 by `\@m` 27
- `\showoutput`: Use `\showoverfull` to save space 30
- `\tracingall`: Use `\showoutput` to save space 30

1995-07-03 ltdefns.dtx v1.2o
- `\set@typeset@protect`: Use `\@typeset@protect` for init . . 46

1995-07-03 ltfntcmd.dtx v3.3s
- `\t@st@ic`: Use clean interface for jump 294

1995-07-05 ltfntcmd.dtx v3.3s
- `\t@st@ic`: Renamed from `\test@next` 293

1995-07-05 ltspace.dtx v1.2h
- `\@gnewline`: Use `\break` 76
- `\@no@pgbk`: Macro replaces `\@pgbk` and `\@nopgbk` 75
- `\nopagebreak`: Reimplemented both using `\@no@pgbk` 74

1995-07-09 ltcntrl.dtx v1.0f
- `\@iforloop`: Reimplemented using Kabelschacht method 59
- `\@iwhiledim`: Reimplemented using Kabelschacht method . . 57
- `\@iwhilenum`: Reimplemented using Kabelschacht method . . 57
- `\@iwhilesw`: Reimplemented using Kabelschacht method 57
- `\@tfor`: Reimplemented using Kabelschacht method 59

1995-07-09 ltlists.dtx v1.0j
- `enumerate`: Use `\expandafter` . . 375
- `itemize`: Use `\expandafter` 375

1995-10-03 fontdef.dtx v2.2l
General: `\@@sqrt` from patch file for /1701 264
1995-10-03 ltdefns.dtx v1.2r
`\typein`: Add missing `\@typein` for /1710 (from patch file) . . . 38
1995-10-03 ltpictur.dtx v1.1e
General: New autoload code 411
1995-10-04 ltfssbas.dtx v3.0g
General: Modify autoload code . . 154
1995-10-04 ltfsstrc.dtx v3.0d
General: (DPC) Modify autoload code 195
1995-10-04 lttab.dtx v1.1d
General: Modify autoload support 389
1995-10-06 ltfiles.dtx v1.0w
`\@missingfileerror`: Autoload error 99
1995-10-09 lterror.dtx v1.2j
General: Modify autoload support 60
1995-10-09 ltoutenc.dtx v1.7m
`\@inmathwarn`: Autoload error . . 107
1995-10-10 ltfssbas.dtx v3.0h
`\showhyphens`: Use `\normalfont` and make colour safe, and autoloadable 175
1995-10-10 ltfssdcl.dtx v3.0c
`\non@alpherr`: (DPC) autoload error message 226
1995-10-10 ltplain.dtx v1.1r
General: Autoload tracing code . 15
1995-10-10 ltthm.dtx v1.0f
General: Make `\newtheorem` 'only preamble' 434
1995-10-11 ltoutput.dtx v1.1r
`\clearpage`: Added a check so that it does not lose the argument of `\twocolumn[...]` 491
1995-10-16 ltbibl.dtx v1.1j
`\cite`: (DPC) Make robust 472
1995-10-16 ltboxes.dtx v1.0w
General: Clarify makebox description 377
1995-10-16 ltdefns.dtx v1.2u
`\@ifstar`: (DPC) New implementation, for /1910 . . . 50
`\new@command`: (DPC) Use `\@testopt` /1911 39
`\new@environment`: (DPC) Use `\@testopt` /1911 41
`\typein`: (DPC) Use `\@testopt` /1911 37
1995-10-16 ltfssini.dtx v3.0f
`\reset@font`: Added `\relax` after `\usefont`, as the latter eats up spaces. 261
1995-10-16 ltmath.dtx v1.0y
`\@yeqncr`: (DPC) Use `\@testopt` /1911 356
`\sqrt`: (DPC) Make robust /1808 356
1995-10-16 ltspace.dtx v1.2j
`\nolinebreak`: (DPC) Use `\@testopt` /1911 74
`\nopagebreak`: (DPC) Use `\@testopt` /1911 74
1995-10-16 ltthm.dtx v1.0g
General: Revert to previous `\newtheorem` behaviour 434
1995-10-17 ltclass.dtx v1.0r
`\@providesfile`: Delay definition of `\ProvidesFile` till ltfinal 559
`\ProcessOptions*`: Reset `\CurrentOption` for graphics/1873 561
1995-10-17 ltdirchk.dtx v1.0l
General: Modify initex version of `\ProvidesFile` 4
1995-10-17 ltfinal.dtx v1.0v
`\@providesfile`: reset macro . . . 626
`\reserved@b`: reset here after the `\input` above 625
1995-10-17 ltplain.dtx v1.1s
`\eject`: Move `\supereject` to compat file 28
1995-10-17 lttab.dtx v1.1e
`\@cline`: (DPC) Use `\@multicnt` 410
`\@multispan`: (DPC) Macro added. 410
1995-10-19 ltfinal.dtx v1.0w
`\@filelist`: Move after `\reserved@a` setting:-) 626
1995-10-20 ltbibl.dtx v1.1k
`\@citex`: Removed refundefined flag 472
`\nocite`: Removed refundefined flag 473
1995-10-20 ltclass.dtx v1.0s
`\@begindocumenthook`: Make setting conditional, for autoload version 567
1995-10-20 ltfssbas.dtx v3.0i
General: (DPC) Modify autoload code, change `\undefined` . . . 154
1995-10-20 ltfsstrc.dtx v3.0e
General: (DPC) Modify autoload code 195

Replaced octal number 27 by decimal number 23 to protect against the quote character being active. 119
Replaced some 0's by \z@ (faster). 119

1995-11-10 ltoutput.dtx v1.1s
\@shipoutsetup: Command removed 505
\@writesetup: Command removed 505
In-lined 505

1995-11-14 ltclass.dtx v1.0t
\@@unprocessedoptions: Allow empty option 568
\@loadwithoptions: macro added 562
\LoadClassWithOptions: macro added 563
\RequirePackageWithOptions: macro added 563

1995-11-17 ltfssbas.dtx v3.0m
\@wrong@font@char: (DPC) Macro added. latex/1676 172
\define@newfont: Redefine \typeout latex/1676 165
\wrong@fontshape: Support \@wrong@font@char latex/1676 170

1995-11-17 ltoutenc.dtx v1.7p
\UseTextSymbol: Support \@wrong@font@char latex/1676 111

1995-11-18 ltoutenc.dtx v1.7q
\UseTextSymbol: Modify message slightly 111

1995-11-21 fontdef.dtx v2.2n
General: Incorporate changed figures, as in plain.tex 280

1995-11-27 ltfssbas.dtx v3.0n
\nfss@catcodes: Reset hash, for definitions in fd files 167

1995-11-28 ltfloat.dtx v1.1n
General: documentation fixes . . . 450

1995-11-28 ltfsstrc.dtx v3.0g
General: documentation fixes . . . 195

1995-11-28 ltoutenc.dtx v1.7r
General: Added math mode checks to text commands. 106
doc fixes 102
Renamed \@changed@x@err to \TextSymbolUnavailable. . . 106

1995-11-29 ltoutenc.dtx v1.7t
General: Added \textasciicircum, \textasciitilde, \textbackslash, \textbar, \textgreater and \textless. 121
Added \textasciicircum, \textasciitilde, \textregistered and \texttrademark. 115
Added \textbackslash and \textbar. 115, 125
Added \textless and \textgreater. 115, 126

1995-12-01 ltoutenc.dtx v1.7u
General: Made \SS a Default, rather than having the default point to the OT1 definition. 115

1995-12-04 ltspace.dtx v1.2k
\nobreakspace: (Macro added . . 84

1995-12-04 ltspace.dtx v1.2l
\@xobeysp: (braces added to definition of tilde 84

1995-12-04 preload.dtx v2.4e
General: Ulrik Vieth. added 12pt OMS and OML preloads /1989 286

1995-12-05 ltdefns.dtx 1.2w
\@unexpandable@protect: Removed \unexpandable@noexpand as never used. internal/1733 . . . 45

1995-12-05 ltfiles.dtx v1.1c
\document: \ignorespaces added for latex/1933 92

1995-12-05 ltfloat.dtx v1.1n
\@textsuperscript: Use \ensuremath for latex/1984. 467

1995-12-05 ltoutenc.dtx v1.7v
\@inmathwarn: Changed \TextSymbolUnavailable text 107

1995-12-06 ltfssbas.dtx v3.00
\nfss@catcodes: Reset hat, for typeouts etc in fd files 167

1995-12-07 ltbibl.dtx v1.1l
\@citex: Restored name of \G@refundefinedtrue 472

1995-12-07 ltfloat.dtx v1.1m
\@textsuperscript: Move \m@th out of the \ensuremath for latex/1984. 467

1995-12-07 ltxref.dtx v1.1i
\@setref: Switch for refundefined restored 331
\G@refundefinedtrue: Renamed (back) from \G@refundefined 330

1995-12-11 ltoutenc.dtx v1.7w
General: Modified `\copyright` . . 115

1995-12-13 ltdefns.dtx 1.2x
`\-`: Documentation changed. . . . 51

1996-01-10 ltfiles.dtx v1.1d
`\@iffileonpath`: Change argument handling to not require doubled hash. latex/2024 97

1996-01-20 ltidxglo.dtx v1.1e
`\makeglossary`: Make no-op after use pr/2048 470
`\makeindex`: Make no-op after use pr/2048 470

1996-01-20 ltspace.dtx v1.2m
`\vspace`: Made robust 83

1996-03-25 ltmath.dtx v1.1a
`\@ensuredmath`: Macro added for amslatex/2104 357
`\ensuremath`: Reimplement for amslatex/2104 357

1996-04-18 ltpage.dtx v1.0i
General: Improve documentation 475

1996-04-22 ltmiscen.dtx v1.1c
General: Improve Documentation 333

1996-04-22 ltspace.dtx v1.2n
General: Documentation Improvements 71

1996-04-22 lttab.dtx v1.1g
`\@tabclassz`: (DPC) Extra `\hskip` keeps tabcolsep in empty columns internal/2122 408

1996-04-23 ltcounts.dtx v1.1d
General: Documentation improvements 146

1996-04-24 ltfiles.dtx v1.1e
`\document`: (DPC) Reset `\AtBeginDocument` eg for latex/1297 91

1996-05-08 ltfsstrc.dtx v3.0h
`\math@egroup`: Use `\bgroup` instead of `\begingroup` to match a kernel change made in 1994!! 205

1996-05-09 ltfntcmd.dtx v3.3t
`\check@icr`: Default definitions added 292

1996-05-17 fontdef.dtx v2.2o
General: `\@@sqrt` removed, at last 264, 280

1996-05-17 ltfiles.dtx v1.1f
`\nofiles`: added `\write` to `\protected@write` for latex/2146 93

1996-05-18 ltoutenc.dtx v1.7x
General: Produce error if encoding not found. pr/2054 143

1996-05-21 ltoutenc.dtx v1.7y
General: Corrected error message (CAR) 143

1996-05-21 ltsect.dtx v1.0s
`\@sect`: (DPC) Added extra braces for internal/2148 442
(DPC) Moved brace to allow commands like `\MakeUppercase` in 6th argument. Changed `\par` to `\endgraf` to allow non-long commands. internal/2148 . . 442
`\@ssect`: (DPC) Added extra braces for internal/2148 445
(DPC) Moved brace to allow commands like `\MakeUppercase` in 4th argument. Changed `\par` to `\endgraf` to allow non-long commands. internal/2148 . . 444

1996-05-23 ltoutenc.dtx v1.7z
`\@strip@args`: `\expandafter` added to match other changes for latex/2133 111
`\add@accent`: macro added. latex/2133 108
`\DeclareTextAccent`: Reimplemented using `\add@accent` to save space latex/2133 108
`\DeclareTextCompositeCommand`: Modified to cope with new `\add@accent` command: required removal of check for one argument-command . . . 109

1996-05-24 ltoutput.dtx v1.1t
`\@specialoutput`: Check that `\@colroom` is less than `\vsize`, indicating that a float has been added 496
Cut-off point changed to 1.5`\baselineskip` 496
`\@topnewpage`: Cut-off point changed to 2.5`\baselineskip` 495

1996-05-25 ltoutput.dtx v1.1u
`\@specialoutput`: Correct the above check 496

1996-06-03 ltmiscen.dtx v1.1d
`\@verbatim`: Exchanged the following two code lines so that `\dospecials` cannot reset the

category code of characters handled by \@noligs. 344

General: Move setting of verbatim font and \@noligs. 333

\verb: Put setting of verbatim font after \dospecials so that \dospecials cannot reset the category code of characters handled by \@noligs. 346

1996-06-10 ltboxes.dtx v1.0y

\@parboxto: (DPC) Changed \endgraf to \@@par 383

1996-06-10 ltsect.dtx v1.0t

\@sect: (DPC) Changed \endgraf to \@@par 442

\@ssect: (DPC) Changed \endgraf to \@@par 444

1996-06-13 ltdirchk.dtx v1.0r

General: documentation improvements mainly from internal/2174 1

1996-06-14 lttab.dtx v1.1h

\@tabclassz: (DPC) Change both\z@skip to 1sp for latex/2160 408

1996-06-22 ltspace.dtx v1.2o

General: Documentation of problems added 71

1996-07-10 ltfinal.dtx v1.0y

\toks: Free up memory from scratch registers /2213 625

1996-07-19 ltoutenc.dtx v1.8a

\@strip@args: Use char 0 not @ as carrier for \lowercase /2197 111

1996-07-26 ltboxes.dtx v1.0z

\if@minipage: put \global into definition 385

1996-07-26 ltclass.dtx v1.0u

\@classoptionslist: made only preamble 554

\@unusedoptionlist: made only preamble 555

1996-07-26 ltdefns.dtx v1.2y

\@reargdef: third arg picked up by \@yargdef 41

\renew@command: use \noexpand instead of \string 41

use \relax in place of empty arg 41

\renew@environment: use \relax in place of empty arg 42

1996-07-26 ltfloat.dtx v1.1n

\@endfloatbox: remove unecessary \global before \@minipage. 459

\@savemarbox: remove unecessary \global before \@minipage. 462

\@setminipage: remove unecessary \global before \@minipage. 457

\@setnobreak: remove unecessary \global before \@nobreak. . . . 457

1996-07-26 ltfssbas.dtx v3.0p

\@DeclareMathSizes: use faster \if test 160

\nfss@catcodes: omit \relax as not needed 167

1996-07-26 ltfssdcl.dtx v3.0e

\init@restore@version: Removed \ifrestore@version switch and replaced by \init@restore@version . . . 226

1996-07-26 ltfsstrc.dtx v3.0i

\init@restore@glb@settings: macro added replacing \if@inmath switch 204

1996-07-26 ltlists.dtx v1.0l

\@item: Remove unecessary \global before \@minipage. 372

Remove unecessary \global before \@nobreak. 373

1996-07-26 ltmath.dtx v1.1b

General: Removed \global before \@ignoretrue in various places. 348

1996-07-26 ltmiscen.dtx v1.1e

\@ignorefalse: put \global into definition 334

\begin: remove \global before \@ignore. 338

\end: remove \global before \@ignore. 340

\ignorespacesafterend: user level macro added 334

1996-07-26 ltoutput.dtx v1.1v

\@testfp: remove \global before \@test. 538

\@xtryfc: remove \global before \@test. 513

\@ztryfc: remove \global before \@test. 514

General: put \global into definition 488

remove \global before \@test. 488, 489

\clearpage: add number of missing percents 491

`\@iiiminipage`: Use it or lose it (@setminpage): Frank will want to lose it 385

1996-10-24 ltfloat.dtx v1.1p
- `\@floatboxreset`: Added local settings of flags: dangerous! 457
- `\@marginparreset`: Added local settings of flags: dangerous! 463
- `\@xfloat`: Added `\@nodocument` to trap floats in the preamble 454

1996-10-24 ltoutput.dtx v1.1z
- `\@addtocurcol`: Added `\nobreak`, etc as appropriate 518, 522
- `\@specialoutput`: Added `\nobreak` as appropriate 498
- `\@topnewpage`: Added `\@nodocument` to trap `\twocolumn` in the preamble 493
- `\newpage`: Better checks for noskipsec and inlabel added, plus nobreak 492

1996-10-25 ltlists.dtx v1.0n
- `\endtrivlist`: Change `\indent` to `\leavevmode` 370
  - Reset flags explicitly 370

1996-10-25 ltoutput.dtx v1.2a
- `\newpage`: Reset all flags explicitly 492

1996-10-26 ltlists.dtx v1.0o
- `\endtrivlist`: Correct typo 370

1996-10-27 ltoutenc.dtx v1.8c
- `\@strip@args`: Removed macro 109
- General: Added `\r` A 119
  - Added `\textasteriskcentered` 115, 125
  - Corrected syntax descriptions 103
  - Removed `\aa` and `\AA` 115, 119, 121

1996-10-28 ltplain.dtx v1.1u
- General: (CAR) More doc changes 15
- `\dotfill`: Removed math mode 30

1996-10-29 ltplain.dtx v1.1v
- `\dotfill`: Got arithmetic correct (CAR) 30

1996-10-29 ltspace.dtx v1.2u
- `\@gnewline`: Added macro 76
- `\@no@lnbk`: Macro replaces `\@lnbk` and `\@nolnbk` 75
- `\\`: Corrected and rationalised code 75
- `\nolinebreak`: Reimplemented both using `\@no@lnbk` 74

1996-10-31 ltfinal.dtx v1.0z
- General: Added extra `\lcode`, hoping it does no harm in T1 (pr/1969) 616, 623

1996-10-31 ltlists.dtx v1.0p
- `\@trivlist`: Added check for missing item in outer list 369

1996-10-31 ltsect.dtx v1.0y
- General: Corrected and tidied documentation; removed long lines 438

1996-11-03 ltplain.dtx v1.1w
- `\dotfill`: Saved tokens by using `\hb@xt@` 30

1996-11-04 lterror.dtx v1.2m
- `\@nodocument`: Always define `\@nodocument` in kernel, so that it can be cleared by `\document`. 66

1996-11-04 ltlists.dtx v1.0q
- `\@trivlist`: Moved check for missing item: only checked when not inlabel flag is false 369

1996-11-05 ltfiles.dtx v1.1i
- `\nofiles`: Standard `\if@nobreak` test added 93

1996-11-09 ltmath.dtx v1.1c
- `\@ensuredmath`: Made long, as it was before. /2104 357

1996-11-18 ltfssbas.dtx v3.0s
- `\define@newfont`: (DPC) lowercase fd file names. internal/1044 166

1996-11-18 ltoutenc.dtx v1.8d
- General: (DPC) lowercase external file names. internal/1044 143

1996-11-20 fontdef.dtx v2.2p
- General: lowercase fd and enc.def file names /1044 264

1996-11-20 ltvers.dtx v1.0f
- General: Check for old format modified /2319 33

1996-11-23 ltoutenc.dtx v1.8e
- General: Corrected description 103
  - Extended description 104

1996-11-28 ltvers.dtx v1.0g
- General: Check for old format modified /2319 33

1996-12-06 ltdirchk.dtx v1.0u
- `\IfFileExists`: *** removed from various messages for GNU Make. internal/2338 10

1996-12-06 ltfloat.dtx v1.1r
- `\@caption`: Call `\@setminpage` if needed. latex/2318 453

1996-12-06 ltfssini.dtx v3.0h
- General: (DPC) Remove *** from messages internal/2338 263

1996-12-17 ltclass.dtx v1.0w
- `\g@addto@macro`: Use `\begingroup` to save making a mathord . . 567

1996-12-20 ltsect.dtx v1.0z
- `\@dottedtocline`: Added `\nobreak` for latex/2343 . . . 448

1997-01-08 fontdef.dtx v2.2q
- General: Use `\DeclareMathDelimiter` to set delimiter codes 273
- `\mathparagraph`: Define using `\DeclareMathSymbol` 282

1997-01-08 ltfiles.dtx v1.1j
- `\@include`: reset `\deadcycles` latex/2365 95

1997-01-08 ltmath.dtx v1.1d
- `\root`: (DPC) Remove spurious space tokens from plain TeX definition /2359 350

1997-02-05 ltclass.dtx v1.0x
- `\g@addto@macro`: missing percent /2402 567

1997-02-21 ltlists.dtx v1.0r
- `\@item`: `\ifvoid` check added for `\noindent`. latex/2414 372

1997-03-21 ltcounts.dtx v1.1e
- `\fnsymbol`: Use `\mathsection` and `\mathparagraph`. latex/2445 150

1997-04-14 ltfiles.dtx v1.1k
- `\document`: Set the document space factor defaults. latex/2404 91
- `\normalsfcodes`: Macro added (from patch file) latex/2404 . . 93

1997-04-14 ltoutput.dtx v1.2b
- `\@writesetup`: Call `\normalsfcodes` (from patch file) latex/2404 507
  - Move `\label` and `\index` (from patch file) 507

1997-04-24 ltbibl.dtx v1.1m
- `\@citex`: `\@empty` to avoid primitive error on empty cite keys. latex/2432 472

1997-04-30 ltoutenc.dtx v1.9a
- General: Changed `\textsc` to `\scshape` 116
  - Introduced `\textcopyright` and modified `\copyright` 115
  - Introduced `\textcopyright` and modify `\copyright` 116
  - Modified `\textunderscore`, removing `\mathunderscore` . 116
  - Modified `\underscore`, removing `\mathunderscore` . 116

1997-04-30 ltoutenc.dtx v1.9b
- General: Added `\leavevmode` to `\textunderscore` 116

1997-05-04 ltoutenc.dtx v1.9c
- General: Added ‘hex index tabs’ . 122
  - Added TS1 encoding v2.2.beta 128

1997-05-07 ltoutenc.dtx v1.9d
- General: Added `\leavevmode` to `\textcompwordmark` 116

1997-05-07 ltspace.dtx v1.2v
- `\newline`: Made completely robust. 76

1997-05-29 ltfsstrc.dtx v3.0j
- General: Replaced `\\` by `\MessageBreak`, as suggested by Donald Arseneau. 197

1997-05-29 ltlogos.dtx v1.1f
- `\LaTeXe`: Added `\m@th` so that the LaTeX 2ε logo works with non-zero values of `\mathsurround`. 87

1997-06-16 ltdirchk.dtx v1.0v
- General: documentation improvements mainly from internal/2520 1

1997-06-16 ltfloat.dtx v1.1s
- General: documentation fixes . . . 450

1997-06-16 ltfntcmd.dtx v3.3v
- General: Fix typo in documentation. 288

1997-08-05 ltoutenc.dtx v1.9e
- General: Corrected order of arguments in `\UseTextSymbol` example. 103

1997-08-29 ltoutenc.dtx v1.9f
- General: Added OT4 encoding, provided by Marcin Woliński. 102

1997-09-09 ltdefns.dtx v1.2z
- `\provide@command`: Use `\begingroup` to avoid generating math ords if used in math mode. pr/2573 43

1997-09-15 ltpictur.dtx v1.1g
- `\@getcirc`: Warn if lines become invisible pr/2524 427
- `\@picture@warn`: Macro added pr/2524 427
- `\@sline`: Warn if lines become invisible pr/2524 418

1997-10-06 ltcounts.dtx v1.1f
- `\@Roman`: Change `\@Roman` to be fully expandable, so that the

1998-01-12 ltoutenc.dtx v1.9k
- General: Added \ProvidesPackage for textcomp.sty 102
  - Adding missing braces and \ushape. 130

1998-01-16 ltoutenc.dtx v1.9m
- General: fixed decimal codes. latex/2734 126

1998-03-04 ltdefns.dtx v1.2z2
- \@xargdef: Unnecessary \expandafter removed: pr/2758 39

1998-03-05 ltoutenc.dtx v1.9n
- General: Added masc/fem ords as in pr/2579 116

1998-03-20 ltdefns.dtx v1.2z3
- \@thirdofthree: Macro added 44

1998-03-20 ltoutenc.dtx v1.9o
- General: Documentation added about order of decls 105
  - Documentation added for pr/2783 104
- \UndeclareTextCommand: Macro added for pr/2783 113

1998-03-20 lttextcomp.dtx v1.9o
- General: Added various \UndeclareTextCommand declarations for pr/2783 324
  - Load decls after defaults for speed. 324

1998-03-21 ltclass.dtx v1.0z
- General: Added to documentation of filecontents 550

1998-03-21 ltclass.dtx v1.1a
- \@providesfile: Allow &. Internal/2702 559
- General: Correct to new onlypreamble command list 582

1998-03-25 ltfssbas.dtx v3.0u
- \showhyphens: Suppress unnecessary error when used in preamble 175

1998-04-11 fontdef.dtx v2.2t
- General: Added \mathring accent (pr2785) 280

1998-04-15 fontdef.dtx v2.2u
- General: Use new syntax for \DeclareMathDelimiter 273

1998-04-15 ltfssdcl.dtx v3.0h
- \@xxDeclareMathDelimiter: Macro added (pr/2662) 240

1998-04-17 fontdef.dtx v2.2v
- General: Reinsert symbol defs for < and > chars. 274

1998-04-18 fontdef.dtx v2.2w
- General: Reinsert symbol def for / char. 274

1998-05-07 ltclass.dtx v1.1b
- \@fileswithoptions: Modify help message for latex/2805 566

1998-05-18 lttab.dtx v1.1j
- \@endpbox: Use \setlength to set \hsize, so that the changes in the calc package apply here. 410
- \tabular*: Use \setlength, so that calc extensions apply. 400

1998-05-20 ltfinal.dtx v1.1b
- General: Set up lccodes before loading hyphenation files: pr/2639 616
  - Set up uc/lccodes after loading hyphenation files: pr/2639 622

1998-05-28 lterror.dtx v1.2n
- \@notdefinable: Added message re 'end...' pr/1555 65

1998-06-04 ltboxes.dtx v1.1c
- \@rule: Support calc-expressions 386

1998-06-12 ltoutenc.dtx v1.9p
- General: Corrected 130 and 131, see pr/2834 131
  - Renamed \textmacron pr/2840 132

1998-06-12 ltoutenc.dtx v1.9q
- \add@accent: Explicitly set \spacefactor after \accent (pr/2877) 109

1998-06-12 lttextcomp.dtx v1.9p
- General: Renamed \textmacron pr/2840 320

1998-06-18 lttab.dtx v1.1k
- General: Small addition to documentation 389

1998-07-06 lttab.dtx v1.1l
- General: Small correction to documentation 389

1998-08-17 ltboxes.dtx v1.1e
- General: (RmS) Minor Documentation fixes. 376

1998-08-17 ltclass.dtx v1.1c
- General: (RmS) Minor documentation fixes. 550

1998-08-17 ltdirchk.dtx v1.0w
- General: (RmS) Documentation improvements. 1

1998-08-17 ltfntcmd.dtx v3.3x
- General: (RmS) Minor documentation fixes. 288

2001-01-07 ltoutput.dtx v1.2j
\@writesetup: And do it in the right macro (pr/3286) 506

2001-02-16 ltxref.dtx v1.1k
\@newl@bel: Added an extra grouplevel (PR3250), jlb . . . 331

2001-05-25 ltclass.dtx v1.1d
\@providesfile: Explicitly set catcode of \endlinechar to 10 (pr/3334) 559

2001-05-25 ltdirchk.dtx v1.0x
General: Explicitly set catcode of \endlinechar to 10 (pr/3334) . 4

2001-05-28 ltoutenc.dtx v1.93
General: Added composites for compatibility with T1, pr/3295 120
Changed the effect of \.\i, pr/3295 122

2001-06-02 fontdef.dtx v2.2y
General: Provide default cfg files (pr/3264) 283

2001-06-04 fontdef.dtx v2.2z
General: Guard against math active equal and pipe sign in \models (pr/3333) 279
Guard against math active equal sign in \Relbar (pr/3333) . . 279

2001-06-04 ltclass.dtx v1.1e
\@providesfile: But only if it is a char (pr/3334) 559

2001-06-04 ltdirchk.dtx v1.0y
General: But only if it is a char (pr/3334) 4

2001-06-04 ltpictur.dtx v1.1j
\@sline: Don't warn for exactly zero pr/3318 418

2001-06-04 ltvers.dtx v1.0i
General: Check for old format disabled 33

2001-06-05 ltoutenc.dtx v1.94
General: Text composite Commands need kludges for ',' – see tlb1903.lvt 120

2001-08-26 ltclass.dtx v1.1f
\@providesfile: Readded setting of space char (pr/3353) 559

2002-02-24 ltplain.dtx v1.1x
\loggingall: Macro added 30
\loggingoutput: Macro added . . 30
\showoutput: Use newly added \loggingoutput 30
\tracingall: Use newly added \loggingoutput 30

2002-06-16 ltoutenc.dtx v1.95
General: Added \textbardbl (pr/3400) 125
Added default for \textbardbl (pr/3400) 115

2002-06-17 ltoutenc.dtx v1.95
General: Corrected \c for T1 (pr/3442) 120
Definition of \textexclamdown changed (pr/3368) 118
Definition of \textquestiondown changed (pr/3368) 118

2002-06-18 ltoutenc.dtx v1.95
General: Changed def for \textregistered to avoid small caps (pr/3420) 116

2002-10-01 ltfloat.dtx v1.1v
\thempfootnote: Use braces around \itshape to keep font change local (pr/3460). 466

2002-10-02 ltfssbas.dtx v3.0x
\DeclareFontSubstitution: Adding \LastDeclaredEncoding introduced a bug as on some occasions that macro name was stored in the internal lists instead of the actual encoding. (pr/3459) 158

2002-10-28 ltlists.dtx v1.0s
\endtrivlist: Check for math mode (pr/3437) 370

2002-10-28 ltoutenc.dtx v1.96
General: coding change, to follow bug fix by DEK in plain.tex (pr/3469) 119, 128

2002-12-13 ltbibl.dtx v1.1n
\@citex: Added \leavevmode in case citation is at start of paragraph (pr/3486) 472

2003-01-01 ltfntcmd.dtx v3.3z
General: Code checked and documentation extended by Chris 290

2003-05-18 ltbibl.dtx v1.1o
\nocite: Check if we are after \document 473

2003-08-27 ltpictur.dtx v1.1k
\@bezier: added missing displacement pr/3566 432
\@sline: check for \@linechar being empty pr/3570 418

2003-10-13 ltfinal.dtx v1.1e
- General: Added extra `\lccode` for `\-` and `\textcompwordmark` . 617

2003-12-16 ltoutput.dtx v1.2k
- `\@makecol`: Ensure that `\@elt` has a defined state (pr/3586) . . . 502

2003-12-30 ltpictur.dtx v1.1j
- `\@getcirc`: issue warning if circle size can't be met pr/3473 . . 427

2004-01-03 ltoutenc.dtx v1.99b
- General: Added `\textogonekcentered` (pr/3532) 120
- Added composites for `\k` (pr/3532) 125
- Use `\ooalign` for `\k` (pr/3532) 120

2004-01-04 ltbibl.dtx v1.1p
- `\nocite`: Changed error message 473

2004-01-04 ltoutenc.dtx v1.99c
- General: More adjustments for ogonek (pr/3532) 120

2004-01-23 ltdefns.dtx v1.1g
- `\@newenva`: Use kernel version of `\@ifnextchar` (pr/3501) 42
- `\@testopt`: Use kernel version of `\@ifnextchar` (pr/3501) 40
- `\@xargdef`: Use kernel version of `\@ifnextchar` (pr/3501) 39
- `\@xdblarg`: Use kernel version of `\@ifnextchar` (pr/3501) 51

2004-01-23 ltdefns.dtx v1.3g
- `\kernel@ifnextchar`: Added macro (pr/3501) 50

2004-01-28 ltclass.dtx v1.1g
- `\@providesfile`: Use kernel version of `\@ifnextchar` (pr/3501) 559

2004-01-28 ltvers.dtx v1.0k
- General: Check for old format made 5 years (pr/3601) 33

2004-02-02 fontdef.dtx v2.3
- General: Many things from here on made robust 278

2004-02-02 ltoutenc.dtx v1.99
- General: Added `\textbigcircle` 125

2004-02-04 fontdef.dtx v2.3a
- General: Added bigtriangle synonyms for stmaryrd 276

2004-02-04 ltspace.dtx v1.3
- `\nobreakdashes`: (Macro added . 84

2004-02-06 ltoutenc.dtx v1.99d
- `\@inmathwarn`: New command added to fix severe bug: pr/3563 106

2004-02-07 ltoutput.dtx v1.2l
- `\@doclearpage`: Empty kludgeins box if necessary, pr/3528 . . . 500

2004-02-13 ltoutenc.dtx v1.99e
- General: Documentation fixes: typos 102

2004-02-15 ltbibl.dtx v1.1q
- `\@cite@ofmt`: Added hook with default value `\hbox` 474
- `\@citex`: Changed to use a hook with default value `\hbox` . . . 473

2004-02-15 ltspace.dtx v1.3a
- `\nobreakdashes`: (Added spacefactor setting 84

2004-10-20 ltoutput.dtx v1.2m
- `\@makecol`: Removed dead code . 501

2005-07-27 ltfssdcl.dtx v3.0j
- `\DeclareMathAlphabet`: (MH) Make document commands robust 232
- `\DeclareSymbolFontAlphabet`: (MH) Make document commands robust 244
- `\new@mathalphabet`: (MH) Make document commands robust 233
- `\non@alpherr`: (MH) Change because command is now properly robust 226
- `\SetMathAlphabet`: (MH) Make document commands robust 234

2005-09-27 ltoutenc.dtx v1.99g
- General: Replace `\sh@ft` by `\ltx@sh@ft` 118, 120, 127

2005-09-27 ltplain.dtx v1.1y
- `\ltx@sh@ft`: New macro 29
- `\sh@ft`: Macro no longer used but left for compatibility 29

2005-11-08 ltoutenc.dtx v1.99h
- General: Added `\ij` and `\IJ` from babel. (pr/3771) . . 115, 119, 121

2005-11-10 ltmath.dtx v1.1g
- `\[`: (MH) Fixed potential problem in `\[` (pr/3399). 354
- General: (MH) Minor documentation fixes. 348

2006-05-18 ltboxes.dtx v1.1g
- `\@parboxto`: Ensure `\@parboxto` holds the value of `\@tempdimb` not the register itself (pr/3867) 383

2006-09-13 ltoutput.dtx v1.1m
- General: Ensure that rule is in `\normalcolor` 545

2007-08-05 ltclass.dtx v1.1h
- \@fileswithoptions: Prevent loss of brackets PR/3965 564

2007-08-06 ltcntrl.dtx v1.0h
- \@fornoop: Really make defs long 58

2007-08-31 ltfssdcl.dtx v3.0l
- \SetSymbolFont@: Font warning changed to info for encoding change (pr/3975) 231

2009-09-24 ltvers.dtx v1.0l
- General: Stop checking for old format 33

2009-10-20 ltfssdcl.dtx v3.0m
- \in@: More robust thanks to Heiko. 222

2009-10-28 lttextcomp.dtx v1.99k
- General: Added Latin Modern and TeX Gyre subsets 326

2009-11-04 lttextcomp.dtx v1.99l
- General: Added more Latin Modern and TeX Gyre subsets 326

2009-12-14 ltfntcmd.dtx v3.4a
- \ifmaybe@ic: Macro added 293
- \maybe@ic@: Use switch \ifmaybe@ic instead of \if@tempswa 293
- \t@st@ic: Use switch \ifmaybe@ic instead of \if@tempswa 293

2010-08-17 ltmiscen.dtx v1.1k
- \enddocument: Use braces around \input arg (pr/4124) 335

2010-08-17 ltmiscen.dtx v1.1l
- \enddocument: Change of plan: use \@@input instead (pr/4124) 335

2011-05-08 ltfssdcl.dtx v3.0n
- \in@: Simplified thanks to Bruno. 222

2011-08-19 ltclass.dtx v1.1i
- \@ifclasswith: Re-jig definition after more stringent \in@ test. 557

2011-09-03 ltfssdcl.dtx v3.0o
- \new@mathversion: (Will) Remove \global before \newcount (unnecessary and caused etex bug). 229

2012-01-20 ltplain.dtx v2.0b
- \loggingall: etex tracing if available 30

2013-07-07 ltclass.dtx v1.1i
- General: Correctly describe how the date in \@ifpackagelater is used 553

2014-04-18 ltoutput.dtx v1.1o
- General: Handle infinite glue from \enlargethispage (pr/4023) 545

2014-04-24 ltoutput.dtx v1.2n
- \fl@tracemessage: Renamed internal trace commands; provide as package 535

2014-04-27 ltfloat.dtx v1.2b
- \end@dblfloat: Inline the code to allow some coexistence with packages that hook into \end@float and do not know about the algorithm change 458

2014-06-10 ltfloat.dtx v1.2b
- \end@dblfloat: missing \fi added 458

2014-12-30 ltfinal.dtx v2.0a
- \newmarks: macro added 612
- \newXeTeXintercharclass: macro added 612

2014-12-30 ltfloat.dtx v1.2a
- \@textsubscript: Command added (latexrelease) 467
- \textsubscript: Command added (latexrelease) 467

2014-12-30 ltfssbas.dtx v3.0y
- \mathgroup: move allocation to ltplain. 154

2014-12-30 ltoutput.dtx v1.2m
- General: Command updated (latexrelease) 544

2014-12-30 ltplain.dtx v2.0a
- \e@alloc: macro added 20
- \e@alloc@chardef: macro added 19
- \e@alloc@top: macro added 19
- \e@ch@ck: macro added 20
- \extrafloats: macro added 20
- \newlanguage: New engine-specific allocation scheme (latexrelease) 18

2014-12-30 ltspace.dtx v1.3b
- \@: \@ discards spaces when moving (pr3039)(latexrelease) 85

2015-01-03 ltdefns.dtx v1.4a
- \typein: use modified definition in luatex 38

2015-01-03 ltdirchk.dtx v1.1
- General: Enable extra primitives when LuaTEX is used 3

2015-01-03 ltfinal.dtx v2.0a
- General: Skip resetting codes with Unicode engines 622
- Unicode data loading added 614

2015-01-07 ltvers.dtx v1.0n
- \IncludeInRelease: macro added 34

2015-01-08 ltboxes.dtx v1.1h
\framebox: Make Robust (latexrelease) 381
\makebox: Make Robust (latexrelease) 377
\parbox: Make Robust (latexrelease) 382
\raisebox: Make Robust (latexrelease) 387
\rule: Make Robust (latexrelease) 386
\savebox: Make Robust (latexrelease) 379

2015-01-08 ltdefns.dtx v1.4a
\MakeRobust: Added macro 47

2015-01-08 ltlength.dtx v1.1c
\setlength: to ensure first length argument is terminated. (latexrelease) 153

2015-01-08 ltmath.dtx v1.1h
\): Make Robust (latexrelease) . . 354
\]: Make Robust (latexrelease) . . 354

2015-01-09 ltfssini.dtx v3.1a
\em: Allow \emph to produce small caps (latexrelease) 259
\eminnershape: macro added (latexrelease) 259

2015-01-09 ltspace.dtx v1.1h
\addpenalty: Donald Arseneau's fix from PR/377703 (latexrelease) 81

2015-01-10 ltcounts.dtx v1.1h
\@fnsymbol: Unse \TextOrMath (latexrelease) 151
\@stpelt: Reset all within counters in one go (latexrelease) 148

2015-01-11 ltcounts.dtx v1.1h
\TextOrMath: Add command to solve robustness issues (pr/3752) (latexrelease) 151

2015-01-11 ltfloat.dtx v1.2b
\@dblfloatplacement: float order in 2-column (latexrelease) . . 459
\@xfloat: Check for valid option (latexrelease) 454
\end@dblfloat: float order in 2-column (latexrelease) 458

2015-01-11 ltfssbas.dtx v3.0y
\@DeclareMathSizes: Allow arbitrary units (latexrelease) 160

2015-01-11 ltspace.dtx v1.3d
\@Esphack: Allow hyphenation (Donald Arseneau pr/3498) (latexrelease) 79
\@esphack: Allow hyphenation (Donald Arseneau pr/3498) (latexrelease) 78

2015-01-14 ltoutput.dtx v1.2n
\@addtocurcol: float order in 2-column (latexrelease) 516
\@addtodblcol: float order in 2-column (latexrelease) 527
\@addtonextcol: float order in 2-column (latexrelease) 523
\@doclearpage: Empty kludgeins box if necessary, pr/3528 . . . 499
float order in 2-column (latexrelease) 499
\@startdblcolumn: float order in 2-column (latexrelease) 511
\@xtryfc: float order in 2-column (latexrelease) 513
\@ztryfc: float order in 2-column (latexrelease) 514

2015-01-14 ltspace.dtx v1.3e
\addpenalty: Avoid adding redundant skips (DPC) 82

2015-01-17 ltvers.dtx v1.0m
\IncludeInRelease: modified with \@currname 34

2015-01-19 ltvers.dtx v1.0o
\IncludeInRelease: Optional argument 34

2015-01-20 ltoutput.dtx v1.2m
\fl@tracemessage: Reset \IncludeInRelease flags . . . 536

2015-01-22 ltvers.dtx v1.0p
General: Preserve any \everyjob material inserted by a loader (.ini file) 34

2015-01-23 ltfinal.dtx v2.0b
\newmarks: use reserved count 256 612
\newXeTeXintercharclass: use reserved count 257 612

2015-01-23 ltplain.dtx v2.0c
\extrafloats: reserve counts 256–265 20

2015-01-24 ltfinal.dtx v2.0c
General: Skip T1-code entirely with Unicode engines 614

2015-02-03 ltfinal.dtx v2.0d
General: Set \lccode for - with Unicode engines 615

2015-02-16 ltoutenc.dtx v1.99m
General: Added \textcommabelow latex/4414 117

2015-02-16 ltoutenc.dtx v1.99n
General: Added `\textcommaabove` 117
Added composites for ç 125
Added composites for `\c` 120

2015-02-16 lttextcomp.dtx v1.99m
General: Added lmtt (Heiko Oberdiek) latex/4415 326

2015-02-19 ltvers.dtx v1.0q
`\IncludeInRelease`: Swap argument order 34

2015-02-20 ltplain.dtx v2.0d
`\loggingall`: Spell commands correctly :-) 30

2015-02-21 ltdefns.dtx v1.4b
General: Removed autoload support 36

2015-02-21 lterror.dtx v1.2o
General: Removed autoload support 60

2015-02-21 ltfiles.dtx v1.1m
General: Removed autoload support 88

2015-02-21 ltfssbas.dtx v3.0z
General: Removed autoload code 154

2015-02-21 ltfsscmp.dtx v3.0d
General: Removed autoload code 218

2015-02-21 ltfssdcl.dtx v3.0p
General: Removed autoload code 222

2015-02-21 ltfsstrc.dtx v3.0k
General: Removed autoload code 195

2015-02-21 ltoutenc.dtx v1.99m
General: Removed autoload code 102

2015-02-21 ltoutput.dtx v1.2n
General: Removed autoload code 478
`\f@depth`: macro added(latexrelease) 498

2015-02-21 ltpictur.dtx v1.1k
General: Removed autoload code 411

2015-02-21 ltplain.dtx v2.0e
General: Removed autoload code 15

2015-02-21 lttab.dtx v1.1n
General: Removed autoload code 389

2015-02-21 ltvers.dtx v1.0r
General: Removed autoload code 33

2015-02-21 ltvers.dtx v1.0w
`\IncludeInRelease`: set `\@currname` empty here (in case `\IncludeInRelease` input early) 34

2015-02-22 ltfsscmp.dtx v3.0e
General: Moved all code into latexrelease - obsolete commands are no longer automatically part of the kernel 218

2015-03-02 ltplain.dtx v2.0f
`\e@mathgroup@top`: macro added 19
`\newlanguage`: allow 255 math groups in Unicode engines . . . 18

2015-03-10 ltplain.dtx v2.0g
`\hideoutput`: macro added 31
`\loggingall`: Reorganise to be less noisy 30
`\tracingnone`: macro added 31

2015-03-12 ltoutput.dtx v1.2m
General: initialise `\@dbldeferlist` again 490

2015-03-18 ltfssdcl.dtx v3.0q
`\DeclareSymbolFont`: Restrict Symbol fonts to 0-15 230
`\document@select@group`: Introduce `\e@mathgroup@top` 226
`\select@group`: Introduce `\e@mathgroup@top` 224

2015-03-26 ltfinal.dtx v2.0d
General: Use renamed `unicode-letters.def` 614

2015-04-07 ltfssbas.dtx v3.1a
`\wrong@fontshape`: Try loading fd file if family has changed . . . 170

2015-04-28 ltfinal.dtx v2.0f
`\newXeTeXintercharclass`: define `\xe@alloc@intercharclass` for compatibility with older xelatex initilisation 612

2015-05-10 ltlists.dtx v1.0t
`\@doendpe`: Explicitly reset `\clubpenalty` before clearing `\everypar`; see also pr/0462 and pr/4065 371

2015-06-19 ltfinal.dtx v2.0g
`\e@alloc@intercharclass@top`: Use −1 for first range to get contiguous allocation 612
`\newmarks`: Use −1 for first range to get contiguous allocation . 612

2015-06-19 ltplain.dtx v2.0h
General: delete spurious old definition of `\newtoks` 23
`\e@alloc`: extra braces in case arguments not single token . . 20
`\newlanguage`: Use −1 for first range to get contiguous allocation 18

2015-06-23 ltfinal.dtx v2.0h
General: set `\patch@level` in ltvers rather than in

ltfinal/ltpatch 624

2015-06-23 ltvers.dtx v1.0t
General: set `\patch@level` in ltvers rather than in ltfinal/ltpatch 33

2015-08-06 ltplain.dtx v2.0i
`\extrafloats`: Add `\string` in case argument is not an unexpandable primitive 21

2015-08-23 ltdirchk.dtx v1.2
General: Do not use luatex prefix . 3

2015-08-23 ltvers.dtx v1.0v
General: Allow negative patchlevel for pre-release 34

2015-08-30 ltplain.dtx v2.1a
`\newinsert`: new `\newinsert` implementation 22

2015-09-205 ltoutput.dtx v1.3a
General: extended `\@freelist` . . 489

2015-09-24 ltluatex.dtx v1.0a
`call_callback`: Function added . 607
`callback.register`: Function modified 603
`callback_descriptions`: Function added 609
`\catcodetable@atletter`: Macro added 592
`\catcodetable@initex`: Macro added 592
`\catcodetable@latex`: Macro added 592
`\catcodetable@string`: Macro added 592
`add_to_callback`: Function added 607
`remove_from_callback`: Function added 608
`new_attribute`: Function added . 599
`disable_callback`: Function added 609
`in_callback`: Function added . . . 609
`\newattribute`: Macro added . . . 591
`\newcatcodetable`: Macro added 591
`\newluabytecode`: Macro added . 594
`\newluachunkname`: Macro added 594
`\newluafunction`: Macro added . 593
`\newwhatsit`: Macro added 594
`module_error`: Function added . . 597
`module_info`: Function added . . . 597
`module_warning`: Function added 597
`modules`: Function modified 596
`create_callback`: Function added 606
`provides_module`: Function added 596
`luatexbase`: Table added 596

2015-10-02 ltdirchk.dtx v1.2a
General: Allow backing out of unprefixed names 3

2015-10-02 ltluatex.dtx v1.0b
General: Fix backing out of TEX code 595

2015-10-02 ltluatex.dtx v1.0c
General: Allow backing out of Lua code 595

2015-10-02 ltluatex.dtx v1.0e
`uninstall`: Function added 610

2015-10-03 ltluatex.dtx v1.0f
`provides_module`: use luatexbase_log 596

2015-10-27 ltplain.dtx v2.1b
`\extrafloats`: Use global assignment when switching to extended range 20

2015-11-07 ltspace.dtx v1.3f
`\@esphack`: Only space if there is no space at the end of the hlist latex/4443 78

2015-11-14 ltluatex.dtx v1.0g
General: Track LuaTEX changes for `(new)token.create` 598

2015-11-18 ltplain.dtx v2.2a
`\newlanguage`: Extended stream allocation in luatex (0.85) . . . 18

2015-11-19 ltplain.dtx v2.2b
`\newlanguage`: Only extend allocation of write streams (see luatex list) 18

2015-11-27 ltluatex.dtx v1.0h
`callback_descriptions`: Match test in in-callback latex/4445 609
`in_callback`: Guard against undefined list latex/4445 . . . 609

2015-11-29 ltluatex.dtx v1.0i
General: Declare this as local before used in the module error definitions (PHG) 596
`call_callback`: Check name is not nil in error message (PHG) . 607
`create_callback`: Check name is not nil in error message (PHG) 606

2015-12-02 ltluatex.dtx v1.0j
General: Adjust hashtokens to store the result of tex.hashtokens()), not the function (PHG) 598

- Assorted typos fixed (PHG) . . 589
- Declaration/use of first_head fixed (PHG) 597
- Remove nonlocal iteration variables (PHG) 589
- Remove unreachable code after calls to error() (PHG) 589

2015-12-02 ltluatex.dtx v1.0k
- General: resolve name and i.description (PHG) 605
- `call_callback`: Give more specific error messages (PHG) 607
- `add_to_callback`: Give more specific error messages (PHG) 607
- `remove_from_callback`: adjust initialisation of cb local (PHG) 608
  - Give more specific error messages (PHG) 608
- `create_callback`: Give more specific error messages (PHG) 606

2015-12-10 ltfinal.dtx v2.0i
- General: Use new common Unicode data loaders 614

2015-12-18 ltluatex.dtx v1.0l
- General: Load Unicode data from source 592

2016-01-04 ltfinal.dtx v2.0j
- General: Do not set up inter character classes for XeTeX . 614
- `\e@alloc@intercharclass@top`: Start allocation at one not three 612

2016-01-05 ltfinal.dtx v2.0k
- `\e@alloc@intercharclass@top`: Remove duplicated code . . . 612

2016-01-05 ltfinal.dtx v2.0l
- General: Correct latexrelease guards 614
  - Ensure old definitions for inter-character class toks are available using latexrelease . . 614
  - Missing brace 614

2016-01-05 ltfinal.dtx v2.0m
- General: Undefine XeTeX classes when using patching an older kernel 614

2016-01-05 ltfinal.dtx v2.0p
- General: Only apply XeTeX change if XeTeX is in use . . 614

2016-02-11 ltluatex.dtx v1.0m
- General: pdf_stream_filter_callback removed 603
  - process_rule, [hv]pack_quality append_to_vlist_filter added . 602
  - read_cidmap_file added 602
  - show_warning_message added . 603
  - token_filter removed 602

2016-02-18 ltfssdcl.dtx v3.0r
- `\@DeclareMathDelimiter`: Check for delimiter not `\delimiter` 240
- `\DeclareMathAccent`: Check for mathaccent not `\mathaccemt` 236
- `\DeclareMathRadical`: Check for radical not `\radical` 243
- `\DeclareMathSymbol`: Check for mathchar not `\mathchar` . . . 238

2016-03-13 ltluatex.dtx v1.0n
- General: contribute_filter added . 602
  - insert_local_par added 602

2016-03-29 ltpictur.dtx v1.1l
- `\@oval`: add setting of line tests . 428
  - initialise tests 428
- `\@ovhorz`: use glue not leaders if horizontal line not required . 428
- `\@ovvert`: use glue not leaders if vertical line not required . . . 428
- `\if@ovhline`: macro added (latex/4452) 427
- `\if@ovvline`: macro added (latex/4452) 427

2016-04-22 ltfinal.dtx v2.0q
- `\e@alloc@intercharclass@top`: XeTeX 0.99996 has 4096 char classes not 256 612

2016-06-19 ltoutenc.dtx v1.99m
- General: OT1 definition (was duplicate T1 definition) 120

2016-06-20 ltclass.dtx v1.1j
- `\@ifclasslater`: don't declare as `\@onlypreamble` 556

2016-07-29 ltplain.dtx v2.2c
- `\extrafloats`: use `\global` `\chardef` 21
- `\newinsert`: fix for tlb-newinsert-001 22

2016-10-02 ltclass.dtx v1.2a
- `\@ifclasswith`: Ignore spaces while checking for option clash 557
- `\ExecuteOptions`: Ignore spaces in argument 561

2016-10-15 ltdirchk.dtx v1.2b
- General: Require eTeX 4

2016-10-15 lterror.dtx v1.2p
- General: Require eTeX 60

2016-10-15 ltfinal.dtx v2.0r
- General: Require eTeX 612

- 2016-10-15 ltfinal.dtx v2.0s
  - General: Tidy up status of char 127 . . . 612
- 2016-10-15 ltfssini.dtx v3.1b
  - General: Require eTEX . . . 247
- 2016-10-15 ltplain.dtx v2.2d
  - General: Require eTEX . . . 15
- 2016-10-16 ltplain.dtx v2.3a
  - `\newlanguage`: Allow languages up to 16383 in luatex . . . 18
- 2016-10-19 ltcounts.dtx v1.1j
  - `\TextOrMath`: Test directly for `\protected` . . . 151
- 2016-11-06 ltplain.dtx v2.3b
  - General: Drop `\outer` entirely . . . 15
- 2016-11-09 ltclass.dtx v2.1b
  - `\@fileswithoptions`: Improve `\ifx` tests PR/4497 . . . 564
- 2016-11-17 ltluatex.dtx v1.0p
  - General: call_edit added . . . 603
- 2016-12-03 fontdef.dtx v3.0a
  - General: (DPC) Default to TU encoding for Unicode TeX engines . . . 266
  - `\shapedefault`: (DPC) Default to TU encoding for Unicode TeX engines . . . 270
- 2016-12-04 ltoutenc.dtx v2.0a
  - General: Added TU encoding . . . 133
- 2017-01-01 ltoutput.dtx v1.3b
  - General: make fpmin negative so ignored even if float height is negative . . . 544
- 2017-01-10 ltfssbas.dtx v3.2a
  - `\showhyphens`: Add version of `\showhyphens` that works with XeTEX. . . . 175
- 2017-01-23 ltoutenc.dtx v2.0b
  - General: Added TU specific commands in ASCII range pr/4500 . . . 133
- 2017-01-24 ltoutenc.dtx v2.0c
  - General: Declare TU composites for i and j . . . 133
  - Make `\textasteriskcentered` U+2217 not U+204E . . . 133
  - TeX ligature syntax for xetex and luatex reversed . . . 133
- 2017-01-24 ltoutenc.dtx v2.0d
  - General: Declare macron composites for YyGg . . . 133
- 2017-02-12 ltoutenc.dtx v2.0e
  - General: Declare fallback code for `\textasteriskcentered` . . . 133
- 2017-02-18 ltluatex.dtx v1.1c
  - `new_attribute`: Parameterise count used in tracking . . . 599
  - `new_bytecode`: Parameterise count used in tracking . . . 600
  - `new_chunkname`: Parameterise count used in tracking . . . 600
  - `new_whatsit`: Parameterise count used in tracking . . . 599
- 2017-02-19 ltoutenc.dtx v2.0f
  - General: add `\@empty` to guard against 3rd argument being empty . . . 119
  - declare composites with empty base for hat and tilde, use same slots for `\textasciicircum` ans `\textasciitilde` . . . 133
  - declare straight quotes using new `\remove@tlig` command . . . 133
- 2017-02-22 ltoutenc.dtx v2.0g
  - General: Fix typo introduced at 2.0f . . . 133
- 2017-02-24 ltoutenc.dtx v2.0h
  - General: introduce `\DeclareUnicodeAccent` . . . 133
  - `\DeclareTextCompositeCommand`: add check whether the accent command is defined for this encoding . . . 109
- 2017-03-08 ltclass.dtx v1.2c
  - `\@ifclasslater`: add `\@parse@version@dash` to support yyyy-mm-dd as well as yyyy/mm/dd . . . 556
- 2017-03-09 ltfinal.dtx v2.0t
  - `\l@nohyphenation`: ensure `\l@nohyphenation` is defined. 617
- 2017-03-09 ltmiscen.dtx v1.1m
  - `\@verbatim`: Use `\language` not `\hyphenchar` . . . 344
  - `\verb`: Use `\language` to stop hyphenation . . . 346
- 2017-03-10 ltfiles.dtx v1.1n
  - `\document`: Save language default 91
- 2017-03-10 ltoutput.dtx v1.3c
  - `\@writesetup`: Reset `\language` . 505
- 2017-03-13 ltdefns.dtx v1.5a
  - `\-`: Define `\-` in terms of `\hyphenchar` . . . 51
- 2017-03-27 ltdefns.dtx v1.5b
  - `\@dischyph`: Define `\@dischyph` after `\-` . . . 51

2017-03-28 ltluatex.dtx v1.1e
General: glyph_stream_provider added 603

2017-03-29 ltboxes.dtx v1.3a
\@arrayparboxrestore: Reset \lineskiplimit 384

2017-04-05 ltoutenc.dtx v2.0i
\DeclareTextCompositeCommand: Declare accent command if not already declared when declaring a composite. 109

2017-04-10 ltplain.dtx v2.3c
\newlanguage: Correction to code to skip write18 in luatex 18

2017-04-11 ltoutput.dtx v2.4a
\newpage: account for the depth of the last row of the page 492

2017-12-17 ltoutput.dtx v1.4b
\@addtonextcol: fix doc guards . 523

2018-01-06 ltdefns.dtx 1.5c
\@ifundefined: Avoid defining undefined commands to \relax 49

2018-02-18 ltclass.dtx v1.2d
\@ifclasslater: Added 0 up front to make bad data come out as 0. 556
General: Introduce rollback concept 574

2018-03-08 ltcounts.dtx v1.1k
\@ifbothcounters: Interface added 149
\@removefromreset: Interface added 148
\counterwithin: Interface added 149
\counterwithout: Interface added 149

2018-03-24 ltclass.dtx v1.2e
\pkgcls@use@this@release: Use full file name for old release . 580

2018-03-25 ltfinal.dtx v2.1a
General: default to UTF-8 618
\UseRawInputEncoding: Macro added 619

2018-03-27 ltclass.dtx v1.2f
\endfilecontents: Use full file name for old release 571

2018-04-06 ltfinal.dtx v2.1b
\UseRawInputEncoding: Undo changes to \DeclareFontEncoding@ and definition of \DeclareUnicodeCharacter . 619

2018-04-07 ltfinal.dtx v2.1c
\UseRawInputEncoding: Undefine \inputencodingname 619

2018-04-08 ltclass.dtx v1.2g
\@ifclasslater: Strip leading spaces from dates. 556

2018-04-08 ltclass.dtx v1.2h
\@onefilewithoptions: Pass expanded date 576

2018-04-08 ltfinal.dtx v2.1d
General: Delay full UTF-8 handling to \everyjob 619

2018-04-11 ltcounts.dtx v1.1l
\counterwithin: Correct default (issue/38) 149

2018-05-02 ltluatex.dtx v1.1g
General: find_sfd_file removed . . . 602
finish_synctex_callback added . 603
glyph_not_found added 603
read_sfd_file removed 602

2018-05-08 ltclass.dtx v1.2i
\pkgcls@parse@date@arg: Make suspicious rollback a warning not error: github issue 43 . . 577

2018-05-11 ltfinal.dtx v2.18
General: Make invalid UTF-8 also safe, for legacy filesystem encodings 619

2018-05-29 ltclass.dtx v1.2j
\endfilecontents: use \csname not \@undefined 572

2018-08-11 ltoutenc.dtx v2.0j
General: Provide \guillemetleft and \guillemetright . 122, 127, 135

2018-08-18 ltluatex.dtx v1.1h
General: append_to_vlist_filter is exclusive 602

2018-08-24 ltfinal.dtx v2.1f
\document@default@language: Add to latexrelease (github/68) 617

2018-09-02 ltsect.dtx v1.1b
\@dottedtocline: Prevent protrusion (https://tex.stackexchange.com/q/172785/10109) 448

2018-09-24 fontdef.dtx v3.0b
General: Start LR-mode if necessary (git/49) 282

2018-09-24 ltmath.dtx v1.2b
\smash: Start LR-mode if necessary (git/49) 351

- `\vphantom`: Start LR-mode if necessary (git/49) 351

2018-09-24 ltspace.dtx v1.3h
- `\enspace`: Start LR-mode if necessary (git/49) 85
- `\leavevmode@ifvmode`: Macro added (git/49) 85

2018-09-26 ltdefns.dtx v1.5e
- `\renew@command`: Always explicitly generate a space after the csname and not rely on `\noexpand` to save tokens (git/41) 41

2018-09-26 ltmiscen.dtx v1.1n
- `\@writefile`: Sometimes mask the endline char when writing to files (github/73) 337
- `\add@percent@to@temptokena`: Sometimes mask the endline char when writing to files (github/73) 336
- `\protected@file@percent`: Sometimes mask the endline char when writing to files (github/73) 336

2018-09-26 ltsect.dtx v1.1c
- `\addcontentsline`: Sometimes mask the endline char when writing to files (github/73) . 447

2018-10-10 ltspace.dtx v1.3i
- `\@esphack`: Don't introduce breakpoints if @nobreak is true and after sections 78

2018-10-11 ltmiscen.dtx v1.1o
- `\@setupverbvisiblespace`: Provide visible space in `\verb*` also for XeTeX and LuaTeX (github/69) 345
- `\@sverb`: Provide visible space in `\verb*` also for XeTeX and LuaTeX (github/69) 345
- `\@verbvisiblespacebox`: Provide visible space in `\verb*` also for XeTeX and LuaTeX (github/69) 345
- `\asciispace`: Provide visible space in `\verb*` also for XeTeX and LuaTeX (github/69) 344
- `verbatim*`: Provide visible space in `\verb*` also for XeTeX and LuaTeX (github/69) 346
- `\verbvisiblespace`: Provide `\verbvisiblespace` such that it is usable in normal text (github/70) 344
  - Provide visible space in `\verb*` also for XeTeX and LuaTeX (github/69) 344

2018-10-21 ltluatex.dtx v1.1i
- `new_luafunction`: Function added 601

2018-11-09 ltbibl.dtx LaTeX2e
- `\bibliography`: Zap spaces in the argument as BibTeX doesn't support them (github/88) . . 473

2018-11-18 ltoutenc.dtx v2.0k
- General: Provide `\Hwithstroke` and `\hwithstroke` 137

2018-11-19 ltoutenc.dtx v2.0k
- General: Added `\Hwithstroke` and `\hwithstroke` 121

2018-11-28 ltoutput.dtx v1.4d
- `\@combinedblfloats`: Unbox `\@outputbox` to preserve boxing level (github/94) . . . 510

2018-12-30 lttab.dtx v1.1p
- `\@tabclassz`: Add extra `\hskip` to guard against an `\unskip` at the start of a c-column cell (gh/102) 407

2019-02-07 ltfiles.dtx v1.1o
- `\@swaptwoargs`: Helper macro added 98
- `\InputIfFileExists`: Expand `\@filef@und` before executing second argument (github/109) 98

2019-02-07 ltfssbas.dtx v3.2b
- `\define@newfont`: Changed wording of warning (github/107) 166

2019-06-18 ltluatex.dtx v1.1j
- General: finish_synctex_callback renamed finish_synctex 603
  - font_descriptor_objnum_provider added 603
  - make_extensible added 603
  - new_graf added 602
  - page_objnum_provider added . 603
  - process_pdf_image_content added 603
  - wrapup_run added 603

2019-07-01 ltfiles.dtx v1.2a
- `\IfFileExists`: Support UTF-8 . 97
- `\includeonly`: Support UTF-8 . . 94
- `\set@curr@file`: Support UTF-8 96

- 2019-07-09 ltfssbas.dtx v3.2c
  - `\DeclareErrorFont`: Don’t set any `\f@...` macros 169
- 2019-07-09 ltfssini.dtx v3.1c
  - General: Explicitly set some defaults 262
- 2019-08-22 ltxref.dtx v1.1l
  - `\labelformat`: Commanded moved from `varioref.sty` 332
  - `\Ref`: Commanded moved from `varioref.sty` 332
  - `\refstepcounter`: Allow `\p@...` to have an argument 332
- 2019-08-27 fontdef.dtx v3.0c
  - General: Various commands made robust throughout the file . . 275
- 2019-08-27 ltboxes.dtx v1.3b
  - General: Various commands made robust 376
- 2019-08-27 ltdefns.dtx v1.5f
  - General: Make various commands robust 52
  - `\MakeRobust`: Make the assignments global as we may need to apply them inside a group 47
- 2019-08-27 ltfiles.dtx v1.2b
  - `\IfFileExists`: Make command robust 97
  - `\InputIfFileExists`: Make command robust 98
- 2019-08-27 ltfssbas.dtx v3.2d
  - General: Make various commands robust 154
- 2019-08-27 ltfssdcl.dtx v3.0s
  - `\DeclareMathAccent`: Make math accents robust 236
  - `\set@mathdelimiter`: Make math delimiters robust 242
- 2019-08-27 ltfssini.dtx v3.1d
  - General: Make various commands robust 247
- 2019-08-27 ltidxglo.dtx v1.1f
  - General: Make `\index` and `\gloassary` robust 469
- 2019-08-27 ltlength.dtx v1.1d
  - General: Make various command robust 153
- 2019-08-27 ltlogos.dtx v1.1j
  - `\TeX`: Make `\TeX` command robust 87
- 2019-08-27 ltmath.dtx v1.2c
  - General: Make various commands robust 348
- 2019-08-27 ltmiscen.dtx v1.1p
  - General: Make various commands robust 333
  - `\begin`: Make command robust . 338
  - `\end`: Make command robust . . . 340
- 2019-08-27 ltoutput.dtx v1.4e
  - `\@begindvibox`: Make `\AtBeginDvi` robust 490
- 2019-08-27 ltpage.dtx v1.0l
  - General: Make various commands robust 475
- 2019-08-27 ltpictur.dtx v1.1m
  - General: Make various commands robust 411
    - Remove several unnecessary `\gdef` definitions 411
- 2019-08-27 ltsect.dtx v1.1d
  - General: Make various commands robust 438
- 2019-08-27 ltspace.dtx v1.3j
  - General: Make various commands robust 71
- 2019-08-27 lttab.dtx v1.1q
  - General: Make various commands robust 389
    - Remove several unnecessary `\gdef` definitions 389
- 2019-08-30 lterror.dtx v1.2q
  - `\conditionally@traceoff`: Macro added 68
  - `\conditionally@traceon`: Macro added 68
- 2019-09-09 ltfssdcl.dtx v3.0s
  - `\DeclareMathSymbol`: Allow definition if the math symbol was a command already robust 238
- 2019-09-14 ltfinal.dtx v2.1h
  - `\@uclclist`: Expand UTF8 chrs when case changing (github/177) 623
- 2019-09-16 ltxref.dtx v1.1m
  - General: Correctly revert the `\p@...` change 332
- 2019-09-21 fontdef.dtx v3.0d
  - General: Distangle alias (gh/184) 277, 281
- 2019-10-02 ltfinal.dtx v2.2
  - General: Load ltexpl 625
- 2019-10-02 ltluatex.dtx v1.1k
  - General: linebreak_filter is `exclusive` 602
    - mlist_to_hlist is `exclusive` . . . 602
    - process_rule is `exclusive` 602

2019-10-07 lttab.dtx v1.1q
- `\extracolsep`: This needs to expand . . . 400

2019-10-11 ltfiles.dtx v1.2c
- `\set@curr@file`: Remove one brace group . . . 96

2019-10-11 ltfsstrc.dtx v3.0l
- `\@font@aliasinfo`: Added 'alias' size function . . . 216

2019-10-18 ltclass.dtx v1.3d
- `\@fileswithoptions`: Initialize `\...-h@@k` only when loading the package or class (gh/198) 565

2019-10-22 ltluatex.dtx v1.1j
- General: page_objnum_provider and process_pdf_image_content classified data . . . 603

2019-10-25 ltmiscen.dtx v1.1q
- `\add@percent@to@temptokena`: Allow unbalanced conditionals in `#1` (gh/202) . . . 336

2019-10-26 ltfiles.dtx v1.2d
- `\@iffileonpath`: quote on openin 97
- `\IfFileExists`: dont quote name 97
- `\IfFileExists@`: quote on openin 97
- `\set@curr@file`: remove quotes . 96

2019-11-01 ltdirchk.dtx v1.3a
- `\filename@parse`: take last . not first . . . 12

2019-11-02 ltmiscen.dtx v1.1s
- `\@centercr`: Make `\@centercr` robust (gh/203) . . . 341

2019-11-02 ltspace.dtx v1.3k
- `\@normalcr`: Make also `\@normalcr` robust . . . 75

2019-11-09 ltfiles.dtx v1.2e
- `\set@curr@file`: expand and `\string` before removing quotes . . . 96

2019-11-10 ltmiscen.dtx v1.1r
- `\add@percent@to@temptokena`: fix to special comment catcodes (gh/202) . . . 336

2019-11-11 ltfiles.dtx v1.2f
- `\@iffileonpath`: make `\@filef@und` match quoting used on `\openin` . . . 97

2019-11-22 ltoutenc.dtx v2.0l
- General: Avoid spurious if fontenc selects LY1 as default encoding (gh/199) . . . 144

2019-11-29 ltclass.dtx v1.3e
- `\@pr@videpackage`: Protect package info text (gh/52) . . 558

2019-12-17 fontdef.dtx v3.0e
- `\mddefault`: Set `\bfdefault` to “b” . . . 269
- `\updefault`: Set `\updefault` to “up” . . . 269

2019-12-17 ltfntcmd.dtx v3.4c
- `textssc`: Macro added . . . 291

2019-12-17 ltfssbas.dtx v3.2e
- `\usefont`: Don't call `\fontseries` or `\fontshape` . . . 162

2019-12-17 ltfssini.dtx v3.1e
- General: Provide custom series settings a la mweights . . . 248
- `\DeclareEmphSequence`: Provide `\emph` sequences . . . 257

2019-12-18 ltoutenc.dtx v2.0m
- General: Don't fake `\textcompwordmark`; take default from T1 instead . . . 116
- `\add@accent`: Avoid code that breaks `\accent` . . . 109

2019-12-21 fontdef.dtx v3.0e
- General: Distangle alias (gh/184) . . . 275–278

2020-01-11 ltfssini.dtx v3.1f
- `\rmfamily`: Streamlined implementation with hook . . 252
- `\ttfamily`: Streamlined implementation with hook . . . 252, 253

2020-01-20 ltfssdcl.dtx v3.0t
- `\set@mathdelimiter`: fix for gh/251 . . . 242

2020-01-20 ltoutenc.dtx v2.0n
- General: fix for gh/251 . . . 116

2020-01-22 lttextcomp.dtx v1.0b
- `\tc@subst`: The overall default is `\textcompsubstdefault` not `\substdefault` . . . 300

2020-01-25 fontdef.dtx v3.0f
- General: Load t1enc.def last (gh/255) . . . 266

2020-01-25 ltoutenc.dtx v2.0m
- General: Load each encoding file only once (gh/255) . . . 143

2020-01-27 ltclass.dtx v1.3g
- `\endfilecontents`: Fix typo in error message . . . 570

2020-01-28 ltclass.dtx v1.3h
- `\endfilecontents`: Allow spaces in option string and display only unknown options not the whole option list (gh/256) . . 569

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.

File Key: a=ltdirchk.dtx, b=ltplain.dtx, c=ltvers.dtx, d=ltdefns.dtx, e=ltalloc.dtx, f=ltcntrl.dtx, g=lterror.dtx, h=ltpar.dtx, i=ltspace.dtx, j=ltlogos.dtx, k=ltfiles.dtx, l=ltoutenc.dtx, m=ltcounts.dtx, n=ltlength.dtx, o=ltfssbas.dtx, p=ltfssaxes.dtx, q=ltfsstrc.dtx, r=ltfsscmp.dtx, s=ltfssdcl.dtx, t=ltfssini.dtx, u=fontdef.dtx, v=preload.dtx, w=ltfntcmd.dtx, x=lttextcomp.dtx, y=ltpageno.dtx, z=ltxref.dtx, A=ltmiscen.dtx, B=ltmath.dtx, C=ltlists.dtx, D=ltboxes.dtx, E=lttab.dtx, F=ltpictur.dtx, G=ltthm.dtx, H=ltsect.dtx, I=ltfloat.dtx, J=ltidxglo.dtx, K=ltbibl.dtx, L=ltpage.dtx, M=ltoutput.dtx, N=ltclass.dtx, O=lthyphen.dtx, P=ltluatex.dtx, Q=ltfinal.dtx

File Key: a=ltdirchk.dtx, b=ltplain.dtx, c=ltvers.dtx, d=ltdefns.dtx, e=ltalloc.dtx, f=ltcntrl.dtx, g=lterror.dtx, h=ltpar.dtx, i=ltspace.dtx, j=ltlogos.dtx, k=ltfiles.dtx, l=ltoutenc.dtx, m=ltcounts.dtx, n=ltlength.dtx, o=ltfssbas.dtx, p=ltfssaxes.dtx, q=ltfsstrc.dtx, r=ltfsscmp.dtx, s=ltfssdcl.dtx, t=ltfssini.dtx, u=fontdef.dtx, v=preload.dtx, w=ltfntcmd.dtx, x=lttextcomp.dtx, y=ltpageno.dtx, z=ltxref.dtx, A=ltmiscen.dtx, B=ltmath.dtx, C=ltlists.dtx, D=ltboxes.dtx, E=lttab.dtx, F=ltpictur.dtx, G=ltthm.dtx, H=ltsect.dtx, I=ltfloat.dtx, J=ltidxglo.dtx, K=ltbibl.dtx, L=ltpage.dtx, M=ltoutput.dtx, N=ltclass.dtx, O=lthyphen.dtx, P=ltluatex.dtx, Q=ltfinal.dtx

File Key: a=ltdirchk.dtx, b=ltplain.dtx, c=ltvers.dtx, d=ltdefns.dtx, e=ltalloc.dtx, f=ltcntrl.dtx, g=lterror.dtx, h=ltpar.dtx, i=ltspace.dtx, j=ltlogos.dtx, k=ltfiles.dtx, l=ltoutenc.dtx, m=ltcounts.dtx, n=ltlength.dtx, o=ltfssbas.dtx, p=ltfssaxes.dtx, q=ltfsstrc.dtx, r=ltfsscmp.dtx, s=ltfssdcl.dtx, t=ltfssini.dtx, u=fontdef.dtx, v=preload.dtx, w=ltfntcmd.dtx, x=lttextcomp.dtx, y=ltpageno.dtx, z=ltxref.dtx, A=ltmiscen.dtx, B=ltmath.dtx, C=ltlists.dtx, D=ltboxes.dtx, E=lttab.dtx, F=ltpictur.dtx, G=ltthm.dtx, H=ltsect.dtx, I=ltfloat.dtx, J=ltidxglo.dtx, K=ltbibl.dtx, L=ltpage.dtx, M=ltoutput.dtx, N=ltclass.dtx, O=lthyphen.dtx, P=ltluatex.dtx, Q=ltfinal.dtx

File Key: a=ltdirchk.dtx, b=ltplain.dtx, c=ltvers.dtx, d=ltdefns.dtx, e=ltalloc.dtx, f=ltcntrl.dtx, g=lterror.dtx, h=ltpar.dtx, i=ltspace.dtx, j=ltlogos.dtx, k=ltfiles.dtx, l=ltoutenc.dtx, m=ltcounts.dtx, n=ltlength.dtx, o=ltfssbas.dtx, p=ltfssaxes.dtx, q=ltfsstrc.dtx, r=ltfsscmp.dtx, s=ltfssdcl.dtx, t=ltfssini.dtx, u=fontdef.dtx, v=preload.dtx, w=ltfntcmd.dtx, x=lttextcomp.dtx, y=ltpageno.dtx, z=ltxref.dtx, A=ltmiscen.dtx, B=ltmath.dtx, C=ltlists.dtx, D=ltboxes.dtx, E=lttab.dtx, F=ltpictur.dtx, G=ltthm.dtx, H=ltsect.dtx, I=ltfloat.dtx, J=ltidxglo.dtx, K=ltbibl.dtx, L=ltpage.dtx, M=ltoutput.dtx, N=ltclass.dtx, O=lthyphen.dtx, P=ltluatex.dtx, Q=ltfinal.dtx

File Key: a=ltdirchk.dtx, b=ltplain.dtx, c=ltvers.dtx, d=ltdefns.dtx, e=ltalloc.dtx, f=ltcntrl.dtx, g=lterror.dtx, h=ltpar.dtx, i=ltspace.dtx, j=ltlogos.dtx, k=ltfiles.dtx, l=ltoutenc.dtx, m=ltcounts.dtx, n=ltlength.dtx, o=ltfssbas.dtx, p=ltfssaxes.dtx, q=ltfsstrc.dtx, r=ltfsscmp.dtx, s=ltfssdcl.dtx, t=ltfssini.dtx, u=fontdef.dtx, v=preload.dtx, w=ltfntcmd.dtx, x=lttextcomp.dtx, y=ltpageno.dtx, z=ltxref.dtx, A=ltmiscen.dtx, B=ltmath.dtx, C=ltlists.dtx, D=ltboxes.dtx, E=lttab.dtx, F=ltpictur.dtx, G=ltthm.dtx, H=ltsect.dtx, I=ltfloat.dtx, J=ltidxglo.dtx, K=ltbibl.dtx, L=ltpage.dtx, M=ltoutput.dtx, N=ltclass.dtx, O=lthyphen.dtx, P=ltluatex.dtx, Q=ltfinal.dtx

File Key: a=ltdirchk.dtx, b=ltplain.dtx, c=ltvers.dtx, d=ltdefns.dtx, e=ltalloc.dtx, f=ltcntrl.dtx, g=lterror.dtx, h=ltpar.dtx, i=ltspace.dtx, j=ltlogos.dtx, k=ltfiles.dtx, l=ltoutenc.dtx, m=ltcounts.dtx, n=ltlength.dtx, o=ltfssbas.dtx, p=ltfssaxes.dtx, q=ltfsstrc.dtx, r=ltfsscmp.dtx, s=ltfssdcl.dtx, t=ltfssini.dtx, u=fontdef.dtx, v=preload.dtx, w=ltfntcmd.dtx, x=lttextcomp.dtx, y=ltpageno.dtx, z=ltxref.dtx, A=ltmiscen.dtx, B=ltmath.dtx, C=ltlists.dtx, D=ltboxes.dtx, E=lttab.dtx, F=ltpictur.dtx, G=ltthm.dtx, H=ltsect.dtx, I=ltfloat.dtx, J=ltidxglo.dtx, K=ltbibl.dtx, L=ltpage.dtx, M=ltoutput.dtx, N=ltclass.dtx, O=lthyphen.dtx, P=ltluatex.dtx, Q=ltfinal.dtx

File Key: a=ltdirchk.dtx, b=ltplain.dtx, c=ltvers.dtx, d=ltdefns.dtx, e=ltalloc.dtx, f=ltcntrl.dtx, g=lterror.dtx, h=ltpar.dtx, i=ltspace.dtx, j=ltlogos.dtx, k=ltfiles.dtx, l=ltoutenc.dtx, m=ltcounts.dtx, n=ltlength.dtx, o=ltfssbas.dtx, p=ltfssaxes.dtx, q=ltfsstrc.dtx, r=ltfsscmp.dtx, s=ltfssdcl.dtx, t=ltfssini.dtx, u=fontdef.dtx, v=preload.dtx, w=ltfntcmd.dtx, x=lttextcomp.dtx, y=ltpageno.dtx, z=ltxref.dtx, A=ltmiscen.dtx, B=ltmath.dtx, C=ltlists.dtx, D=ltboxes.dtx, E=lttab.dtx, F=ltpictur.dtx, G=ltthm.dtx, H=ltsect.dtx, I=ltfloat.dtx, J=ltidxglo.dtx, K=ltbibl.dtx, L=ltpage.dtx, M=ltoutput.dtx, N=ltclass.dtx, O=lthyphen.dtx, P=ltluatex.dtx, Q=ltfinal.dtx

File Key: a=ltdirchk.dtx, b=ltplain.dtx, c=ltvers.dtx, d=ltdefns.dtx, e=ltalloc.dtx, f=ltcntrl.dtx, g=lterror.dtx, h=ltpar.dtx, i=ltspace.dtx, j=ltlogos.dtx, k=ltfiles.dtx, l=ltoutenc.dtx, m=ltcounts.dtx, n=ltlength.dtx, o=ltfssbas.dtx, p=ltfssaxes.dtx, q=ltfsstrc.dtx, r=ltfsscmp.dtx, s=ltfssdcl.dtx, t=ltfssini.dtx, u=fontdef.dtx, v=preload.dtx, w=ltfntcmd.dtx, x=lttextcomp.dtx, y=ltpageno.dtx, z=ltxref.dtx, A=ltmiscen.dtx, B=ltmath.dtx, C=ltlists.dtx, D=ltboxes.dtx, E=lttab.dtx, F=ltpictur.dtx, G=ltthm.dtx, H=ltsect.dtx, I=ltfloat.dtx, J=ltidxglo.dtx, K=ltbibl.dtx, L=ltpage.dtx, M=ltoutput.dtx, N=ltclass.dtx, O=lthyphen.dtx, P=ltluatex.dtx, Q=ltfinal.dtx

File Key: a=ltdirchk.dtx, b=ltplain.dtx, c=ltvers.dtx, d=ltdefns.dtx, e=ltalloc.dtx, f=ltcntrl.dtx, g=lterror.dtx, h=ltpar.dtx, i=ltspace.dtx, j=ltlogos.dtx, k=ltfiles.dtx, l=ltoutenc.dtx, m=ltcounts.dtx, n=ltlength.dtx, o=ltfssbas.dtx, p=ltfssaxes.dtx, q=ltfsstrc.dtx, r=ltfsscmp.dtx, s=ltfssdcl.dtx, t=ltfssini.dtx, u=fontdef.dtx, v=preload.dtx, w=ltfntcmd.dtx, x=lttextcomp.dtx, y=ltpageno.dtx, z=ltxref.dtx, A=ltmiscen.dtx, B=ltmath.dtx, C=ltlists.dtx, D=ltboxes.dtx, E=lttab.dtx, F=ltpictur.dtx, G=ltthm.dtx, H=ltsect.dtx, I=ltfloat.dtx, J=ltidxglo.dtx, K=ltbibl.dtx, L=ltpage.dtx, M=ltoutput.dtx, N=ltclass.dtx, O=lthyphen.dtx, P=ltluatex.dtx, Q=ltfinal.dtx

File Key: a=ltdirchk.dtx, b=ltplain.dtx, c=ltvers.dtx, d=ltdefns.dtx, e=ltalloc.dtx, f=ltcntrl.dtx, g=lterror.dtx, h=ltpar.dtx, i=ltspace.dtx, j=ltlogos.dtx, k=ltfiles.dtx, l=ltoutenc.dtx, m=ltcounts.dtx, n=ltlength.dtx, o=ltfssbas.dtx, p=ltfssaxes.dtx, q=ltfsstrc.dtx, r=ltfsscmp.dtx, s=ltfssdcl.dtx, t=ltfssini.dtx, u=fontdef.dtx, v=preload.dtx, w=ltfntcmd.dtx, x=lttextcomp.dtx, y=ltpageno.dtx, z=ltxref.dtx, A=ltmiscen.dtx, B=ltmath.dtx, C=ltlists.dtx, D=ltboxes.dtx, E=lttab.dtx, F=ltpictur.dtx, G=ltthm.dtx, H=ltsect.dtx, I=ltfloat.dtx, J=ltidxglo.dtx, K=ltbibl.dtx, L=ltpage.dtx, M=ltoutput.dtx, N=ltclass.dtx, O=lthyphen.dtx, P=ltluatex.dtx, Q=ltfinal.dtx

File Key: a=ltdirchk.dtx, b=ltplain.dtx, c=ltvers.dtx, d=ltdefns.dtx, e=ltalloc.dtx, f=ltcntrl.dtx, g=lterror.dtx, h=ltpar.dtx, i=ltspace.dtx, j=ltlogos.dtx, k=ltfiles.dtx, l=ltoutenc.dtx, m=ltcounts.dtx, n=ltlength.dtx, o=ltfssbas.dtx, p=ltfssaxes.dtx, q=ltfsstrc.dtx, r=ltfsscmp.dtx, s=ltfssdcl.dtx, t=ltfssini.dtx, u=fontdef.dtx, v=preload.dtx, w=ltfntcmd.dtx, x=lttextcomp.dtx, y=ltpageno.dtx, z=ltxref.dtx, A=ltmiscen.dtx, B=ltmath.dtx, C=ltlists.dtx, D=ltboxes.dtx, E=lttab.dtx, F=ltpictur.dtx, G=ltthm.dtx, H=ltsect.dtx, I=ltfloat.dtx, J=ltidxglo.dtx, K=ltbibl.dtx, L=ltpage.dtx, M=ltoutput.dtx, N=ltclass.dtx, O=lthyphen.dtx, P=ltluatex.dtx, Q=ltfinal.dtx

File Key: a=ltdirchk.dtx, b=ltplain.dtx, c=ltvers.dtx, d=ltdefns.dtx, e=ltalloc.dtx, f=ltcntrl.dtx, g=lterror.dtx, h=ltpar.dtx, i=ltspace.dtx, j=ltlogos.dtx, k=ltfiles.dtx, l=ltoutenc.dtx, m=ltcounts.dtx, n=ltlength.dtx, o=ltfssbas.dtx, p=ltfssaxes.dtx, q=ltfsstrc.dtx, r=ltfsscmp.dtx, s=ltfssdcl.dtx, t=ltfssini.dtx, u=fontdef.dtx, v=preload.dtx, w=ltfntcmd.dtx, x=lttextcomp.dtx, y=ltpageno.dtx, z=ltxref.dtx, A=ltmiscen.dtx, B=ltmath.dtx, C=ltlists.dtx, D=ltboxes.dtx, E=lttab.dtx, F=ltpictur.dtx, G=ltthm.dtx, H=ltsect.dtx, I=ltfloat.dtx, J=ltidxglo.dtx, K=ltbibl.dtx, L=ltpage.dtx, M=ltoutput.dtx, N=ltclass.dtx, O=lthyphen.dtx, P=ltluatex.dtx, Q=ltfinal.dtx

File Key: a=ltdirchk.dtx, b=ltplain.dtx, c=ltvers.dtx, d=ltdefns.dtx, e=ltalloc.dtx, f=ltcntrl.dtx, g=lterror.dtx, h=ltpar.dtx, i=ltspace.dtx, j=ltlogos.dtx, k=ltfiles.dtx, l=ltoutenc.dtx, m=ltcounts.dtx, n=ltlength.dtx, o=ltfssbas.dtx, p=ltfssaxes.dtx, q=ltfsstrc.dtx, r=ltfsscmp.dtx, s=ltfssdcl.dtx, t=ltfssini.dtx, u=fontdef.dtx, v=preload.dtx, w=ltfntcmd.dtx, x=lttextcomp.dtx, y=ltpageno.dtx, z=ltxref.dtx, A=ltmiscen.dtx, B=ltmath.dtx, C=ltlists.dtx, D=ltboxes.dtx, E=lttab.dtx, F=ltpictur.dtx, G=ltthm.dtx, H=ltsect.dtx, I=ltfloat.dtx, J=ltidxglo.dtx, K=ltbibl.dtx, L=ltpage.dtx, M=ltoutput.dtx, N=ltclass.dtx, O=lthyphen.dtx, P=ltluatex.dtx, Q=ltfinal.dtx

File Key: a=`ltdirchk.dtx`, b=`ltplain.dtx`, c=`ltvers.dtx`, d=`ltdefns.dtx`, e=`ltalloc.dtx`, f=`ltcntrl.dtx`, g=`lterror.dtx`, h=`ltpar.dtx`, i=`ltspace.dtx`, j=`ltlogos.dtx`, k=`ltfiles.dtx`, l=`ltoutenc.dtx`, m=`ltcounts.dtx`, n=`ltlength.dtx`, o=`ltfssbas.dtx`, p=`ltfssaxes.dtx`, q=`ltfsstrc.dtx`, r=`ltfsscmp.dtx`, s=`ltfssdcl.dtx`, t=`ltfssini.dtx`, u=`fontdef.dtx`, v=`preload.dtx`, w=`ltfntcmd.dtx`, x=`lttextcomp.dtx`, y=`ltpageno.dtx`, z=`ltxref.dtx`, A=`ltmiscen.dtx`, B=`ltmath.dtx`, C=`ltlists.dtx`, D=`ltboxes.dtx`, E=`lttab.dtx`, F=`ltpictur.dtx`, G=`ltthm.dtx`, H=`ltsect.dtx`, I=`ltfloat.dtx`, J=`ltidxglo.dtx`, K=`ltbibl.dtx`, L=`ltpage.dtx`, M=`ltoutput.dtx`, N=`ltclass.dtx`, O=`lthyphen.dtx`, P=`ltluatex.dtx`, Q=`ltfinal.dtx`

File Key: a=ltdirchk.dtx, b=ltplain.dtx, c=ltvers.dtx, d=ltdefns.dtx, e=ltalloc.dtx, f=ltcntrl.dtx, g=lterror.dtx, h=ltpar.dtx, i=ltspace.dtx, j=ltlogos.dtx, k=ltfiles.dtx, l=ltoutenc.dtx, m=ltcounts.dtx, n=ltlength.dtx, o=ltfssbas.dtx, p=ltfssaxes.dtx, q=ltfsstrc.dtx, r=ltfsscmp.dtx, s=ltfssdcl.dtx, t=ltfssini.dtx, u=fontdef.dtx, v=preload.dtx, w=ltfntcmd.dtx, x=lttextcomp.dtx, y=ltpageno.dtx, z=ltxref.dtx, A=ltmiscen.dtx, B=ltmath.dtx, C=ltlists.dtx, D=ltboxes.dtx, E=lttab.dtx, F=ltpictur.dtx, G=ltthm.dtx, H=ltsect.dtx, I=ltfloat.dtx, J=ltidxglo.dtx, K=ltbibl.dtx, L=ltpage.dtx, M=ltoutput.dtx, N=ltclass.dtx, O=lthyphen.dtx, P=ltluatex.dtx, Q=ltfinal.dtx

File Key: a=ltdirchk.dtx, b=ltplain.dtx, c=ltvers.dtx, d=ltdefns.dtx, e=ltalloc.dtx, f=ltcntrl.dtx, g=lterror.dtx, h=ltpar.dtx, i=ltspace.dtx, j=ltlogos.dtx, k=ltfiles.dtx, l=ltoutenc.dtx, m=ltcounts.dtx, n=ltlength.dtx, o=ltfssbas.dtx, p=ltfssaxes.dtx, q=ltfsstrc.dtx, r=ltfsscmp.dtx, s=ltfssdcl.dtx, t=ltfssini.dtx, u=fontdef.dtx, v=preload.dtx, w=ltfntcmd.dtx, x=lttextcomp.dtx, y=ltpageno.dtx, z=ltxref.dtx, A=ltmiscen.dtx, B=ltmath.dtx, C=ltlists.dtx, D=ltboxes.dtx, E=lttab.dtx, F=ltpictur.dtx, G=ltthm.dtx, H=ltsect.dtx, I=ltfloat.dtx, J=ltidxglo.dtx, K=ltbibl.dtx, L=ltpage.dtx, M=ltoutput.dtx, N=ltclass.dtx, O=lthyphen.dtx, P=ltluatex.dtx, Q=ltfinal.dtx

File Key: a=ltdirchk.dtx, b=ltplain.dtx, c=ltvers.dtx, d=ltdefns.dtx, e=ltalloc.dtx, f=ltcntrl.dtx, g=lterror.dtx, h=ltpar.dtx, i=ltspace.dtx, j=ltlogos.dtx, k=ltfiles.dtx, l=ltoutenc.dtx, m=ltcounts.dtx, n=ltlength.dtx, o=ltfssbas.dtx, p=ltfssaxes.dtx, q=ltfsstrc.dtx, r=ltfsscmp.dtx, s=ltfssdcl.dtx, t=ltfssini.dtx, u=fontdef.dtx, v=preload.dtx, w=ltfntcmd.dtx, x=lttextcomp.dtx, y=ltpageno.dtx, z=ltxref.dtx, A=ltmiscen.dtx, B=ltmath.dtx, C=ltlists.dtx, D=ltboxes.dtx, E=lttab.dtx, F=ltpictur.dtx, G=ltthm.dtx, H=ltsect.dtx, I=ltfloat.dtx, J=ltidxglo.dtx, K=ltbibl.dtx, L=ltpage.dtx, M=ltoutput.dtx, N=ltclass.dtx, O=lthyphen.dtx, P=ltluatex.dtx, Q=ltfinal.dtx

File Key: a=ltdirchk.dtx, b=ltplain.dtx, c=ltvers.dtx, d=ltdefns.dtx, e=ltalloc.dtx, f=ltcntrl.dtx, g=lterror.dtx, h=ltpar.dtx, i=ltspace.dtx, j=ltlogos.dtx, k=ltfiles.dtx, l=ltoutenc.dtx, m=ltcounts.dtx, n=ltlength.dtx, o=ltfssbas.dtx, p=ltfssaxes.dtx, q=ltfsstrc.dtx, r=ltfsscmp.dtx, s=ltfssdcl.dtx, t=ltfssini.dtx, u=fontdef.dtx, v=preload.dtx, w=ltfntcmd.dtx, x=lttextcomp.dtx, y=ltpageno.dtx, z=ltxref.dtx, A=ltmiscen.dtx, B=ltmath.dtx, C=ltlists.dtx, D=ltboxes.dtx, E=lttab.dtx, F=ltpictur.dtx, G=ltthm.dtx, H=ltsect.dtx, I=ltfloat.dtx, J=ltidxglo.dtx, K=ltbibl.dtx, L=ltpage.dtx, M=ltoutput.dtx, N=ltclass.dtx, O=lthyphen.dtx, P=ltluatex.dtx, Q=ltfinal.dtx

File Key: a=ltdirchk.dtx, b=ltplain.dtx, c=ltvers.dtx, d=ltdefns.dtx, e=ltalloc.dtx, f=ltcntrl.dtx, g=lterror.dtx, h=ltpar.dtx, i=ltspace.dtx, j=ltlogos.dtx, k=ltfiles.dtx, l=ltoutenc.dtx, m=ltcounts.dtx, n=ltlength.dtx, o=ltfssbas.dtx, p=ltfssaxes.dtx, q=ltfsstrc.dtx, r=ltfsscmp.dtx, s=ltfssdcl.dtx, t=ltfssini.dtx, u=fontdef.dtx, v=preload.dtx, w=ltfntcmd.dtx, x=lttextcomp.dtx, y=ltpageno.dtx, z=ltxref.dtx, A=ltmiscen.dtx, B=ltmath.dtx, C=ltlists.dtx, D=ltboxes.dtx, E=lttab.dtx, F=ltpictur.dtx, G=ltthm.dtx, H=ltsect.dtx, I=ltfloat.dtx, J=ltidxglo.dtx, K=ltbibl.dtx, L=ltpage.dtx, M=ltoutput.dtx, N=ltclass.dtx, O=lthyphen.dtx, P=ltluatex.dtx, Q=ltfinal.dtx

File Key: a=ltdirchk.dtx, b=ltplain.dtx, c=ltvers.dtx, d=ltdefns.dtx, e=ltalloc.dtx, f=ltcntrl.dtx, g=lterror.dtx, h=ltpar.dtx, i=ltspace.dtx, j=ltlogos.dtx, k=ltfiles.dtx, l=ltoutenc.dtx, m=ltcounts.dtx, n=ltlength.dtx, o=ltfssbas.dtx, p=ltfssaxes.dtx, q=ltfsstrc.dtx, r=ltfsscmp.dtx, s=ltfssdcl.dtx, t=ltfssini.dtx, u=fontdef.dtx, v=preload.dtx, w=ltfntcmd.dtx, x=lttextcomp.dtx, y=ltpageno.dtx, z=ltxref.dtx, A=ltmiscen.dtx, B=ltmath.dtx, C=ltlists.dtx, D=ltboxes.dtx, E=lttab.dtx, F=ltpictur.dtx, G=ltthm.dtx, H=ltsect.dtx, I=ltfloat.dtx, J=ltidxglo.dtx, K=ltbibl.dtx, L=ltpage.dtx, M=ltoutput.dtx, N=ltclass.dtx, O=lthyphen.dtx, P=ltluatex.dtx, Q=ltfinal.dtx

File Key: a=ltdirchk.dtx, b=ltplain.dtx, c=ltvers.dtx, d=ltdefns.dtx, e=ltalloc.dtx, f=ltcntrl.dtx, g=lterror.dtx, h=ltpar.dtx, i=ltspace.dtx, j=ltlogos.dtx, k=ltfiles.dtx, l=ltoutenc.dtx, m=ltcounts.dtx, n=ltlength.dtx, o=ltfssbas.dtx, p=ltfssaxes.dtx, q=ltfsstrc.dtx, r=ltfsscmp.dtx, s=ltfssdcl.dtx, t=ltfssini.dtx, u=fontdef.dtx, v=preload.dtx, w=ltfntcmd.dtx, x=lttextcomp.dtx, y=ltpageno.dtx, z=ltxref.dtx, A=ltmiscen.dtx, B=ltmath.dtx, C=ltlists.dtx, D=ltboxes.dtx, E=lttab.dtx, F=ltpictur.dtx, G=ltthm.dtx, H=ltsect.dtx, I=ltfloat.dtx, J=ltidxglo.dtx, K=ltbibl.dtx, L=ltpage.dtx, M=ltoutput.dtx, N=ltclass.dtx, O=lthyphen.dtx, P=ltluatex.dtx, Q=ltfinal.dtx

File Key: a=ltdirchk.dtx, b=ltplain.dtx, c=ltvers.dtx, d=ltdefns.dtx, e=ltalloc.dtx, f=ltcntrl.dtx, g=lterror.dtx, h=ltpar.dtx, i=ltspace.dtx, j=ltlogos.dtx, k=ltfiles.dtx, l=ltoutenc.dtx, m=ltcounts.dtx, n=ltlength.dtx, o=ltfssbas.dtx, p=ltfssaxes.dtx, q=ltfsstrc.dtx, r=ltfsscmp.dtx, s=ltfssdcl.dtx, t=ltfssini.dtx, u=fontdef.dtx, v=preload.dtx, w=ltfntcmd.dtx, x=lttextcomp.dtx, y=ltpageno.dtx, z=ltxref.dtx, A=ltmiscen.dtx, B=ltmath.dtx, C=ltlists.dtx, D=ltboxes.dtx, E=lttab.dtx, F=ltpictur.dtx, G=ltthm.dtx, H=ltsect.dtx, I=ltfloat.dtx, J=ltidxglo.dtx, K=ltbibl.dtx, L=ltpage.dtx, M=ltoutput.dtx, N=ltclass.dtx, O=lthyphen.dtx, P=ltluatex.dtx, Q=ltfinal.dtx

File Key: a=ltdirchk.dtx, b=ltplain.dtx, c=ltvers.dtx, d=ltdefns.dtx, e=ltalloc.dtx, f=ltcntrl.dtx, g=lterror.dtx, h=ltpar.dtx, i=ltspace.dtx, j=ltlogos.dtx, k=ltfiles.dtx, l=ltoutenc.dtx, m=ltcounts.dtx, n=ltlength.dtx, o=ltfssbas.dtx, p=ltfssaxes.dtx, q=ltfsstrc.dtx, r=ltfsscmp.dtx, s=ltfssdcl.dtx, t=ltfssini.dtx, u=fontdef.dtx, v=preload.dtx, w=ltfntcmd.dtx, x=lttextcomp.dtx, y=ltpageno.dtx, z=ltxref.dtx, A=ltmiscen.dtx, B=ltmath.dtx, C=ltlists.dtx, D=ltboxes.dtx, E=lttab.dtx, F=ltpictur.dtx, G=ltthm.dtx, H=ltsect.dtx, I=ltfloat.dtx, J=ltidxglo.dtx, K=ltbibl.dtx, L=ltpage.dtx, M=ltoutput.dtx, N=ltclass.dtx, O=lthyphen.dtx, P=ltluatex.dtx, Q=ltfinal.dtx

File Key: a=ltdirchk.dtx, b=ltplain.dtx, c=ltvers.dtx, d=ltdefns.dtx, e=ltalloc.dtx, f=ltcntrl.dtx, g=lterror.dtx, h=ltpar.dtx, i=ltspace.dtx, j=ltlogos.dtx, k=ltfiles.dtx, l=ltoutenc.dtx, m=ltcounts.dtx, n=ltlength.dtx, o=ltfssbas.dtx, p=ltfssaxes.dtx, q=ltfsstrc.dtx, r=ltfsscmp.dtx, s=ltfssdcl.dtx, t=ltfssini.dtx, u=fontdef.dtx, v=preload.dtx, w=ltfntcmd.dtx, x=lttextcomp.dtx, y=ltpageno.dtx, z=ltxref.dtx, A=ltmiscen.dtx, B=ltmath.dtx, C=ltlists.dtx, D=ltboxes.dtx, E=lttab.dtx, F=ltpictur.dtx, G=ltthm.dtx, H=ltsect.dtx, I=ltfloat.dtx, J=ltidxglo.dtx, K=ltbibl.dtx, L=ltpage.dtx, M=ltoutput.dtx, N=ltclass.dtx, O=lthyphen.dtx, P=ltluatex.dtx, Q=ltfinal.dtx

File Key: a=ltdirchk.dtx, b=ltplain.dtx, c=ltvers.dtx, d=ltdefns.dtx, e=ltalloc.dtx, f=ltcntrl.dtx, g=lterror.dtx, h=ltpar.dtx, i=ltspace.dtx, j=ltlogos.dtx, k=ltfiles.dtx, l=ltoutenc.dtx, m=ltcounts.dtx, n=ltlength.dtx, o=ltfssbas.dtx, p=ltfssaxes.dtx, q=ltfsstrc.dtx, r=ltfsscmp.dtx, s=ltfssdcl.dtx, t=ltfssini.dtx, u=fontdef.dtx, v=preload.dtx, w=ltfntcmd.dtx, x=lttextcomp.dtx, y=ltpageno.dtx, z=ltxref.dtx, A=ltmiscen.dtx, B=ltmath.dtx, C=ltlists.dtx, D=ltboxes.dtx, E=lttab.dtx, F=ltpictur.dtx, G=ltthm.dtx, H=ltsect.dtx, I=ltfloat.dtx, J=ltidxglo.dtx, K=ltbibl.dtx, L=ltpage.dtx, M=ltoutput.dtx, N=ltclass.dtx, O=lthyphen.dtx, P=ltluatex.dtx, Q=ltfinal.dtx

File Key: a=ltdirchk.dtx, b=ltplain.dtx, c=ltvers.dtx, d=ltdefns.dtx, e=ltalloc.dtx, f=ltcntrl.dtx, g=lterror.dtx, h=ltpar.dtx, i=ltspace.dtx, j=ltlogos.dtx, k=ltfiles.dtx, l=ltoutenc.dtx, m=ltcounts.dtx, n=ltlength.dtx, o=ltfssbas.dtx, p=ltfssaxes.dtx, q=ltfsstrc.dtx, r=ltfsscmp.dtx, s=ltfssdcl.dtx, t=ltfssini.dtx, u=fontdef.dtx, v=preload.dtx, w=ltfntcmd.dtx, x=lttextcomp.dtx, y=ltpageno.dtx, z=ltxref.dtx, A=ltmiscen.dtx, B=ltmath.dtx, C=ltlists.dtx, D=ltboxes.dtx, E=lttab.dtx, F=ltpictur.dtx, G=ltthm.dtx, H=ltsect.dtx, I=ltfloat.dtx, J=ltidxglo.dtx, K=ltbibl.dtx, L=ltpage.dtx, M=ltoutput.dtx, N=ltclass.dtx, O=lthyphen.dtx, P=ltluatex.dtx, Q=ltfinal.dtx

File Key: a=ltdirchk.dtx, b=ltplain.dtx, c=ltvers.dtx, d=ltdefns.dtx, e=ltalloc.dtx, f=ltcntrl.dtx, g=lterror.dtx, h=ltpar.dtx, i=ltspace.dtx, j=ltlogos.dtx, k=ltfiles.dtx, l=ltoutenc.dtx, m=ltcounts.dtx, n=ltlength.dtx, o=ltfssbas.dtx, p=ltfssaxes.dtx, q=ltfsstrc.dtx, r=ltfsscmp.dtx, s=ltfssdcl.dtx, t=ltfssini.dtx, u=fontdef.dtx, v=preload.dtx, w=ltfntcmd.dtx, x=lttextcomp.dtx, y=ltpageno.dtx, z=ltxref.dtx, A=ltmiscen.dtx, B=ltmath.dtx, C=ltlists.dtx, D=ltboxes.dtx, E=lttab.dtx, F=ltpictur.dtx, G=ltthm.dtx, H=ltsect.dtx, I=ltfloat.dtx, J=ltidxglo.dtx, K=ltbibl.dtx, L=ltpage.dtx, M=ltoutput.dtx, N=ltclass.dtx, O=lthyphen.dtx, P=ltluatex.dtx, Q=ltfinal.dtx

File Key: a=ltdirchk.dtx, b=ltplain.dtx, c=ltvers.dtx, d=ltdefns.dtx, e=ltalloc.dtx, f=ltcntrl.dtx, g=lterror.dtx, h=ltpar.dtx, i=ltspace.dtx, j=ltlogos.dtx, k=ltfiles.dtx, l=ltoutenc.dtx, m=ltcounts.dtx, n=ltlength.dtx, o=ltfssbas.dtx, p=ltfssaxes.dtx, q=ltfsstrc.dtx, r=ltfsscmp.dtx, s=ltfssdcl.dtx, t=ltfssini.dtx, u=fontdef.dtx, v=preload.dtx, w=ltfntcmd.dtx, x=lttextcomp.dtx, y=ltpageno.dtx, z=ltxref.dtx, A=ltmiscen.dtx, B=ltmath.dtx, C=ltlists.dtx, D=ltboxes.dtx, E=lttab.dtx, F=ltpictur.dtx, G=ltthm.dtx, H=ltsect.dtx, I=ltfloat.dtx, J=ltidxglo.dtx, K=ltbibl.dtx, L=ltpage.dtx, M=ltoutput.dtx, N=ltclass.dtx, O=lthyphen.dtx, P=ltluatex.dtx, Q=ltfinal.dtx

File Key: a=ltdirchk.dtx, b=ltplain.dtx, c=ltvers.dtx, d=ltdefns.dtx, e=ltalloc.dtx, f=ltcntrl.dtx, g=lterror.dtx, h=ltpar.dtx, i=ltspace.dtx, j=ltlogos.dtx, k=ltfiles.dtx, l=ltoutenc.dtx, m=ltcounts.dtx, n=ltlength.dtx, o=ltfssbas.dtx, p=ltfssaxes.dtx, q=ltfsstrc.dtx, r=ltfsscmp.dtx, s=ltfssdcl.dtx, t=ltfssini.dtx, u=fontdef.dtx, v=preload.dtx, w=ltfntcmd.dtx, x=lttextcomp.dtx, y=ltpageno.dtx, z=ltxref.dtx, A=ltmiscen.dtx, B=ltmath.dtx, C=ltlists.dtx, D=ltboxes.dtx, E=lttab.dtx, F=ltpictur.dtx, G=ltthm.dtx, H=ltsect.dtx, I=ltfloat.dtx, J=ltidxglo.dtx, K=ltbibl.dtx, L=ltpage.dtx, M=ltoutput.dtx, N=ltclass.dtx, O=lthyphen.dtx, P=ltluatex.dtx, Q=ltfinal.dtx

File Key: a=ltdirchk.dtx, b=ltplain.dtx, c=ltvers.dtx, d=ltdefns.dtx, e=ltalloc.dtx, f=ltcntrl.dtx, g=lterror.dtx, h=ltpar.dtx, i=ltspace.dtx, j=ltlogos.dtx, k=ltfiles.dtx, l=ltoutenc.dtx, m=ltcounts.dtx, n=ltlength.dtx, o=ltfssbas.dtx, p=ltfssaxes.dtx, q=ltfsstrc.dtx, r=ltfsscmp.dtx, s=ltfssdcl.dtx, t=ltfssini.dtx, u=fontdef.dtx, v=preload.dtx, w=ltfntcmd.dtx, x=lttextcomp.dtx, y=ltpageno.dtx, z=ltxref.dtx, A=ltmiscen.dtx, B=ltmath.dtx, C=ltlists.dtx, D=ltboxes.dtx, E=lttab.dtx, F=ltpictur.dtx, G=ltthm.dtx, H=ltsect.dtx, I=ltfloat.dtx, J=ltidxglo.dtx, K=ltbibl.dtx, L=ltpage.dtx, M=ltoutput.dtx, N=ltclass.dtx, O=lthyphen.dtx, P=ltluatex.dtx, Q=ltfinal.dtx

File Key: a=ltdirchk.dtx, b=ltplain.dtx, c=ltvers.dtx, d=ltdefns.dtx, e=ltalloc.dtx, f=ltcntrl.dtx, g=lterror.dtx, h=ltpar.dtx, i=ltspace.dtx, j=ltlogos.dtx, k=ltfiles.dtx, l=ltoutenc.dtx, m=ltcounts.dtx, n=ltlength.dtx, o=ltfssbas.dtx, p=ltfssaxes.dtx, q=ltfsstrc.dtx, r=ltfsscmp.dtx, s=ltfssdcl.dtx, t=ltfssini.dtx, u=fontdef.dtx, v=preload.dtx, w=ltfntcmd.dtx, x=lttextcomp.dtx, y=ltpageno.dtx, z=ltxref.dtx, A=ltmiscen.dtx, B=ltmath.dtx, C=ltlists.dtx, D=ltboxes.dtx, E=lttab.dtx, F=ltpictur.dtx, G=ltthm.dtx, H=ltsect.dtx, I=ltfloat.dtx, J=ltidxglo.dtx, K=ltbibl.dtx, L=ltpage.dtx, M=ltoutput.dtx, N=ltclass.dtx, O=lthyphen.dtx, P=ltluatex.dtx, Q=ltfinal.dtx

File Key: a=ltdirchk.dtx, b=ltplain.dtx, c=ltvers.dtx, d=ltdefns.dtx, e=ltalloc.dtx, f=ltcntrl.dtx, g=lterror.dtx, h=ltpar.dtx, i=ltspace.dtx, j=ltlogos.dtx, k=ltfiles.dtx, l=ltoutenc.dtx, m=ltcounts.dtx, n=ltlength.dtx, o=ltfssbas.dtx, p=ltfssaxes.dtx, q=ltfsstrc.dtx, r=ltfsscmp.dtx, s=ltfssdcl.dtx, t=ltfssini.dtx, u=fontdef.dtx, v=preload.dtx, w=ltfntcmd.dtx, x=lttextcomp.dtx, y=ltpageno.dtx, z=ltxref.dtx, A=ltmiscen.dtx, B=ltmath.dtx, C=ltlists.dtx, D=ltboxes.dtx, E=lttab.dtx, F=ltpictur.dtx, G=ltthm.dtx, H=ltsect.dtx, I=ltfloat.dtx, J=ltidxglo.dtx, K=ltbibl.dtx, L=ltpage.dtx, M=ltoutput.dtx, N=ltclass.dtx, O=lthyphen.dtx, P=ltluatex.dtx, Q=ltfinal.dtx

File Key: a=ltdirchk.dtx, b=ltplain.dtx, c=ltvers.dtx, d=ltdefns.dtx, e=ltalloc.dtx, f=ltcntrl.dtx, g=lterror.dtx, h=ltpar.dtx, i=ltspace.dtx, j=ltlogos.dtx, k=ltfiles.dtx, l=ltoutenc.dtx, m=ltcounts.dtx, n=ltlength.dtx, o=ltfssbas.dtx, p=ltfssaxes.dtx, q=ltfsstrc.dtx, r=ltfsscmp.dtx, s=ltfssdcl.dtx, t=ltfssini.dtx, u=fontdef.dtx, v=preload.dtx, w=ltfntcmd.dtx, x=lttextcomp.dtx, y=ltpageno.dtx, z=ltxref.dtx, A=ltmiscen.dtx, B=ltmath.dtx, C=ltlists.dtx, D=ltboxes.dtx, E=lttab.dtx, F=ltpictur.dtx, G=ltthm.dtx, H=ltsect.dtx, I=ltfloat.dtx, J=ltidxglo.dtx, K=ltbibl.dtx, L=ltpage.dtx, M=ltoutput.dtx, N=ltclass.dtx, O=lthyphen.dtx, P=ltluatex.dtx, Q=ltfinal.dtx

File Key: a=ltdirchk.dtx, b=ltplain.dtx, c=ltvers.dtx, d=ltdefns.dtx, e=ltalloc.dtx, f=ltcntrl.dtx, g=lterror.dtx, h=ltpar.dtx, i=ltspace.dtx, j=ltlogos.dtx, k=ltfiles.dtx, l=ltoutenc.dtx, m=ltcounts.dtx, n=ltlength.dtx, o=ltfssbas.dtx, p=ltfssaxes.dtx, q=ltfsstrc.dtx, r=ltfsscmp.dtx, s=ltfssdcl.dtx, t=ltfssini.dtx, u=fontdef.dtx, v=preload.dtx, w=ltfntcmd.dtx, x=lttextcomp.dtx, y=ltpageno.dtx, z=ltxref.dtx, A=ltmiscen.dtx, B=ltmath.dtx, C=ltlists.dtx, D=ltboxes.dtx, E=lttab.dtx, F=ltpictur.dtx, G=ltthm.dtx, H=ltsect.dtx, I=ltfloat.dtx, J=ltidxglo.dtx, K=ltbibl.dtx, L=ltpage.dtx, M=ltoutput.dtx, N=ltclass.dtx, O=lthyphen.dtx, P=ltluatex.dtx, Q=ltfinal.dtx

File Key: a=ltdirchk.dtx, b=ltplain.dtx, c=ltvers.dtx, d=ltdefns.dtx, e=ltalloc.dtx, f=ltcntrl.dtx, g=lterror.dtx, h=ltpar.dtx, i=ltspace.dtx, j=ltlogos.dtx, k=ltfiles.dtx, l=ltoutenc.dtx, m=ltcounts.dtx, n=ltlength.dtx, o=ltfssbas.dtx, p=ltfssaxes.dtx, q=ltfsstrc.dtx, r=ltfsscmp.dtx, s=ltfssdcl.dtx, t=ltfssini.dtx, u=fontdef.dtx, v=preload.dtx, w=ltfntcmd.dtx, x=lttextcomp.dtx, y=ltpageno.dtx, z=ltxref.dtx, A=ltmiscen.dtx, B=ltmath.dtx, C=ltlists.dtx, D=ltboxes.dtx, E=lttab.dtx, F=ltpictur.dtx, G=ltthm.dtx, H=ltsect.dtx, I=ltfloat.dtx, J=ltidxglo.dtx, K=ltbibl.dtx, L=ltpage.dtx, M=ltoutput.dtx, N=ltclass.dtx, O=lthyphen.dtx, P=ltluatex.dtx, Q=ltfinal.dtx

File Key: a=ltdirchk.dtx, b=ltplain.dtx, c=ltvers.dtx, d=ltdefns.dtx, e=ltalloc.dtx, f=ltcntrl.dtx, g=lterror.dtx, h=ltpar.dtx, i=ltspace.dtx, j=ltlogos.dtx, k=ltfiles.dtx, l=ltoutenc.dtx, m=ltcounts.dtx, n=ltlength.dtx, o=ltfssbas.dtx, p=ltfssaxes.dtx, q=ltfsstrc.dtx, r=ltfsscmp.dtx, s=ltfssdcl.dtx, t=ltfssini.dtx, u=fontdef.dtx, v=preload.dtx, w=ltfntcmd.dtx, x=lttextcomp.dtx, y=ltpageno.dtx, z=ltxref.dtx, A=ltmiscen.dtx, B=ltmath.dtx, C=ltlists.dtx, D=ltboxes.dtx, E=lttab.dtx, F=ltpictur.dtx, G=ltthm.dtx, H=ltsect.dtx, I=ltfloat.dtx, J=ltidxglo.dtx, K=ltbibl.dtx, L=ltpage.dtx, M=ltoutput.dtx, N=ltclass.dtx, O=lthyphen.dtx, P=ltluatex.dtx, Q=ltfinal.dtx

File Key: a=`ltdirchk.dtx`, b=`ltplain.dtx`, c=`ltvers.dtx`, d=`ltdefns.dtx`, e=`ltalloc.dtx`, f=`ltcntrl.dtx`, g=`lterror.dtx`, h=`ltpar.dtx`, i=`ltspace.dtx`, j=`ltlogos.dtx`, k=`ltfiles.dtx`, l=`ltoutenc.dtx`, m=`ltcounts.dtx`, n=`ltlength.dtx`, o=`ltfssbas.dtx`, p=`ltfssaxes.dtx`, q=`ltfsstrc.dtx`, r=`ltfsscmp.dtx`, s=`ltfssdcl.dtx`, t=`ltfssini.dtx`, u=`fontdef.dtx`, v=`preload.dtx`, w=`ltfntcmd.dtx`, x=`lttextcomp.dtx`, y=`ltpageno.dtx`, z=`ltxref.dtx`, A=`ltmiscen.dtx`, B=`ltmath.dtx`, C=`ltlists.dtx`, D=`ltboxes.dtx`, E=`lttab.dtx`, F=`ltpictur.dtx`, G=`ltthm.dtx`, H=`ltsect.dtx`, I=`ltfloat.dtx`, J=`ltidxglo.dtx`, K=`ltbibl.dtx`, L=`ltpage.dtx`, M=`ltoutput.dtx`, N=`ltclass.dtx`, O=`lthyphen.dtx`, P=`ltluatex.dtx`, Q=`ltfinal.dtx`

File Key: a=ltdirchk.dtx, b=ltplain.dtx, c=ltvers.dtx, d=ltdefns.dtx, e=ltalloc.dtx, f=ltcntrl.dtx, g=lterror.dtx, h=ltpar.dtx, i=ltspace.dtx, j=ltlogos.dtx, k=ltfiles.dtx, l=ltoutenc.dtx, m=ltcounts.dtx, n=ltlength.dtx, o=ltfssbas.dtx, p=ltfssaxes.dtx, q=ltfsstrc.dtx, r=ltfsscmp.dtx, s=ltfssdcl.dtx, t=ltfssini.dtx, u=fontdef.dtx, v=preload.dtx, w=ltfntcmd.dtx, x=lttextcomp.dtx, y=ltpageno.dtx, z=ltxref.dtx, A=ltmiscen.dtx, B=ltmath.dtx, C=ltlists.dtx, D=ltboxes.dtx, E=lttab.dtx, F=ltpictur.dtx, G=ltthm.dtx, H=ltsect.dtx, I=ltfloat.dtx, J=ltidxglo.dtx, K=ltbibl.dtx, L=ltpage.dtx, M=ltoutput.dtx, N=ltclass.dtx, O=lthyphen.dtx, P=ltluatex.dtx, Q=ltfinal.dtx

File Key: a=ltdirchk.dtx, b=ltplain.dtx, c=ltvers.dtx, d=ltdefns.dtx, e=ltalloc.dtx, f=ltcntrl.dtx, g=lterror.dtx, h=ltpar.dtx, i=ltspace.dtx, j=ltlogos.dtx, k=ltfiles.dtx, l=ltoutenc.dtx, m=ltcounts.dtx, n=ltlength.dtx, o=ltfssbas.dtx, p=ltfssaxes.dtx, q=ltfsstrc.dtx, r=ltfsscmp.dtx, s=ltfssdcl.dtx, t=ltfssini.dtx, u=fontdef.dtx, v=preload.dtx, w=ltfntcmd.dtx, x=lttextcomp.dtx, y=ltpageno.dtx, z=ltxref.dtx, A=ltmiscen.dtx, B=ltmath.dtx, C=ltlists.dtx, D=ltboxes.dtx, E=lttab.dtx, F=ltpictur.dtx, G=ltthm.dtx, H=ltsect.dtx, I=ltfloat.dtx, J=ltidxglo.dtx, K=ltbibl.dtx, L=ltpage.dtx, M=ltoutput.dtx, N=ltclass.dtx, O=lthyphen.dtx, P=ltluatex.dtx, Q=ltfinal.dtx

File Key: a=ltdirchk.dtx, b=ltplain.dtx, c=ltvers.dtx, d=ltdefns.dtx, e=ltalloc.dtx, f=ltcntrl.dtx, g=lterror.dtx, h=ltpar.dtx, i=ltspace.dtx, j=ltlogos.dtx, k=ltfiles.dtx, l=ltoutenc.dtx, m=ltcounts.dtx, n=ltlength.dtx, o=ltfssbas.dtx, p=ltfssaxes.dtx, q=ltfsstrc.dtx, r=ltfsscmp.dtx, s=ltfssdcl.dtx, t=ltfssini.dtx, u=fontdef.dtx, v=preload.dtx, w=ltfntcmd.dtx, x=lttextcomp.dtx, y=ltpageno.dtx, z=ltxref.dtx, A=ltmiscen.dtx, B=ltmath.dtx, C=ltlists.dtx, D=ltboxes.dtx, E=lttab.dtx, F=ltpictur.dtx, G=ltthm.dtx, H=ltsect.dtx, I=ltfloat.dtx, J=ltidxglo.dtx, K=ltbibl.dtx, L=ltpage.dtx, M=ltoutput.dtx, N=ltclass.dtx, O=lthyphen.dtx, P=ltluatex.dtx, Q=ltfinal.dtx

File Key: a=ltdirchk.dtx, b=ltplain.dtx, c=ltvers.dtx, d=ltdefns.dtx, e=ltalloc.dtx, f=ltcntrl.dtx, g=lterror.dtx, h=ltpar.dtx, i=ltspace.dtx, j=ltlogos.dtx, k=ltfiles.dtx, l=ltoutenc.dtx, m=ltcounts.dtx, n=ltlength.dtx, o=ltfssbas.dtx, p=ltfssaxes.dtx, q=ltfsstrc.dtx, r=ltfsscmp.dtx, s=ltfssdcl.dtx, t=ltfssini.dtx, u=fontdef.dtx, v=preload.dtx, w=ltfntcmd.dtx, x=lttextcomp.dtx, y=ltpageno.dtx, z=ltxref.dtx, A=ltmiscen.dtx, B=ltmath.dtx, C=ltlists.dtx, D=ltboxes.dtx, E=lttab.dtx, F=ltpictur.dtx, G=ltthm.dtx, H=ltsect.dtx, I=ltfloat.dtx, J=ltidxglo.dtx, K=ltbibl.dtx, L=ltpage.dtx, M=ltoutput.dtx, N=ltclass.dtx, O=lthyphen.dtx, P=ltluatex.dtx, Q=ltfinal.dtx

File Key: a=ltdirchk.dtx, b=ltplain.dtx, c=ltvers.dtx, d=ltdefns.dtx, e=ltalloc.dtx, f=ltcntrl.dtx, g=lterror.dtx, h=ltpar.dtx, i=ltspace.dtx, j=ltlogos.dtx, k=ltfiles.dtx, l=ltoutenc.dtx, m=ltcounts.dtx, n=ltlength.dtx, o=ltfssbas.dtx, p=ltfssaxes.dtx, q=ltfsstrc.dtx, r=ltfsscmp.dtx, s=ltfssdcl.dtx, t=ltfssini.dtx, u=fontdef.dtx, v=preload.dtx, w=ltfntcmd.dtx, x=lttextcomp.dtx, y=ltpageno.dtx, z=ltxref.dtx, A=ltmiscen.dtx, B=ltmath.dtx, C=ltlists.dtx, D=ltboxes.dtx, E=lttab.dtx, F=ltpictur.dtx, G=ltthm.dtx, H=ltsect.dtx, I=ltfloat.dtx, J=ltidxglo.dtx, K=ltbibl.dtx, L=ltpage.dtx, M=ltoutput.dtx, N=ltclass.dtx, O=lthyphen.dtx, P=ltluatex.dtx, Q=ltfinal.dtx

File Key: a=ltdirchk.dtx, b=ltplain.dtx, c=ltvers.dtx, d=ltdefns.dtx, e=ltalloc.dtx, f=ltcntrl.dtx, g=lterror.dtx, h=ltpar.dtx, i=ltspace.dtx, j=ltlogos.dtx, k=ltfiles.dtx, l=ltoutenc.dtx, m=ltcounts.dtx, n=ltlength.dtx, o=ltfssbas.dtx, p=ltfssaxes.dtx, q=ltfsstrc.dtx, r=ltfsscmp.dtx, s=ltfssdcl.dtx, t=ltfssini.dtx, u=fontdef.dtx, v=preload.dtx, w=ltfntcmd.dtx, x=lttextcomp.dtx, y=ltpageno.dtx, z=ltxref.dtx, A=ltmiscen.dtx, B=ltmath.dtx, C=ltlists.dtx, D=ltboxes.dtx, E=lttab.dtx, F=ltpictur.dtx, G=ltthm.dtx, H=ltsect.dtx, I=ltfloat.dtx, J=ltidxglo.dtx, K=ltbibl.dtx, L=ltpage.dtx, M=ltoutput.dtx, N=ltclass.dtx, O=lthyphen.dtx, P=ltluatex.dtx, Q=ltfinal.dtx

File Key: a=ltdirchk.dtx, b=ltplain.dtx, c=ltvers.dtx, d=ltdefns.dtx, e=ltalloc.dtx, f=ltcntrl.dtx, g=lterror.dtx, h=ltpar.dtx, i=ltspace.dtx, j=ltlogos.dtx, k=ltfiles.dtx, l=ltoutenc.dtx, m=ltcounts.dtx, n=ltlength.dtx, o=ltfssbas.dtx, p=ltfssaxes.dtx, q=ltfsstrc.dtx, r=ltfsscmp.dtx, s=ltfssdcl.dtx, t=ltfssini.dtx, u=fontdef.dtx, v=preload.dtx, w=ltfntcmd.dtx, x=lttextcomp.dtx, y=ltpageno.dtx, z=ltxref.dtx, A=ltmiscen.dtx, B=ltmath.dtx, C=ltlists.dtx, D=ltboxes.dtx, E=lttab.dtx, F=ltpictur.dtx, G=ltthm.dtx, H=ltsect.dtx, I=ltfloat.dtx, J=ltidxglo.dtx, K=ltbibl.dtx, L=ltpage.dtx, M=ltoutput.dtx, N=ltclass.dtx, O=lthyphen.dtx, P=ltluatex.dtx, Q=ltfinal.dtx