

Grzegorz ‘Natr0r’ Murzynowski

**The gmdoc Package
i.e., gmdoc.sty and gmdocc.cls**

March 2010

Contents

a. The gmdoc.sty Package	4
Readme	4
Installation	4
Contents of the gmdoc.zip archive	5
Compiling of the documentation	5
Dependencies	5
Bonus: base drivers	6
Introduction	6
The user interface	6
Used terms	6
Preparing of the source file	7
The main input commands	8
Package options	10
The packages required	11
Automatic marking of definitions	12
Manual marking of the macros and environments	14
Index ex/inclusions	16
The DocStrip directives	17
The changes history	17
The parameters	19
The narration macros	22
A queerness of \label	24
doc-compatibility	25
The driver part	25
The code	27
The package options	28
The dependencies and preliminaries	29
The core	33
Numbering (or not) of the lines	44
Spacing with \everypar	45
Life among queer EOLs	46
Adjustments of verbatim and \verb	48
Macros for marking of the macros	49
Automatic detection of definitions	54
Indexing of CSes	65
Index exclude list	79
Index parameters	84
The DocStrip directives	86
The changes history	87
The checksum	92
Macros from ltxdoc	94
\DocInclude and the ltxdoc-like setup	95
Redefinition of \maketitle	100
The file's date and version information	104
Miscellanea	105
doc-compatibility	112
gmdocing doc.dtx	117
\OCRInclude	118
Polishing, development and bugs	118
[No] <eof>	119
b. The gmdocc Class For gmdoc Driver	120
Files	120
Intro	120
Usage	120
The Code	121
c. The gmutils Package	127
Intro	127
Contents of the gmutils.zip archive	127
Options	128
A couple of abbreviations	128
\ifs as a four-argument L ^A T _E X command robust to unbalanced \ifs and \fis	129
\firstofone and the queer \catcodes	129
\afterfi and pals	129
\foone	130
\@ifempty, \IfAmong, \IfIntersect \@ifinmeaning	130
Global Boolean switches	131
\gm@ifundefined—a test that doesn't create any hash entry unlike \ifundefined	133
Some 'other' and active stuff	134
\@ifnextcat, \@ifnextac, catcode-independent \gm@ifstar, \@ifnextsin- gle, \@ifnextgroup	136
Storing and restoring the meanings of CSes	139
\DeclareCommand and \DeclareEnvironment	143
Ampulex Compressa-like modifications of macros	161
Environments redefined	163
Almost an environment or redefinition of \begin	163
\@ifenvir and improvement of \end	164

Storing and restoring the catcodes	
of specials	165
From <code>resize</code>	165
Meta-symbols	166
Macros for printing macros and filenames	167
Typesetting arguments and commands	170
Not only preamble!	173
Third person pronouns	174
Improvements to <code>mwcls</code> sectioning commands	174
An improvement of MW's \SetSectionFormatting	176
Negative <code>\addvspace</code>	177
My heading setup for <code>mwcls</code>	179
Compatibilising standard and <code>mwcls</code> sectionings	180
<code>enumerate*</code> and <code>itemize*</code>	181
The logos	182
Expandable turning stuff all into 'other'	185
Brave New World of X _E T _X	185
Fractions	186
\resizographics	187
Settings for mathematics in main font	188
Minion and Garamond Premier kerning and ligature fixes	200
A left-slanted font	200
Fake Old-style Numbers	201
Varia	205
\include not only .tex's	210
Switching on and off parts of one file	211
Fix of including when <code>fontspec</code> is used	212
Faked small caps	213
See above/see below	214
luzniej and napa- pierki—environments used in page breaking for money	214
Typesetting dates in my memoirs	218
For dati under poems	222
Thousand separator	223
hyperref's \nolinkurl into \url*	224
Footnotes suggested by Andrzej Tomaszewski	224
A fix to the <code>url</code> package	226
Conditional tilde	230
Storing the catcode of line end	231
A really empty page	231
d. The <code>gmiflink</code> Package	233
Introduction, usage	233
Contents of the <code>gmiflink.zip</code> archive	233
The code	234
e. The <code>gmverb</code> Package	236
Intro, usage	236
The package options	238
Contents of the <code>gmverb.zip</code> archive	238
The code	239
Preliminaries	239
The breakables	239
Almost-Knuthian \ttverbatim	240
The core: from <code>shortverb</code>	241
doc- and <code>shortverb</code> -compatibility	247
Grey visible spaces	248
Verbatim specials—CSes in verbatims	248
Partial \verb in arguments	250
f. The <code>gmoldcomm</code> Package	252
Change History	254
Index	263

a. The gmdoc.sty Package¹

March 4, 2010

This is (a documentation of) file gmdoc.sty, intended to be used with L^AT_EX 2_E as a package for documenting (L^A)T_EX files and to be documented with itself.

Written by Natror (Grzegorz Murzynowski),
natror at o2 dot pl

© 2006, 2007, 2008, 2009, 2010 by Natror (Grzegorz Murzynowski).

This program is subject to the L^AT_EX Project Public License.

See [http://www.ctan.org/tex--archive/help/Catalogue/licenses.lppl.html](http://www.ctan.org/tex-archive/help/Catalogue/licenses.lppl.html) for the details of that license.

LPPL status: "author-maintained".

Many thanks to my T_EX Guru Marcin Woliński for his T_EXnical support.

```
84 \ifnum\catcode`@=11 % Why this test here—will come out in chapter The driver.  
87 \NeedsTeXFormat{LaTeX2e}  
88 \ProvidesPackage{gmdoc}  
89 [2010/03/01 v0.991 a documenting package (GM)]  
90 \fi
```

Readme

This package is a tool for documenting of (L^A)T_EX packages, classes etc., i.e., the .sty, .cls files etc. The author just writes the code and adds the commentary preceded with % sign (or another properly declared). No special environments are necessary.

The package tends to be (optionally) compatible with the standard doc.sty package, i.e., the .dtx files are also compilable with gmdoc (they may need a tiny adjustment in some special cases).

The tools are integrated with hyperref's advantages such as hyperlinking of index entries, contents entries and cross-references.

The package also works with X_ET_EX (switches automatically).

Installation

Unpack the gmdoc-tds.zip archive (this is an archive conforming the TDS standard, see CTAN/tds/tds.pdf) in a texmf directory or put the gmdoc.sty, gmdoc.cls and gmoldcomm.sty somewhere in the texmf/tex/latex branch on your own. (Creating a texmf/tex/latex/gm directory may be advisable if you consider using other packages written by me. And you *have* to use four of them to make gmdoc work.)

¹ This file has version number v0.991 dated 2010/03/01.

You should also install `gmverb.sty`, `gmutils.sty` and `gmiflink.sty` (e.g., put them into the same `gm` directory). These packages are available on CTAN as separate .zip archives also in TDS-compliant zip archives.

Moreover, you should put the `gmglo.ist` file, a MakeIndex style for the changes' history, into some `texmf/makeindex` (sub)directory.

Then you should refresh your \TeX distribution's files' database most probably.

Contents of the `gmdoc.zip` archive

The distribution of the `gmdoc` package consists of the following five files and a TDS-compliant archive.

```
gmdoc.sty  
gmdocc.cls  
gmglo.ist  
README  
gmdoc.pdf  
gmdoc.tds.zip
```

Compiling of the documentation

The last of the above files (the `.pdf`, i.e., *this file*) is a documentation compiled from the `.sty` and `.cls` files by running \XeLaTeX on the `gmdoc.sty` twice (`xelatex gmdoc.sty` in the directory you wish the documentation to be in, you don't have copy the `.sty` file there, \TeX will find it), then MakeIndex on the `gmdoc.idx` and `gmdoc.glo` files, and then \XeLaTeX on `gmdoc.sty` once more. (Using \LaTeX instead of \XeLaTeX should do, too.)

MakeIndex shell commands:

```
makeindex -r gmdoc  
makeindex -r -s gmglo.ist -o gmdoc.gls gmdoc.glo
```

The `-r` switch is to forbid MakeIndex to make implicit ranges since the (code line) numbers will be hyperlinks.

Compiling the documentation requires the packages: `gmdoc` (`gmdoc.sty` and `gmdocc.cls`), `gmutils.sty`, `gmverb.sty`, `gmiflink.sty` and also some standard packages: `hyperref.sty`, `xcolor.sty`, `geometry.sty`, `multicol.sty`, `lmodern.sty`, `fontenc.sty` that should be installed on your computer by default.

If you had not installed the `mwcls` classes (available on CTAN and present in \TeX Live e.g.), the result of your compilation might differ a bit from the `.pdf` provided in this `.zip` archive in formatting: If you had not installed `mwcls`, the standard `article.cls` class would be used.

Dependencies

The `gmdoc` bundle depends on some other packages of mine:

```
gmutils.sty,  
gmverb.sty,  
gmiflink.sty
```

and also on some standard packages:

```
hyperref.sty,  
color.sty,  
geometry.sty,  
multicol.sty,  
lmodern.sty,
```

`fontenc.sty`

that should be installed on your computer by default.

Bonus: base drivers

As a bonus and example of doc-compatibility there are driver files included (cf. Palestina, *Missa papae Marcelli* ;-):

`source2e_gmdoc.tex`
`docstrip_gmdoc.tex`
`doc_gmdoc.tex`

`gmoldcomm.sty`
(`gmsource2e.ist` is generated from `source2e_gmdoc.tex`)

These drivers typeset the respective files from the

`.../texmf-dist/source/latex/base`

directory of the TeXLive2007 distribution (they only read that directory).

Probably you should redefine the `\BasePath` macro in them so that it points that directory on your computer.

Introduction

There are very sophisticated and effective tools for documenting L^AT_EX macro packages, namely the `doc` package and the `ltxdoc` class. Why did I write another documenting package then?

I like comfort and `doc` is not comfortable enough for me. It requires special marking of the macro code to be properly typeset when documented. I want T_EX to know 'itself' where the code begins and ends, without additional marks.

That's the difference. One more difference, more important for the people for whom the `doc`'s conventions are acceptable, is that `gmdoc` makes use of hyperref advantages and makes a hyperlinking index and toc entries and the cross-references, too. (The CSes in the code maybe in the future.)

The rest is striving to level the very high `doc/ltxdoc`'s standard, such as (optional) numbering of the codelines and automatic indexing the control sequences e.g.

The `doc` package was and still is a great inspiration for me and I would like this humble package to be considered as a sort of homage to it². If I mention copying some code or narrative but do not state the source explicitly, I mean the `doc` package's documentation (I have v2.1b dated 2004/02/09).

The user interface

Used terms

When I write of a **macro**, I mean a macro in *The T_EXbook*'s meaning, i.e., a control sequence whose meaning is `\[e|g|x]`defined. By a **macro's parameter** I mean each of `#<digit>`s in its definition. When I write about a **macro's argument**, I mean the value (list of tokens) substituting the corresponding parameter of this macro. (These understandings are according to *The T_EXbook*, I hope: T_EX is a religion of Book ;-).)

I'll use a shorthand for 'control sequence', **CS**.

² As Grieg's Piano Concerto is a homage to the Schumann's.

When I talk of a **declaration**, I mean a macro that expands to a certain assignment, such as `\itshape` or `\@onlypreamble{<CS>}`.

Talking of declarations, I'll use the **OCSR** acronym as a shorthand for 'observes/ing common TeX scoping rules'.

By a **command** I mean a certain abstract visible to the end user as a CS but consisting possibly of more than one macro. I'll talk of a **command's argument** also in the 'sense -for-the-end-user', e.g., I'll talk of the `\verb command's argument` although *the macro \verb* has no #`<digit>` in its definition.

The **code** to be typeset verbatim (and with all the bells and whistles) is everything that's not commented out in the source file and what is not a leading space(s).

The **commentary** or **narrative** is everything after the comment char till the end of a line. The **comment char** is a character the `\catcode` of which is 14 usually i.e., when the file works; if you don't play with the `\catcodes`, it's just the % . When the file is documented with gmdoc, such a char is re`\catcoded` and its rôle is else: it becomes the **code delimiter**.

A line containing any TeX code (not commented out) will be called a **codeline**. A line that begins with (some leading spaces and) a code delimiter will be called a **comment line** or **narration line**.

The **user** of this package will also be addressed as **you**.

Not to favour any particular gender (of the amazingly rich variety, I mean, not of the vulgarly simplified two-element set), in this documentation I use alternating pronouns of third person (`\heshe` etc. commands provided by `gutils`), so let one be not surprised if 'they' sees 'themself' altered in the same sentence :-).

Preparing of the source file

When (L)TeX with `gmdoc.sty` package loaded typesets the comment lines, the code delimiter is omitted. If the comment continues a codeline, the code delimiter is printed. It's done so because ending a TeX code line with a % is just a concatenation with the next line sometimes. Comments longer than one line are typeset continuously with the code delimiters omitted.

The user should just write their splendid code and brilliant commentary. In the latter they may use usual (L)TeX commands. The only requirement is, if an argument is divided in two lines, to end such a dividing line with `\^M (<line end>)` or with `\^B` sequence that'll enter the (active) `<char2>` which shall gobble the line end.

But there is also a gmdoc version of `\footnote` provided that sets the catcodes so that you don't bother about `\^B` in the argument, `\qfootnote` that takes the same argument(s) as the standard `\footnote` and for emphasis there is `\qemph{<text to emphasize>}` . Both of them work also in the 'straight' EOLs' scope so you don't bother. The `\arg gutils` command also works without `\^B`.

Moreover, if they wants to add a meta-comment i.e., a text that doesn't appear in the code layer nor in the narrative, they may use the `\^A` sequence that'll be read by TeX as `<char1>`, which in gmdoc is active and defined to gobble the stuff between itself and the line end.

Note that `\^A` behaves much like comment char although it's active in fact: it re`\catcode`s the special characters including \, { and } so you don't have to worry about unbalanced braces or `\ifs` in its scope. But `\^B` doesn't re`\catcode` anything (which would be useless in an argument) so any text between `\^B` and line end has to be balanced.

However, it may be a bit confusing for someone acquainted with the doc conventions. If you don't fancy the `\^B` special sequence, instead you may restore the standard meaning of the line end with the `\StraightEOL` declaration which OCSR. As almost all the control sequences, it may be used also as an environment, i.e., `\begin{StraightEOL}`

\QueerEOL ... \end{StraightEOL}. However, if for any reason you don't want to make an environment (a group), there's a \StraightEOL's counterpart, the \QueerEOL declaration that restores again the queer³ gmdoc's meaning of the line end. It OCSR, too. One more point to use \StraightEOL is where you wish some code lines to be executed both while loading the file and during the documentation pass (it's analogous to doc's not embracing some code lines in a macrocode environment).

As in standard TeXing, one gets a paragraph by a blank line. Such a line should be %ed of course. A fully blank line is considered a blank *code line* and hence results in a vertical space in the documentation. As in the environments for poetry known to me, subsequent blank lines do not increase such a space.

Then they should prepare a main document file, a **driver** henceforth, to set all the required formattings such as \documentclass, paper size etc., and load this package with a standard command i.e., \usepackage{gmdoc}, just as doc's documentation says:

"If one is going to document a set of macros with the [gm]doc package one has to prepare a special driver file which produces the formatted document. This driver file has the following characteristics:

```
\documentclass[<options>]{<document class>}
\usepackage[<options, probably none>]{gmdoc}
  <preamble>
\begin{document}
  <special input commands>
\end{document}
"
```

The main input commands

\DocInput To typeset a source file you may use the \DocInput macro that takes the (path and name of the file *with the extension* as the only argument, e.g., \DocInput{mybril|liantpackage.sty}⁴.

(Note that an *installed* package or class file is findable to TeX even if you don't specify the path.)

\OldDocInput If a source file is written with rather doc than gmdoc in mind, then the \OldDocInput command may be more appropriate (e.g., if you break the arguments of commands in the commentary in lines). It also takes the file (path and) name as the argument.

macrocode When using \OldDocInput, you have to wrap all the code in macrocode environments, which is not necessary when you use \DocInput. Moreover, with \OldDocInput the macrocode[*] environments require to be ended with

```
%……\end{macrocode[*]}
```

as in doc. (With \DocInput you are not obliged to precede \end{macrocode[*]} with The Four Spaces.)

\DocInclude If you wish to document many files in one document, you are provided \DocIn|clude command, analogous to LATEX's \include and very likely to ltxdoc's command of the same name. In gmdoc it has one mandatory argument that should be the file name *without extension*, just like for \include.

³ In my understanding 'queer' and 'straight' are not the opposites excluding each other but the counterparts that may cooperate in harmony for people's good. And, as I try to show with the \QueerEOL and \StraightEOL declarations, 'queer' may be very useful and recommended while 'straight' is the standard but not necessarily normative.

⁴ I use the 'broken bar' character as a hyphen in verbatim texts and hyperlinks. If you don't like it, see \verbDiscretionaryHyphen in gmverb.

The file extensions supported by \DocInclude are .fdd, .dtx, .cls, .sty, .tex and .fd. The macro looks for one of those extensions in the order just given. If you need to document files of other extensions, please let me know and most probably we'll make it possible.

\DocInclude has also an optional first argument that is intended to be the path of the included file with the levels separated by / (slash) and also ended with a slash. The path given to \DocInclude as the first and optional argument will not appear in the headings nor in the footers.

\maketitle

\DocInclude redefines \maketitle so that it makes a chapter heading or, in the classes that don't support \chapter, a part heading, in both cases with respective toc entries. The default assumption is that all the files have the same author(s) so there's no need to print them in the file heading. If you wish the authors names to be printed, you should write \PrintFilesAuthors in the preamble or before the relevant \DocIncludes. If you wish to undeclare printing the authors names, there is \SkipFilesAuthors declaration.

Like in ltxdoc, the name of an included file appears in the footer of each page with date and version info (if they are provided).

The \DocIncluded files are numbered with the letters, the lowercase first, as in ltxdoc. Such a file-marker also precedes the index entries, if the (default) codeline index option is in force.

As with \include, you may declare \includeonly {\textcolor{red}{filenames separated with commas}} for the draft versions.

If you want to put the driver into the same .sty or .cls file (see chapter 1928 to see how), you may write \DocInput {\jobname.sty}, or \DocInclude {\jobname}, but there's also a shorthand for the latter \SelfInclude that takes no arguments. By the way, to avoid an infinite recursive input of .aux files in the case of self-inclusion an .auxx file is used instead of (main) .aux.

By the way, to say TeX to (self)include only the current file, most probably you should say \includeonly {\jobname} not \includeonly {myfile} because of the catcodes.

At the default settings, the \(\textcolor{red}{(Doc|Self)}\)\Included files constitute chapters if \chapter is known and parts otherwise. The \maketitles of those files result in the respective headings.

If you prefer more ltxdocish look, in which the files always constitute the parts and those parts have a part's title pages with the file name and the files' \maketitles result in (article-like) titles not division headings, then you are provided the \ltxLookSetup declaration (allowed only in the preamble). However, even after this declaration the files will be included according to gmdoc's rules not necessarily to the doc's ones (i.e., with minimal marking necessary at the price of active line ends (therefore not allowed between a command and its argument nor inside an argument)).

On the other hand, if you like the look offered by me but you have the files prepared for doc not for gmdoc, then you should declare \olddocIncludes. Unlike the previous one, this may be used anywhere, because I have the account of including both doc-like and gmdoc-like files into one document. This declaration just changes the internal input command and doesn't change the sectioning settings.

It seems possible that you wish to document the 'old-doc' files first and the 'new-doc' ones after, so the above declaration has its counterpart, \gmdocIncludes, that may be used anywhere, too. Before the respective \DocInclude(s), of course.

Both these declarations OCSR.

If you wish to document your files as with ltxdoc *and* as with doc, you should declare \ltxLookSetup in the preamble *and* \olddocIncludes.

\ltxPageLayout	Talking of analogies with \ltxdoc, if you like only the page layout provided by that class, there is the \ltxPageLayout declaration (allowed only in preamble) that only changes the margins and the text width (it's intended to be used with the default paper size). This declaration is contained in the \ltxLookSetup declaration.
\AtBeginInput	If you need to add something at the beginning of the input of file, there's the \AtBeginInput declaration that takes one mandatory argument which is the stuff to be added. This declaration is global. It may be used more than one time and the arguments of each occurrence of it add up and are put at the beginning of input of every subsequent files.
\AtEndInput	Simili modo, for the end of input, there's the \AtEndInput declaration, also one-argument, global and cumulative.
\AtBeginInputOnce	If you need to add something at the beginning of input of only one file, put before the respective input command an \AtBeginInputOnce {<the stuff to be added>} declaration. It's also global which means that the groups do not limit its scope but it adds its argument only at the first input succeeding it (the argument gets wrapped in a macro that's \relaxed at the first use). \AtBeginInputOnce add up, too.
\IndexInput	One more input command is \IndexInput (the name and idea of effect comes from doc). It takes the same argument as \DocInput, the file's (path and) name with extension. (It has \DocInput inside). It works properly if the input file doesn't contain explicit <char1> (^A is OK).
	The effect of this command is typesetting of all the input file verbatim, with the code lines numbered and the CSes automatically indexed (gmdoc.sty options are in force).
	Package options
	As many good packages, this also provides some options:
linesnotnum	Due to best T _E X documenting traditions the codelines will be numbered. But if the user doesn't wish that, they may turn it off with the linesnotnum option.
uresetlinecount	However, if they agrees to have the lines numbered, they may wish to reset the counter of lines themself, e.g., when they documents many source files in one document. Then they may wish the line numbers to be reset with every {section}'s turn for instance. This is the rôle of the uresetlinecount option, which seems to be a bit obsolete however, since the \DocInclude command takes care of a proper reset.
countalllines	Talking of line numbering further, a tradition seems to exist to number only the codelines and not to number the lines of commentary. That's the default behaviour of gmdoc but, if someone wants the comment lines to be numbered too, which may be convenient for reference purposes, they is provided the countalllines option. This option switches things to use the \inputlineno primitive for codeline numbers so you get the numbers of the source file instead of number only of the codelines. Note however, that there are no hypertargets made to the narration lines and the value of \ref is the number of the most recent codeline.
countalllines*	Moreover, if they wants to get the narration lines' number printed, there is the starred version of that option, countalllines*. I imagine someone may use it for debug. This option is not finished in details, it causes errors with \addvspace because it puts a hyperlabel at every line. When it is in force, all the index entries are referenced with the line numbers and ⁴⁴¹ the narration acquires a bit biblical look ;-), ⁴⁴² as shown in this short example. This option is intended ⁴⁴³ for the draft versions and it is not perfect (as if anything ⁴⁴⁴ in this package was). As you see, the lines ⁴⁴⁵ are typeset continuously with the numbers printed.
noindex	By default the makeidx package is loaded and initialised and the CSes occurring in the code are automatically (hyper)indexed thanks to the hyperref package. If the user doesn't wish to index anything, she should use the noindex option.

`pageindex` The index comes two possible ways: with the line numbers (if the lines are numbered) and that's the default, or with the page numbers, if the `pageindex` option is set.

The references in the change history are of the same: when `index` is line number, then the changes history too.

By default, `gmdoc` excludes some 300 CSes from being indexed. They are the most common CSes, L^AT_EX internal macros and T_EX primitives. To learn what CSes are excluded actually, see lines [5681–5807](#).

`indexallmacros` If you don't want all those exclusions, you may turn them off with the `indexallmacros` option.

If you have ambiguous feelings about whether to let the default exclusions or forbid them, see p. [16](#) to feed this ambiguity with a couple of declarations.

In `doc` package there's a default behaviour of putting marked macro's or environment's name to a marginpar. In the standard classes it's alright but not all the classes support marginpars. That is the reason why this package enables marginpar-ing when in standard classes, enables or disables it due to the respective option when with Marcin Woliński's classes and in any case provides the options `withmarginpar` and `nomarginpar`. So, in non-standard classes the default behaviour is to disable marginpars. If the marginpars are enabled in `gmdoc`, it will put marked control sequences and environments into marginpars (see [\TextUsage etc.](#)). These options do not affect common using marginpars, which depends on the document class.

My suggestion is to make the spaces in the code visible except the leading ones and that's the default. But if you wish all the code spaces to be blank, I give the option `codespacesblank` reluctantly. Moreover, if you wish the code spaces to be blank only in some areas, then there's `\CodeSpacesBlank` declaration (OCSR).

Another space formatting option is `codespacesgrey` suggested by Will Robertson. It makes the spaces of code visible only not black but grey. The name of their colour is `visspacesgrey` and by default it's defined as `{gray}{.5}`, you can change it with `xcolor`'s `\definecolor`. There is also an OCSR declaration `\CodeSpacesGrey`.

If for any reason you wish the code spaces blank in general and visible and grey in `verbatim*`s, use the declaration `\VisSpacesGrey` of the `gmverb` package. If you like little tricks, you can also specify `codespacesgrey`, `\codespacesblank` in `gmdoc` options (in this order).

The packages required

`gmdoc` requires (loads if they're not loaded yet) some other packages of mine, namely `gmutils`, `gmverb`, analogous to Frank Mittelbach's `shortverb`, and `gmiflink` for conditional making of hyperlinks. It also requires `hyperref`, `multicol`, `color` and `makeidx`.

`gmverb` The `gmverb` package redefines the `\verb` command and the `verbatim` environment in such a way that `\`, `{` and `}` are breakable, the first with no 'hyphen' and the other two with the comment char as a hyphen, i.e., `{<subsequent text>}` breaks into `{%` `<subsequent text>` `}` and `<text>\mylittlemacro` breaks into `<text>%` `\mylittlemacro`.

This package provides the `\verbatimspecials` declaration that is used in `gmdoc.cls` as

```
\verbatimspecials /«»[;]
```

to set `/` (fractional slash) to the escape char, `«` and `»` to group begin and end respectively and `;` to math shift in `verbatims` (also the short ones). Note however that this declaration *has no effect on the code layer*.

\verb+eolK As the standard L^AT_EX one, my \verb issues an error when a line end occurs in its scope. But, if you'd like to allow line ends in short verbatims, there's the \verb+eolK declaration. The plain \verb typesets spaces blank and \verb* makes them visible, as in the standard version(s).

\MakeShortVerb Moreover, gmverb provides the \MakeShortVerb declaration that takes a one-char control sequence as the only argument and turns the char used into a short verbatim delimiter, e.g., after

```
\MakeShortVerb* \|
```

(as you see, the declaration has the starred version, which is for visible spaces, and non-starred for blank spaces) to get \mylittlemacro you may type | \mylittlemacro | instead of \verb+\mylittlemacro+. Because the char used in the last example is my favourite and is used this way by DEK in *The T_EXbook*'s format, gmverb provides a macro \dekclubs that expands to the example displayed above.

Be careful because such active chars may interfere with other things, e.g., the | with the vertical line marker in tabulars and with the tikz package. If this happens, you can declare e.g., \DeleteShortVerb \| and the previous meaning of the char used shall be restored.

One more difference between gmverb and shortverb is that the chars \activeated by \MakeShortVerb, behave as if they were 'other' in math mode, so you may type e.g., \\$k|n\\$ to get k|n etc.

gutils The gutils package provides a couple of macros similar to some basic (L^A)T_EX ones, rather strictly technical and (I hope) tricky, such as \afterfi, \ifnextcat, \ad| dtomacro etc. It's this package that provides the macros for formatting of names of macros and files, such as \cs, \marg, \pk etc. Moreover, it provides a powerful tool for defining commands with weird optional and Knuthian arguments, \DeclareCom|m|and, inspired by ancient (pre-expl3) xparse's \DeclareDocumentCommand.

hyperref The gmdoc package uses a lot of hyperlinking possibilities provided by hyperref which is therefore probably the most important package required. The recommended situation is that the user loads hyperref package with their favourite options *before* loading gmdoc.

If they does not, gmdoc shall load it with *my* favourite options.

gmiflink To avoid an error if a (hyper)referenced label does not exist, gmdoc uses the gmiflink package. It works e.g., in the index when the codeline numbers have been changed: then they are still typeset, only not as hyperlinks but as a common text.

multicol To typeset the index and the change history in balanced columns gmdoc uses the multicol package that seems to be standard these days.

color Also the multicol package, required to define the default colour of the hyperlinks, seems to be standard already, and makeidx.

Automatic marking of definitions

gmdoc implements automatic detection of a couple of definitions. By default it detects all occurrences of the following commands in the code:

1. \def, \newcount, \newdimen, \newskip, \newif, \newtoks, \newbox,
 \newread,
 \newwrite, \newlength, \newcommand[*, \renewcommand[*,
 \providecommand[*, \DeclareRobustCommand[*,
 \DeclareTextCommand[*,
 \DeclareTextCommandDefault[*, \DeclareDocumentCommand,
 \DeclareCommand
2. \newenvironment[*, \renewenvironment[*, \DeclareOption,
3. \newcounter,

of the `xkeyval` package:

4. `\define@key`, `\define@boolkey`, `\define@choicekey`, `\DeclareOptionX`, and of the `kvoptions` package:
5. `\DeclareStringOption`, `\DeclareBoolOption`,
`\DeclareComplementaryOption`,
`\DeclareVoidOption`.

What does ‘detects’ mean? It means that the main argument of detected command will be marked as defined at this point, i.e. thrown to a margin note and indexed with a ‘definition’ entry. Moreover, for the definitions 3–5 an alternate index entries will be created: of the CSes underlying those definitions, e.g. `\newcounter{foo}` in the code will result in indexing `foo` and `\c@foo`.

`\DeclareDefining`
If you want to add detection of a defining command not listed above, use the `\DeclareDefining` declaration. It comes in two flavours: ‘sauté’ and with star. The ‘sauté’ version (without star and without an optional argument) declares a defining command of the kind of `\def` and `\newcommand`: its main argument, whether wrapped in braces or not, is a CS. The starred version (without the optional argument) declares a defining command of the kind of `\newenvironment` and `\DeclareOption`: whose main mandatory argument is text. Both versions provide an optional argument in which you can set the keys.

`type`
Probably the most important key is `type`. Its default value is `cs` and that is set in the ‘sauté’ version. Another possible value is `text` and that is set in the starred version. You can also set three other types (any keyval setting of the type overrides the default and ‘starred’ setting): `dk`, `dox` or `kvo`.

`dk` stands for `\define@key` and is the type of `xkeyval` definitions of keys (group 4 commands). When detected, it scans further code for an optional [`<KVprefix>`], mandatory {`<KVfamily>`} and mandatory {`<key name>`}. The default `<KVprefix>` is `KV`, as in `xkeyval`.

`dox` stands for `\DeclareOptionX` and launches scanning for an optional [`<KVprefix>`], optional `<<KVfamily>>` and mandatory {`<option name>`}. Here the default `<KVprefix>` is also `KV` and the default `<KVfamily>` is the input file name. If you want to set another default family (e.g. if the code of `foo.sty` actually is in file `bar.dtx`), use `\DeclareDOXHead{<KVfamily>}`. This declaration has an optional first argument that is the default `<KVprefix>` for `\DeclareOptionX` definitions.

`kvo` stands for the `kvoptions` package by Heiko Oberdiek. This package provides a handful of option defining commands (the group 5 commands). Detection of such a command launches a scan for mandatory {`<option name>`} and alternate indexing of a CS `\<KVOfamily>@<option name>`. The default `<KVOfamily>` is the input file name. Again, if you want to set something else, you are given the `\DeclareKVOFam{<KVOfamily>}` that sets the default family (and prefix: `<KVOfamily>@`) for all the commands of group 5.

`star`
Next key recognised by `\DeclareDefining` is `star`. It determines whether the starred version of a defining command should be taken into account⁵. For example, `\newcommand` should be declared with `[star=true]` while `\def` with `[star=false]`. You can also write just `[star]` instead of `[star=true]`. It’s the default if the `star` key is omitted.

`KVpref`
`KVfam`
There are also `KVpref` and `KVfam` keys if you want to redeclare the `xkeyval` definitions with another default prefix and family.

For example, if you wish `\@namedef` to be detected (the original L^AT_EX version),

⁵ The `star` key is provided because the default setting of `\MakePrivateLetters` is such that `*` is a letter so e.g. `\newcommand*` is scanned as one CS. However, if the `\makestarlow` declaration is in force (e.g. with the `gmdoc`) this is not so—`\newcommand*` is scanned as the CS `\newcommand` and a star.

declare
 \DeclareDefining*[star=false] \@namedef

or

 \DeclareDefining[type=text, star=false] \@namedef

(as stated above, * is equivalent to [type=text]).

On the other hand, if you want some of the commands listed above *not* to be detected, write \HideDefining\<command> in the commentary. If both <command> and <command*> are detected, then both will be hidden. \HideDefining is always \global. Later you can resume detection of <command> and <command*> with \ResumeDefining<command> which is always \global too. Moreover, if you wish to suspend automatic detection of the defining <command> only once (the next occurrence), there is \HideDefining* which suspends detection of the next occurrence of <command>. So, if you wish to ‘hide’ \providecommand* once, write

 \HideDefining*\providecommand*

If you wish to turn entire detection mechanism off, write \HideAllDefining in the narration layer. Then you can resume detection with \ResumeAllDefining. Both declarations are \global.

The basic definition command, \def, seems to me a bit ambiguous. Definitely *not always* it defines important macros. But first of all, if you \def a CS excluded from indexing (see section [Index ex/inclusions](#)), it will not be marked even if detection of \def is on. But if the \def’s argument is not excluded from indexing and you still don’t want it to be marked at this point, you can write \HideDefining*\def or \UnDef for short.

If you don’t like \def to be detected more times, you may write \HideDefining%\def of course, but there’s a shorthand for this: \HideDef which has the starred version \HideDef* equivalent to \UnDef. To resume detection of \def you are provided also a shorthand, \ResumeDef (but \ResumeDefining\def also works).

Since I use \pdef most often, I provide also \UnPdef, analogous to \UnDef.

If you define things not with easily detectable commands, you can mark them ‘manually’, with the \Define declaration described in the next section.

Manual Marking of the Macros and Environments

The concept (taken from doc) is to index virtually all the control sequences occurring in the code. gmdoc does that by default and needs no special command. (See below about excluding some macros from being indexed.)

The next concept (also taken from doc) is to distinguish some occurrences of some control sequences by putting such a sequence into a marginpar and by special formatting of its index entry. That is what I call marking the macros. gmdoc provides also a possibility of analogous marking for the environments’ names and other sequences such as ^A.

This package provides two kinds of special formatting of the index entries: ‘usage’, with the reference number italic by default, and ‘def’ (in doc called ‘main’), with the reference number roman (upright) and underlined by default. All the reference numbers, also those with no special formatting, are made hyperlinks to the page or the codeline according to the respective indexing option (see p. [11](#)).

The macros and environments to be marked appear either in the code or in the commentary. But all the definitions appear in the code, I suppose. Therefore the ‘def’ marking macro is provided only for the code case. So we have the \Define, \CodeUsage and \TextUsage commands.

\Define
\CodeUsage
\TextUsage

The arguments to all three are as follows:

#1 [*] to indicate whether we mark a single CS or more than one token(s): without star for a single CS, with star for environment names etc., the starred version executes \@sanitize,

[#2] o version to be marginalized and printed here,

#3 m version to be put to the index, and also (printed here and) marginalized if the previous argument is missing.

Note that if you give a single CS to the starred version (e.g. the next \MakePrivateLetters is done so to hyphenate it in the text), you have to wrap it in braces because command \@sanitizes the specials including backslash.

You don't have to bother whether @ is a letter while documenting because even if not, these commands do make it a letter, or more precisely, they execute \MakePrivateLetters whatever it does: At the default settings this command makes * a letter, too, so a starred version of a command is a proper argument to any of the three commands unstarred.

The \Define and \CodeUsage commands, if unstarred, mark the next scanned occurrence of their argument in the code. (By 'scanned occurrence' I mean a situation of the CS having been scanned in the code which happens iff its name was preceded by the char declared as \CodeEscapeChar). The starred versions of those commands mark just the next codeline and don't make TeX looks for the scanned occurrence of their argument (which would never happen if the argument is not a CS). Therefore, if you want to mark a definition of an environment foo, you should put

```
%\Define{*}{foo}
```

right before the code line

```
\newenvironment{foo}{%
```

i.e., not separated by another code line. The starred versions of the \Code... commands are also intended to mark implicit definitions of macros, e.g., \Define*@\foofalse before the line

```
\newif\if@foo.
```

They both are \outer to discourage their use inside macros because they actually re\catcode before taking their arguments.

The \TextUsage (one-argument) command is intended to mark usage of a verbatim occurrence of a TeX object in the commentary. Unlike \CodeUsage or \Define, it typesets its argument which means among others that the marginpar appears usually at the same line as the text you wanted to mark. This command also has the starred version primarily intended for the environments names, and secondarily for ^A-likes and CSes, too. Currently, the most important difference is that the unstarred version executes \MakePrivateLetters while the starred does both \MakePrivateLetters and \MakePrivateOthers before reading the argument.

If you consider the marginpars a sort of sub(sub...)section marks, then you may wish to have a command that makes a marginpar of the desired CS(or whatever) at the beginning of its description, which may be fairly far from the first occurrence of its object. Then you have the \Describe command which puts its argument in a marginpar and indexes it as a 'usage' entry but doesn't print it in the text. It's \outer.

All four commands just described put their (\stringed) argument into a marginpar (if the marginpars are enabled) and create an index entry (if indexing is enabled).

But what if you want just to make a marginpar with macro's or environment's name? Then you have \CodeMarginize to declare what to put into a marginpar in the TeX code (it's \outer) and \TextMarginize to do so in the commentary. According to

the spirit of this part of the interface, these commands also take one argument and have their starred versions for strings other than control sequences.

\marginpartt
The marginpars (if enabled) are ‘reverse’ i.e., at the left margin, and their contents is flush right and typeset in a font declared with \marginpartt. By default, this declaration is \let to \tt but it may be advisable to choose a condensed font if there is any. Such a choice is made by gmdoc.cls if the Latin Modern fonts are available: in this case gmdoc.cls uses Latin Modern Typewriter Light Condensed.

\gmdmarginpar
If you need to put something in a marginpar without making it typewriter font, there’s the \gmdmarginpar macro (that takes one and mandatory argument) that only flushes its contents right.

\DefIndex
\CodeUsgIndex
On the other hand, if you don’t want to put a CS(or another verbatim text) in a marginpar but only to index it, then there are \DefIndex and \CodeUsgIndex to declare special formatting of an entry. The unstarred versions of these commands look for their argument’s scanned occurrence in the code (the argument should be a CS), and the starred ones just take the next code line as the reference point. Both these commands are \outer.

\CodeCommonIndex*
In the code all the control sequences (except the excluded ones, see below) are indexed by default so no explicit command is needed for that. But the environments and other special sequences are not and the two commands described above in their *ed versions contain the command for indexing their argument. But what if you wish to index a not scanned stuff as a usual entry? The \CodeCommonIndex* comes in rescue, starred for the symmetry with the two previous commands (without * it just gobbles it’s argument—it’s indexed automatically anyway). It’s \outer.

\TextUsgIndex
\TextCommonIndex
Similarly, to index a TeX object occurring verbatim in the narrative, you have \Text | UsgIndex and \TextCommonIndex commands with their starless versions for a CS argument and the starred for all kinds of the argument.

macro
environment
Moreover, as in doc, the **macro** and **environment** environments are provided. Both take one argument that should be a CS for **macro** and ‘whatever’ for **environment**. Both add the \MacroTopsep glue before and after their contents, and put their argument in a marginpar at the first line of their contents (since it’s done with \strut, you should not put any blank line (%ed or not) between \begin{macro/environment} and the first line of the contents). Then **macro** commands the first scanned occurrence of its argument to be indexed as ‘def’ entry and **environment** commands TeX to index the argument as if it occurred in the next code line (also as ‘def’ entry).

Since it’s possible that you define a CS implicitly i.e., in such a way that it cannot be scanned in the definition (with \csname ... \endcsname e.g.) and wrapping such a definition (and description) in an **environment** environment would look misguidedly ugly, there’s the **macro*** environment which TeXnically is just an alias for **environment**.

(To be honest, if you give a **macro** environment a non-CS argument, it will accept it and then it’ll work as **environment**.)

Index ex/inclusions

\DoNotIndex
It’s understandable⁶ that you don’t want some control sequences to be indexed in your documentation. The doc package gives a brilliant solution: the \DoNotIndex declaration. So do I (although here, TeXnically it’s done another way). It OCSR. This declaration takes one argument consisting of a list of control sequences not to be indexed. The items

⁶ After reading doc’s documentation ;-).

of this list may be separated with commas, as in `doc`, but it's not obligatory. The whole list should come in curly braces (except when it's one-element), e.g.,

```
\DoNotIndex {\some@macros, \are* \too\auxiliary\?}
```

(The spaces after the control sequences are ignored.) You may use as many `\DoNotIndex`s as you wish (about half as many as many CSes may be declared, because for each CS excluded from indexing a special CS is declared that stores the ban sentence). Excluding the same CS more than once makes no problem.

I assume you wish most of L^AT_EX macros, T_EX primitives etc. to be excluded from your index (as I do). Therefore `gmdoc` excludes some 300 CSes by default. If you don't like it, just set the `indexallmacros` package option.

On the third hand, if you like the default exclusions in general but wish to undo just a couple of them, you are given `\DoIndex` declaration (OCSR) that removes a ban on all the CSes given in the argument, e.g.,

```
\DoIndex {\par \@@par \endgraf}
```

Moreover, you are provided the `\DefaultIndexExclusions` and `\UndoDefaultIndexExclusions` declarations that act according to their names. You may use them in any configuration with the `indexallmacros` option. Both of these declarations OCSR.

The DocStrip directives

`gmdoc` typesets the DocStrip directives and it does it quite likely as `doc`, i.e., with math sans serif font. It does it automatically whether you use the traditional settings or the new.

Advised by my T_EX Guru, I didn't implement the module nesting recognition (MW told it's not that important.)

So far verbatim mode directive is only half-handled. That is, a line beginning with `%<<END-TAG>` will be typeset as a DocStrip directive, but the closing line `%END-TAG>` will be not. It doesn't seem to be hard to implement, if I only receive some message it's really useful for someone.

The changes history

The `doc`'s documentation reads:

"To maintain a change history within the file, the `\changes` command may be placed amongst the description part of the changed code. It takes three arguments, thus:

```
\changes[{\cs}]{\version}{\date}{\text}
```

or, if you prefer the `\ProvidesPackage/Class` syntax,

```
\chgs[{\cs}]{\date}{\version}{\text}
```

The optional `\cs` argument may be a CS(with backslash) or a string. By default it's the most recently defined CS (see section about automatic detection of definitions).

The `changes` may be used to produce an auxiliary file (L^AT_EX's `\glossary` mechanism is used for this) which may be printed after suitable formatting. The `\changes` [command] encloses the `\date` in parentheses and appends the `\text` to form the printed entry in such a change history [... obsolete remark omitted].

To cause the change information to be written out, include `\RecordChanges` in the driver['s preamble or just in the source file (`gmdocc.cls` does it for you)]. To read in and print the sorted change history (in two columns), just put the `\PrintChanges` com-

mand as the last (commented-out, and thus executed during the documentation pass through the file) command in your package file [or in the driver]. Alternatively, this command may form one of the arguments of the `\StopEventually` command, although a change history is probably not required if only the description is being printed. The command assumes that `MakeIndex` or some other program has processed the `.glo` file to generate a sorted `.gls` file. You need a special `MakeIndex` style file; a suitable one is supplied with `doc` [and `gmdoc`], called [... `gmglo.ist` for `gmdoc`]. The `\GlossaryMin`, `\GlossaryPrologue` and `\GlossaryParms` macros are analogous to the `\Index...` versions [see sec. [The parameters](#) p. 21]. (The L^AT_EX ‘glossary’ mechanism is used for the change entries.)”

In `gmdoc` (unless you turn definitions detection off), you can put `\changes` after the line of definition of a command to set the default argument of `\changes` to that command. For example,

```
\newcommand*\dodecaphonic{...}
% \changes{vo.99e}{2007/04/29}{renamed from
  \cs{DodecaPhonic}}
```

results with a history (sub)entry:

```
vo.99e
  ...
  \dodecaphonic:
    renamed from \DodecaPhonic, 18
```

Such a setting is in force till the next definition and *every* detected definition resets it. In `gmdoc` `\changes` is `\outer`.

As mentioned in the introduction, the glossary, the changes history that is, uses a special `MakeIndex` style, `gmglo.ist`. This style declares another set of the control chars but you don’t have to worry: `\changes` takes care of setting them properly. To be precise, `\changes` executes `\MakeGlossaryControls` that is defined as

```
\def\actualchar{=}\def\quotechar{!}%
\def\levelchar{>}\edef\encapchar{\xiiclub}
```

Only if you want to add a control character yourself in a changes entry, to quote some char, that is (using level or encapsulation chars is not recommended since `\changes` uses them itself), use rather `\quotechar`.

Before writing an entry to the `.glo` file, `\changes` checks if the date (the second mandatory = the third argument) is later than the date stored in the counter `ChangesStartDate`. You may set this counter with a

```
\ChangesStart{<version>}{<year>/<month>/<day>}
```

declaration.

If the `ChangesStartDate` is set to a date contemporary to T_EX i.e., not earlier than September 1982⁷, then a note shall appear at the beginning of the changes history that informs the reader of omitting the earlier changes entries.

If the date stored in `ChangesStartDate` is earlier than T_EX, no notification of omitting shall be printed. This is intended for a rather tricky usage of the changes start date feature: you may establish two threads of the changes history: the one for the users, dated with four digit year, and the other for yourself only, dated with two or three digit year. If you declare

```
\ChangesStart{<version?>}{1000/00/00}
```

⁷ DEK in *T_EX The Program* mentions that month as of T_EX Version 0 release.

or so, the changes entries dated with less-than-four digit year shall be omitted and no notification shall be issued of that.

\CheckSum While scanning the CSes in the code, gmdoc counts them and prints the information about their number on the terminal and in .log. Moreover, you may declare \CheckSum {*number*} before the code and TeX will inform you whether the number stated by you is correct or not, and what it is. As you guess, it's not my original idea but I took it from doc.

There it is provided as a tool for testing whether the file is corrupted. My TeX Guru says it's a bit old-fashioned nowadays but I like the idea and use it to document the file's growth. For this purpose gmdoc types out lines like

```
% \chschange{vo.98j}{2006/10/19}{4372}
% \chschange{vo.98j}{06/10/19}{4372}
```

and you may place them at the beginning of the source file. Such a line results in setting the check sum to the number contained in the last pair of braces and in making a 'general' changes entry that states the check sum for version *first brace* dated *second brace* was *third brace*.

\toCTAN There is also \toCTAN {*date*}\{*version*} , a shorthand for
\chgs {*date*}\{*version*}\put\to\acro{CTAN}\on\{*date*}\}

The parameters

The gmdoc package provides some parameters specific to typesetting the TeX code:

\stanzaskip is a vertical space inserted when a blank (code) line is met. It's equal \medskipamount by default. Subsequent blank code lines do not increase this space.

At the points where narration begins a new line after the code or an in-line comment and where a new code line begins after the narration (that is not an in-line comment), a \CodeTopsep glue is added. At the beginning and the end of a macro or environment environment a \MacroTopsep glue is added. By default, these two skips are set equal \stanzaskip.

The \stanzaskip's value is assigned also to the display skips and to \topsep. This is done with the \UniformSkips declaration executed by default. If you want to change some of those values, you should declare \NonUniformSkips in the preamble to discard the default declaration. (To be more precise, by default \UniformSkips is executed twice: when loading gmdoc and again \AtBeginDocument to allow you to change \stanzaskip and have the other glues set due to it. \NonUniformSkips relaxes the \UniformSkips's occurrence at \begin{document}.)

If you want to add a vertical space of \CodeTopsep (equal by default \stanzaskip), you are provided the \stanza command. Similarly, if you want to add a vertical space of the \MacroTopsep amount (by default also equal \stanzaskip), you are given the \chunkskip command. They both act analogously to \addvspace i.e., don't add two consecutive glues but put the bigger of them.

Since \CodeTopsep glue is inserted automatically at each transition from the code (or code with an in-line comment) to the narration and reverse, it may happen that you want not to add such a glue exceptionally. Then there's the \nostanza command. You can use it before narration to remove the vskip before it or after narration to suppress the vskip after it.

\CodeIndent The TeX code is indented with the \CodeIndent glue and a leading space increases indentation of the line by its (space's) width. The default value of \CodeIndent is 1.5em.

\TextIndent There's also a parameter for the indent of the narration, \TextIndent, but you

should use it only in emergency (otherwise what would be the margins for?). It's optional by default.

By default, the end of a `\DocInput` file is marked with

□'

`\EOFMark` given by the `\EOFMark` macro.

If you do use the ε - \TeX 's primitive `\everyeof`, be sure the contents of it begins with `\relax` because it's the token that stops the main macro scanning the code.

The crucial concept of `gmdoc` is to use the line end character as a verbatim group opener and the comment char, usually the `%`, as its delimiter. Therefore the 'knowledge' what char starts a commentary is for this package crucial and utterly important. The default assumption is that you use `%` as we all do. So, if you use another character, then you should declare it with `\CodeDelim` typing the desired char preceded by a backslash, e.g., `\CodeDelim\&`. (As just mentioned implicitly, `\CodeDelim\%` is declared by default.)

This declaration is always global so whenever and wherever you change your mind you should express it with a new `\CodeDelim` declaration.

The unstarred version of `\CodeDelim` changes also the verb 'hyphen', the char appearing at the verbatim line breaks that is and affects the `\narrationmark` which by default typesets `%` followed by an en space.

The starred version, `\CodeDelim*`, changes only the code delimiter and the char typeset remains untouched. Most probably you shouldn't use the starred version.

Talking of special chars, the escape char, `\` by default, is also very important for this package as it marks control sequences and allows automatic indexing them for instance. Therefore, if you for any reason choose another than `\` character to be the escape char, you should tell `gmdoc` about it with the `\CodeEscapeChar` declaration. As the previous one, this too takes its argument preceded by a backslash, e.g., `\CodeEscapeChar\!`. (As you may deduct from the above, `\CodeEscapeChar\\` is declared by default.)

The tradition is that in the packages `@` char is a letter i.e., of catcode `11`. Frank Mittelbach in `doc` takes into account a possibility that a user wishes some other chars to be letters, too, and therefore he (F.M.) provides the `\MakePrivateLetters` macro. So do I and like in `doc`, this macro makes `@` sign a letter. It also makes `*` a letter in order to cover the starred versions of commands.

Analogously but for a slightly different purpose, the `\AddtoPrivateOthers` macro is provided here. It adds its argument, which is supposed to be a one-char CS, to the `\doprivateothers` list, whose rôle is to allow some special chars to appear in the marking commands' arguments (the commands described in section Macros for marking the macros). The default contents of this list is `\space` (the space) and `\^` so you may mark the environments names and special sequences like `\^A` safely. This list is also extended with every char that is `\MakeShortVerbed`. (I don't see a need of removing chars from this list, but if you do, please let me know.)

The line numbers (if enabled) are typeset in the `\LineNumFont` declaration's scope, which is defined as `{\normalfont\tiny}` by default. Let us also remember, that for each counter there is a `\the<counter>` macro available. The counter for the line numbers is called `codelinenum` so the macro printing it is `\thecodelinenum`. By default we don't change its \TeX 's definition which is equivalent to `\arabic{codelinenum}`.

Three more parameter macros, are `\IndexPrefix`, `\EntryPrefix` and `\HLPrefix`. All three are provided with the account of including multiple files in one document. They are equal (almost) `\empty` by default. The first may store main level index entry of which all indexed macros and environments would be sub-entries, e.g., the name of the package. The third may or even should store a text to distinguish equal code-line numbers of distinct source files. It may be the file name too, of course. The second

macro is intended for another concept, namely the one from `\txdoc` class, to distinguish the codeline numbers from different files *in the index* by the file marker. Anyway, if you document just one file per document, there's no need of redefining those macros, nor when you input multiple files with `\DocInclude`.

`gmdoc` automatically indexes the control sequences occurring in the code. Their index entries may be 'common' or distinguished in two (more) ways. The concept is to distinguish the entries indicating the *usage* of the CS and the entries indicating the *definition* of the CS.

The special formattings of 'usage' and 'def' index entries are determined by `\UsgEntry` and `\DefEntry` one-parameter macros (the parameter shall be substituted with the reference number) and by default are defined as `\textit` and `\underline` respectively (as in `doc`).

`\CommonEntryCmd`

There's one more parameter macro, `\CommonEntryCmd` that stores the name of the encapsulation for the 'common' index entries (not special) i.e., a word that'll become a CS that will be put before an entry in the `.ind` file. By default it's defined as `{relax}` and a nontrivial use of it you may see in the source of chapter 1928, where `\def\CommonEntryCmd{\UsgEntry}` makes all the index entries of the driver formatted as 'usage'.

The index comes in a `multicols` environment whose columns number is determined by the `\IndexColumns` counter set by default to 3. To save space, the index begins at the same page as the previous text provided there is at least `\IndexMin` of the page height free. By default, `\IndexMin = 133`.opt.

`\IndexPrologue`

The text put at the beginning of the index is declared with a one-argument `\IndIndexPrologue`. Its default text at current index option you may `admire` on page 263. Of course, you may write your own `\IndexPrologue {<brand new index prologue>}`, but if you like the default and want only to add something to it, you are provided `\AtDIPPrologue` one-argument declaration that adds the stuff after the default text. For instance, I used it to add a label and hypertarget that is referred to two sentences earlier.

By default the colour of the index entry hyperlinks is set black to let Adobe Reader work faster. If you don't want this, `\let\IndexLinksBlack\relax`. That leaves the index links colour alone and hides the text about black links from the default index prologue.

Other index parameters are set with the `\IndexParms` macro defined in line 5926 of the code. If you want to change some of them, you don't have to use `\renewcommand*{\IndexParms}` and set all of the parameters: you may `\gaddtomacro{\IndexParms}{% <only the desired changes>}`. (`\gaddtomacro` is an alias for L^AT_EX's `\g@addto@macro` provided by `gmutils`.)

At the default `gmdoc` settings the `.idx` file is prepared for the default settings of `MakeIndex` (no special style). Therefore the index control chars are as usual. But if you need to use other chars as `MakeIndex` controls, know that they are stored in the four macros: `\actualchar`, `\quotechar`, `\levelchar` and `\encapchar` whose meaning you infer from their names. Any redefinition of them *should be done in the preamble* because the first usage of them takes place at `\begin{document}` and on it depends further tests telling T_EX what characters of a scanned CS name it should quote before writing it to the `.idx` file.

Frank Mittelbach in `doc` provides the `\verbatimchar` macro to (re)define the `\verb`'s delimiter for the index entries of the scanned CS names etc. `gmdoc` also uses `\verbatimchar` but defines it as `{&}`. Moreover, a macro that wraps a CS name in `\verb` checks whether the wrapped CS isn't `\&` and if it is, `$` is taken as the delimiter. So there's hardly chance that you'll need to redefine `\verbatimchar`.

So strange delimiters are chosen deliberately to allow any 'other' chars in the environments names.

```
\StopEventually
  \Finale
\AlsoImplementation
\OnlyDescription
```

There's a quadratus of commands taken from doc: `\StopEventually`, `\Finale`, `\AlsoImplementation` and `\OnlyDescription` that should be explained simultaneously (in a polyphonic song e.g.).

The `\OnlyDescription` and `\AlsoImplementation` declarations are intended to exclude or include the code part from the documentation. The point between the description and the implementation part should be marked with `\StopEventually {<the stuff to be executed anyway>}` and `\Finale` should be typed at the end of file. Then `\On|lyDescription` defines `\StopEventually` to expand to its argument followed by `\endinput` and `\AlsoImplementation` defines `\StopEventually` to do nothing but pass its argument to `\Finale`.

The narration macros

```
\verb
```

To print the control sequences' names you have the `\verb` macro and its 'shortverb' version whatever you define (see the `gmverb` package).

```
\inverb
```

For short verbatim texts in the in-line comments `gmdoc` provides the `\inverb<a char>...<a char>` (the name stands for 'in-line verbatim') command that redefines the `gmverb` breakables to break with `%` at the beginning of the lower line to avoid mistaking such a broken verbatim commentary text for the code.

```
\cs
```

But nor `\verb[*`] neither `\inverb` will work if you put them in an argument of another macro. For such a situation, or if you just prefer, `gmdoc` (`gmutils`) provides a robust command `\cs`, which takes one obligatory argument, the macro's name without the backslash, e.g., `\cs{mymacro}` produces `\mymacro`. I take account of a need of printing some other text verbatim, too, and therefore `\cs` has the first argument optional, which is the text to be typeset before the mandatory argument. It's the backslash by default, but if you wish to typeset something without the `\`, you may write `\cs[] % not \ a~macro`. Moreover, for typesetting the environments' names, `gmdoc` (`gmutils`) provides the `\env` macro, that prints its argument verbatim and without a backslash, e.g., `\env{an environment}` produces `an environment`.

```
\env
```

For usage in the in-line comments there are `\incs` and `\inenv` commands that take analogous arguments and precede the typeset command and environment names with a `%` if at the beginning of a new line. To those who like `\cmd`, there is also `\incmd`, an in-line version of the former.

```
\nlperc
```

```
\+
```

And for line breaking at `\cs` and `\env` there is `\nlperc` to ensure `%` at the beginning of a new line and `\+` to use in `\cs` and `\env` argument for a discretionary hyphen that'll break to `-` at the end of the upper line and `%` at the beginning of the lower line. By default hyphenation of `\cs` and `\env` arguments is off, you can allow it only at `\-` or `\+.`

```
\nlpcent
```

There is also `\nlpcent` if you wish a discretionary `%` without `\incs` or `\inverb`.

By default the multi-line in-line comments are typeset with a hanging indent (that is constant relatively to the current indent of the code) and justified. Since vertical alignment is determined by the parameters as they are at the moment of `\par`, no one can set the code line to be typeset ragged right (to break nicely if it's long) and the following in-line comment to be justified. Moreover, because of the hanging indent the lines of multi-line in-line comments are relatively short, you may get lots of overfulls. Therefore there is a Boolean switch `ilrr` (OCSR), whose name stands for 'In-Line Ragged-Right' and the in-line comments (and their codelines) are typeset justified in the scope of `\ilrrfalse` which is the default. When you write `\ilrrtrue`, then all in-line comments in its scope (and their codelines) will be typeset ragged right (and still with the hanging indent). Moreover, you are provided `\ilrr` and `\ilju` commands that set `\ilrrtrue` and `\ilrrfalse` for the current in-line comment only. Note you can use them anywhere within such a comment, as they set `\rightskip` basically. `\ilrr` and `\ilju` are no-ops in the stand-alone narration.

```
\pk  
\file  
\catletter  
\catother  
\catactive  
\division  
\subdivision  
\subsubdivision
```

To print packages' names sans serif there is a `\pk` one-argument command, and the `\file` command intended for the filenames.

Because we play a lot with the `\catcodes` here and want to talk about it, there are `\catletter`, `\catother` and `\catactive` macros that print $_{11}$, $_{12}$ and $_{13}$ respectively to concisely mark the most used char categories.

I wish my self-documenting code to be able to be typeset each package separately or several in one document. Therefore I need some 'flexible' sectioning commands and here they are: `\division`, `\subdivision` and `\subsubdivision` so far, that by default are `\let` to be `\section`, `\subsection` and `\subsubsection` respectively.

One more kind of flexibility is to allow using mwcls or the standard classes for the same file. There was a trouble with the number and order of the optional arguments of the original mwcls's sectioning commands.

It's resolved in gutils so you are free at this point, and even more free than in the standard classes: if you give a sectioning command just one optional argument, it will be the title to toc and to the running head (that's standard in scs⁸). If you give *two* optionals, the first will go to the running head and the other to toc. (In both cases the mandatory argument goes only to the page).

If you wish the `\DocIncluded` files make other sectionings than the default, you may declare `\SetFileDiv {<sec name without backslash>}`.

```
\gmlonely  
\skipgmlonely
```

`gmdoc.sty` provides also an environment `gmlonely` to wrap some text you think you may want to skip some day. When that day comes, you write `\skipgmlonely` before the instances of `gmlonely` you want to skip. This declaration has an optional argument which is for a text that'll appear in(stead of) the first `gmlonely`'s instance in every `\DocInput` or `\DocIncluded` file within `\skipgmlonely`'s scope.

An example of use you may see in this documentation: the repeated passages about the installation and compiling the documentation are skipped in further chapters thanks to it.

`gmdoc` (gutils, to be precise) provides some TEX-related logos:

```
\AmSTeX  
\BibTeX  
\SLiTTeX  
\PlainTeX  
\Web  
\TeXbook  
\TB  
\eTeX  
\pdfTeX  
\pdfTeX  
\XeTeX  
\LaTeXpar  
\ds
```

typesets $\mathcal{AM}\mathcal{S}$ -TEX,
BEB-TEX,
SLITTEX,
PLAIN TEX,
WEB,
The TEXbook,
The TEXbook
 ε -TEX,
pdf ε -TEX
pdfTEX
 \mathbb{X} TEX (the first E will be reversed if the graphics package is loaded or \mathbb{X} TEX is at work)
and
(LA)TEX.
DocStrip not quite a logo, but still convenient.

```
\copyrnote
```

The `copyrnote` environment is provided to format the copyright note flush left in `\obeylines`' scope.

```
\gmdmarginpar
```

To put an arbitrary text into a marginpar and have it flushed right just like the macros' names, you are provided the `\gmdmarginpar` macro that takes one mandatory argument which is the contents of the marginpar.

⁸ See gutils for some subtle details.

To make a vertical space to separate some piece of text you are given two macros: `\stanza` and `\chunkskip`. The first adds `\stanzaskip` while the latter `\Macro{Topsep}`. Both of them take care of not cumulating the vspace.

`quotation`

The quotation environment is redefined just to enclose its contents in double quotes.

If you don't like it, just call `\RestoreEnvironment{quotation}` after loading `gmdoc`. Note however that other environments using `quotation`, such as `abstract`, keep their shape.

`\GetFileInfo`
 `\filedate`
`\fileversion`
 `\fileinfo`

The `\GetFileInfo{<file name with extension>}` command defines `\filedate`, `\fileversion` and `\fileinfo` as the respective pieces of the info (the optional argument) provided by `\ProvidesClass`/`\Package`/`\File` declarations. The information of the file you process with `gmdoc` is provided (and therefore getable) if the file is also loaded (or the `\Provide...` line occurs in a `\StraightEOL` scope).

`\ProvideFileInfo`

If the input file doesn't contain `\Provides...` in the code layer, there are commands `\ProvideFileInfo{<file name with extension>} [<info>]`. (`<info>` should consist of: `<year>/<month>/<day> ↴ <version number> ↴ <a short note>`.)

`\FileInfo`

Since we may documentally input files that we don't load, `doc` in `gmdoc` e.g., we provide a declaration to be put (in the comment layer) before the line(s) containing `\Provides...`. The `\FileInfo` command takes the subsequent stuff till the closing `]` and subsequent line end, extracts from it the info and writes it to the `.aux` and rescans the stuff. We use an ε -`\TeX` primitive `\scantokens` for that purpose.

`\filenote`

A macro for the standard note is provided, `\filenote`, that expands to "This file has version number `<version number>` dated `<date>`." To place such a note in the document's title (or heading, with `\DocInclude` at the default settings), there's `\thfileinfo` macro that puts `\fileinfo` in `\thanks`.

`\gmdnoindent`

Since `\noindent` didn't want to cooperate with my code and narration layers sometimes, I provide `\gmdnoindent` that forces a not indented paragraph if `\noindent` could not.

`\CDPerc`

If you declare the code delimiter other than `%` and then want `%` back, you may write `\CDPerc` instead of `\CodeDelim* \%`.

`\CDAnd`

If you like `&` as the code delimiter (as I did twice), you may write `\CDAnd` instead of `\CodeDelim \&`.

`\CS`

To get 'CS' which is 'CS' in small caps (in `\acro` to be precise), you can write `\CS`. This macro is `\protected` so you can use it safely in `\changes` e.g. Moreover, it checks whether the next token is a letter and puts a space if so so you don't have to bother about `\CS \u`.

`enumargs`

`\mand`

`\opt`

`enumargs*`

To enumerate the list of command's arguments or macro's parameters there is the `enumargs` environment which is a version of `enumerate` with labels like `#7`. You can use `\item` or, at your option, `\mand` which is just an alias for the former. For an optional arguments use `\opt` which wraps the item label in square brackets. Moreover, to align optional and mandatory arguments digit under digit, use the `enumargs*` environment.

Both environments take an optional argument which is the number of `#`s. It's `1` by default, but also can be `2` or `4` (other numbers will typeset numbers without a `#`). Please feel free to notify me if you really need more hashes in that environment.

For an example driver file see chapter [The driver](#).

A queerness of `\label`

You should be loyally informed that `\label` in `gmdoc` behaves slightly non-standard in the `\DocInput`/ `\Included` files: the automatic redefinitions of `\ref` at each code line

are *global* (since the code is typeset in groups and the `\refs` will be out of those groups), so a `\reference` in the narrative will point at the last code line not the last section, *unlike* in the standard L^AT_EX.

doc-compatibility

One of my goals while writing gmdoc was to make compilation of doc-like files with gmdoc possible. I cannot guarantee the goal has been reached but I *did* compile doc.dtx with not a smallest change of that file (actually, there was a tiny little buggie in line 3299 which I fixed remotely with `\AfterMacrocode` tool written specially for that). So, if you wish to compile a doc-like file with my humble package, just try.

`\AfterMacrocode {<mc number>} {<the stuff>}` defines control sequence `\gmd@mchook<mc number>` with the meaning `<the stuff>` which is put at the end of `macrocode` and `oldmc` number `<mc number>` (after the group).

The doc commands most important in my opinion are supported by gmdoc. Some commands, mostly the obsolete in my opinion, are not supported but give an info on the terminal and in .log.

I assume that if one wishes to use doc's interface then they won't use gmdoc's options but just the default. (Some gmdoc options may interfere with some doc commands, they may cancel them e.g.)

The main input commands compatible with doc are `\OldDocInput` and `\DocIn`, `\DocInclude`, the latter however only in the `\olddocIncludes` declaration's scope.

Within their scope/argument the `macrocode` environments behave as in doc, i.e. they are a kind of verbatim and require to be ended with `%\end{macrocode}[*]`.

The default behaviour of `macrocode` [*] with the 'new' input commands is different however. Remember that in the 'new' fashion the code and narration layers philosophy is in force and that is sustained within `macrocode` [*]. Which means basically that with 'new' settings when you write

```
% \begin{macrocode}
  \alittlemacro % change it to \blaargh
% \end{macrocode}
```

and `\blaargh`'s definition is `{foo}`, you'll get

```
\alittlemacro % change it to foo
```

(Note that 'my' `macrocode` doesn't require the magical `%\end{macrocode}`.)

If you are used to the traditional (doc's) `macrocode` and still wish to use gmdoc new way, you have at least two options: there is the `oldmc` environment analogous to the traditional (doc's) `macrocode` (it also has the starred version), that's the first option (I needed the traditional behaviour once in this documentation, find out where & why). The other is to write `\OldMacros`. That declaration (OCSR) redefines `macrocode` and `macrocode*` to behave the traditional way. (It's always executed by `\OldDocInput` and `\olddocIncludes`.)

For a more detailed discussion of what is doc-compatible and how, see the code section [doc-compatibility](#).

1928 <package>

The driver part

In case of a single package, such as gmuilts, a driver part of the package may look as follows and you put it before `\ProvidesPackage`/`Class`.

```

% \skiplines we skip the driver
\ifnum\catcode` \@=12

\documentclass[outeroff, pagella, fontspec=quiet]{gmdocc}
\usepackage{eufrak}% for |\continuum| in the commentary.
\twocoltoc
\begin{document}

\DocInput{\jobname.sty}
\PrintChanges
\thispagestyle{empty}
\typeout{%
  Produce change log with^^J%
  makeindex -r -s gmglo.ist -o \jobname.gls \jobname.glo^^J
  (gmglo.ist should be put into some texmf/makeindex
  directory.)^^J}
\typeout{%
  Produce index with^^J%
  makeindex -r \jobname^^J}
\afterfi{\end{document} }

\fi% of driver pass
%\endskiplines

```

\skiplines
\endskiplines The advantage of \skiplines... \endskiplines over \iffalse... \fi is that the latter has to contain balanced \ifs and \fis while the former hasn't because it sanitises the stuff. More precisely, it uses the \dospecials list, so it sanitises also the braces.

Moreover, when the countalllines[*] option is in force, \skipfiles... \end{skipfiles} keeps the score of skipped lines.

Note %\iffalse ... %\fi in the code layer that protects the driver against being typeset.

But gmdoc is more baroque and we want to see the driver typeset—behold.

```

1979 \ifnum\catcode` \@=12
1981 \errorcontextlines=100
1984 \documentclass[countalllines, codespacesgrey, outeroff,
  debug, mwrep,
1985 pagella, trebuchet, cursor, fontspec=quiet]{gmdocc}
1987 \verb!LongDashes!
1989 \DoNotIndex{\gmu@tempa \gmu@tempb \gmu@tempc \gmu@tempd %
  \gmu@tempc \gmu@tempf}

1991 \twocoltoc
1992 \title{The \pk{gmdoc} Package \i.e., \pk{gmdoc.sty} and
1993   \pk{gmdocc.cls}}
1994 \author{Grzegorz 'Natrór' Murzynowski}
1995 \date{\ifcase\month\relax\or January\or February\or March%
  \or April\or May\or
1996 June\or July\or August\or September\or October\or%
  November\or
1997 December\fi\the\year}

% \includeonly{gmoldcomm}

2001 \begin{document}

```

```

2007 \maketitle
2009 \setcounter{page}{2}%
2011 hyperref cries if it sees two pages numbered 1.
2012 \tableofcontents
2012 \DoIndex\maketitle
2015 \SelfInclude
2017 \DocInclude{gmdocc}

```

For your convenience I decided to add the documentations of the three auxiliary packages:

```

2021 \skipgmlonely[\stanza The remarks about installation and
2022   compiling
2022   of the documentation are analogous to those in the
2022   chapter
2023   \pk{gmdoc.sty} and therefore omitted.\stanza]
2024 \DocInclude{gmutils}
2025 \DocInclude{gmiflink}
2026 \DocInclude{gmverb}
2028 \DocInclude{gmoldcomm}
2029 \typeout{%
2030   Produce change log with^^J%
2031   makeindex -r -s gmglo.ist -o \jobname.gls \jobname.glo^^J
2032   (gmglo.ist should be put into some texmf/makeindex
2032   directory.)^^J}
2033 \PrintChanges
2034 \typeout{%
2035   Produce index with^^J%
2036   makeindex -r \jobname^^J}
2037 \PrintIndex
2039 \afterfi{%
2040 \end{document}

```

MakeIndex shell commands:

```

2042 makeindex -r gmdoc
2043 makeindex -r -s gmglo.ist -o gmdocDoc.gls gmdocDoc.glo
(gmglo.ist should be put into some texmf/makeindex directory.)

```

And “That’s all, folks” ;).

2050 } \fi% of \ifnum\catcode`\\@=12, of the driver that is.

The code

For debug

```

2060 \catcode`\\^C=9\relax

```

We set the \catcode of this char to ₁₃ in the comment layer.

The basic idea of this package is to re\catcode ¹⁴M (the line end char) and % (or any other comment char) so that they start and finish typesetting of what’s between them as the TeX code i.e., verbatim and with the bells and whistles.

The bells and whistles are (optional) numbering of the codelines, and automatic indexing the CSes, possibly with special format for the ‘def’ and ‘usage’ entries.

As mentioned in the preface, this package aims at a minimal markup of the working code. A package author writes their splendid code and adds a brilliant comment in %ed lines and that's all. Of course, if they wants to make a \section or \emphasise, they has to type respective CSes.

I see the feature described above to be quite a convenience, however it has some price. See section [Life among queer EOLs](#) for details, here I state only that in my opinion the price is not very high.

More detailedly, the idea is to make ^^M (end of line char) active and to define it to check if the next char i.e., the beginning of the next line is a % and if so to gobble it and just continue usual typesetting or else to start a verbatim scope. In fact, every such a line end starts a verbatim scope which is immediately closed, if the next line begins with (leading spaces and) the code delimiter.

Further details are typographical parameters of verbatim scope and how to restore normal settings after such a scope so that a code line could be commented and still displayed, how to deal with leading spaces, how to allow breaking a moving argument in two lines in the comment layer, how to index and marginpar macros etc.

The package options

```
2109 \RequirePackage{gmutils}[2008/08/30]%
  includes redefinition of \newif
  to make the switches \protected.
2111 \RequirePackage{xkeyval}%
  we need key-vals later, but maybe we'll make the
  option key-val as well.
```

Maybe someone wants the code lines not to be numbered.

```
\if@linesnotnum 2117 \newif\if@linesnotnum
linesnotnum 2119 \DeclareOption{linesnotnum}{\@linesnotnumtrue}
```

And maybe he or she wishes to declare resetting the line counter along with some sectioning counter him/herself.

```
\if@uresetlinecount 2124 \newif\if@uresetlinecount
uresetlinecount 2126 \DeclareOption{uresetlinecount}{\@uresetlinecounttrue}
```

And let the user be given a possibility to count the comment lines.

```
\if@countalllines 2131 \newif\if@countalllines
\if@printalllinenos 2132 \newif\if@printalllinenos
countalllines 2134 \DeclareOption{countalllines}{%
  to use the \inputlineno primitive and
  print real line numbers in a file.
  \@countalllinestrue
  \@printalllinenosfalse}
```

```
countalllines* 2139 \DeclareOption{countalllines*}{%
  \@countalllinestrue
  \@printalllinenostrue}
```

Unlike in doc, indexing the macros is the default and the default reference is the code line number.

```
\if@noindex 2147 \newif\if@noindex
noindex 2149 \DeclareOption{noindex}{\@noindextrue}
\if@pageindex 2152 \newif\if@pageindex
```

```
pageindex 2154 \DeclareOption{pageindex}{\@pageindextrue}
```

It would be a great honour to me if someone would like to document L^AT_EX source with this humble package but I don't think it's really probable so let's make an option that'll switch index exclude list properly (see sec. [Index exclude list](#)).

```
\if@indexallmacros 2161 \newif\if@indexallmacros
```

```
indexallmacros 2163 \DeclareOption{indexallmacros}{\@indexallmacrostrue}
```

Some document classes don't support marginpars or disable them by default (as my favourite Marcin Woliński's classes).

```
\if@marginparsused 2173 \@ifundefined{if@marginparsused}{\newif\if@marginparsused}{}{}
```

This switch is copied from mwbk.cls for compatibility with it. Thanks to it loading an mwcls with [withmarginpar] option shall switch marginpars on in this package, too.

To be compatible with the standard classes, let's \let:

```
2180 \@ifclassloaded{article}{\@marginparsusedtrue}{}{}
```

```
2183 \@ifclassloaded{report}{\@marginparsusedtrue}{}{}
```

```
2185 \@ifclassloaded{book}{\@marginparsusedtrue}{}{}
```

And if you don't use mwcls nor standard classes, then you have the options:

```
withmarginpar 2188 \DeclareOption{withmarginpar}{\@marginparsusedtrue}
```

```
nomarginpar 2190 \DeclareOption{nomarginpar}{\@marginparsusedfalse}
```

The order of the above conditional switches and options is significant. Thanks to it the options are available also in the standard classes and in mwcls.

To make the code spaces blank (they are visible by default except the leading ones).

```
codespacesblank 2200 \DeclareOption{codespacesblank}{%
 2201   \AtEndOfPackage{\% to allow codespacesgrey, \codespacesblank}
 2202   \AtBeginDocument{\CodeSpacesBlank}}{}
```

```
codespacesgrey 2205 \DeclareOption{codespacesgrey}{%
 2208   \AtEndOfPackage{\% to put the declaration into the begin-document hook after
 2209     definition of \visiblespace.
 2210   \AtBeginDocument{\CodeSpacesGrey}}}{}
 2212 \ProcessOptions
```

The dependencies and preliminaries

We require another package of mine that provides some tricky macros analogous to the L^AT_EX standard ones, such as \newgif and \ifnextcat. Since 2008/08/08 it also makes \if... switches \protected (redefines \newif)

```
2221 \RequirePackage{gmuutils}[2008/08/08]
```

A standard package for defining colours,

```
2224 \RequirePackage{xcolor}
```

and a colour definition for the hyperlinks not to be too bright

```
2226 \definecolor{deepblue}{rgb}{0,0,.85}
```

And the standard package probably most important for gmdoc: If the user doesn't load hyperref with their favourite options, we do, with *ours*. If they has done it, we change only the links' colour.

```

2239 \@ifpackageloaded{hyperref}{\hypersetup{colorlinks=true,
2240     linkcolor=deepblue, urlcolor=blue, filecolor=blue}}{%
2241     \RequirePackage[colorlinks=true, linkcolor=deepblue,
2242         urlcolor=blue,
2243         filecolor=blue, pdfstartview=FitH, pdfview=FitBH,
2244         pdfpagemode=UseNone]{hyperref}}

```

Now a little addition to `hyperref`, a conditional hyperlinking possibility with the `\gmhypertarget` and `\gmiflink` macros. It *has* to be loaded *after* `hyperref`.

```
2253 \RequirePackage{gmiflink}
```

And a slight redefinition of `verbatim`, `\verb[*]` and providing of `\MakeShortVerb`.

```

2256 \RequirePackage{gmverb}[2008/08/20]
2258 \StoreMacros{\@verbatim\verb}
2260 \if@noindex
2261     \AtBeginDocument{\gag@index}%
    for the latter macro see line 5169.
2263 \else
2264     \RequirePackage{makeidx}\makeindex
2265 \fi

```

Now, a crucial statement about the code delimiter in the input file. Providing a special declaration for the assignment is intended for documenting the packages that play with %'s `\catcode`. Some macros for such plays are defined [further](#).

The declaration comes in the starred and unstarred version. The unstarred version besides declaring the code delimiter declares the same char as the verb(atim) 'hyphen'. The starred version doesn't change the verb 'hyphen'. That is intended for the special tricks e.g. for the `oldmc` environment.

If you want to change the verb 'hyphen', there is the `\VerbHyphen` [`\<one char>`](#) declaration provided by `gmverb`.

```

\CodeDelim 2296 \def\CodeDelim{\@bsphack\gm@ifstar\Code@Delim@St\Code@Delim}
\Code@Delim@St 2298 \def\Code@Delim@St#1{%
2299     {\escapechar\m@ne
2300      \def\@xa\gdef\@xa\code@delim\@xa{\string#1}}%
2301     \@esphack}
    (\@xa is \expandafter, see gmutils.)
```

```
2304 \def\Code@Delim#1{\VerbHyphen{#1}\Code@Delim@St{#1}}
```

It is an invariant of `gmdocing` that `\code@delim` stores the current code delimiter (of catcode 12).

The `\code@delim` should be 12 so a space is not allowed as a code delimiter. I don't think it *really* to be a limitation.

And let's assume you do as we all do:

```
2313 \CodeDelim\%
```

And to typeset this code delimiter pretty, let's `\def`:

```

\narrationmark 2316 \pdef\narrationmark{{\tt\verbhyphen}{\normalfont\enspace}}%
    \ignorespaces

```

We'll play with `\everypar`, a bit, and if you use such things as the `{itemize}` environment, an error would occur if we didn't store the previous value of `\everypar`

and didn't restore it at return to the narration. So let's assign a `\toks` list to store the original `\everypar`:

```
\gmd@preverypar 2327 \newtoks \gmd@preverypar
\settexcodehangi 2329 \newcommand* \settexcodehangi {%
 2330   \hangindent=\verbatimhangindent \hangafter=\@ne}%
  we'll use it in
  the in-line comment case. \verbatimhangindent is provided by the
  gmverb package and = 3em by default.
2334 \@ifdefinable \@@settexcodehangi {\let \@@settexcodehangi=%
  \settexcodehangi}
```

We'll play a bit with `\leftskip`, so let the user have a parameter instead. For normal text (i.e. the comment):

```
\TextIndent 2340 \newlength \TextIndent
```

I assume it's originally equal to `\leftskip`, i.e. `\z@`. And for the TeX code:

```
\CodeIndent 2344 \newlength \CodeIndent
2347 \CodeIndent=1,5em\relax
```

And the vertical space to be inserted where there are blank lines in the source code:

```
2350 \@ifundefined \stanzaskip {\newlength \stanzaskip} {}
```

I use `\stanzaskip` in `gmverse` package and derivatives for typesetting poetry. A computer program code *is* poetry.

```
\stanzaskip 2355 \stanzaskip=\medskipamount
```

A vertical space between the commentary and the code seems to enhance readability so declare

```
2362 \newskip \CodeTopsep
2363 \newskip \MacroTopsep
```

And let's set them. For aesthetic minimality⁹ let's unify them and the other most important vertical spaces used in `gmdoc`. I think a macro that gathers all these assignments may be handy.

```
\UniformSkips 2379 \def \UniformSkips {%
  \CodeTopsep=\stanzaskip
  \MacroTopsep=\stanzaskip
  \abovedisplayskip=\stanzaskip
  %% \abovedisplayshortskip remains untouched as it is 0.0pt plus 3.0pt by default.
  \belowdisplayskip=\stanzaskip
  \belowdisplayshortskip=.5\stanzaskip% due to DEK's idea of making
  the short below display skip half of the normal.
  \advance \belowdisplayshortskip by \smallskipamount
  \advance \belowdisplayshortskip by -1\smallskipamount% We advance
  % \belowdisplayshortskip forth and back to give it the \small %
  % skipamount's shrink- and stretchability components.
  \topsep=\stanzaskip
  \partopsep=\z@
```

⁹ The terms 'minimal' and 'minimalist' used in `gmdoc` are among others inspired by the *South Park* cartoon's episode *Mr. Hankey The Christmas* (...) in which 'Philip Glass, a Minimalist New York composer' appears in a 'non-denominational non-offensive Christmas play' ;). (Philip Glass composed the music to the *Qatsi* trilogy among others).

2398 }

We make it the default,

2400 \UniformSkips

but we allow you to change the benchmark glue i.e., `\stanzaskip` in the preamble and still have the other glues set due to it: we launch `\UniformSkips` again after the preamble.

2405 \AtBeginDocument {\UniformSkips}

So, if you don't want them at all i.e., you don't want to set other glues due to `\stanzaskip`, you should use the following declaration. That shall discard the unwanted setting already placed in the `\begin{document}` hook.

\NonUniformSkips 2412 \newcommand*{\NonUniformSkips}{\relax\enlargethispage{\UniformSkips}}

Why do we launch `\UniformSkips` twice then? The first time is to set all the gmdoc-specific glues *somewhat*, which allows you to set not all of them, and the second time to set them due to a possible change of `\stanzaskip`.

And let's define a macro to insert a space for a chunk of documentation, e.g., to mark the beginning of new macro's explanation and code.

\chunkskip 2422 \newcommand*{\chunkskip}{%
2423 \par\addvspace{
2424 \glueexpr\MacroTopsep
2425 \if@codeskipput-\CodeTopsep\fi
2426 \relax
2427 } \atcodeskipputgtrue}

And, for a smaller part of text,

\stanza 2430 \pdef{\stanza}{%
2431 \par\addvspace{
2432 \glueexpr\stanzaskip
2433 \if@codeskipput-\CodeTopsep\fi
2434 \relax} \atcodeskipputgtrue}

Since the stanza skips are inserted automatically most often (cf. lines [2891](#), [3318](#), [2911](#), [3199](#), [3371](#)), sometimes you may need to forbid them.

\nostanza 2439 \newcommand*{\nostanza}{%
2440 \par
2441 \if@codeskipput\unless\if@nostanza\vskip-\CodeTopsep\relax%
2442 \fi\fi
2443 \atcodeskipputgtrue\atnostanzagtrue
2444 \atafternarrgfalse\ataftercodegtrue} In the 'code to narration' case the
first switch is enough but in the counter-case 'narration to code' both the
second and third are necessary while the first is not.

To count the lines where they have begun not before them

2451 \newgif\if@newline

`\newgif` is `\newif` with a global effect i.e., it defines `\...gtrue` and `\...gfalse` switchers that switch respective Boolean switch *globally*. See `gutils` package for details.

To handle the DocStrip directives not *any* %<....

\if@dsdir 2459 \newgif\if@dsdir

This switch will be falsified at the first char of a code line. (We need a switch independent of the one indicating whether the line has or has not been counted because of two reasons: 1. line numbering is optional, 2. counting the line falsifies that switch *before* the first char.)

The core

Now we define main \inputting command that'll change catcodes. The macros used by it are defined later.

```

2474 \begingroup\catcode`\\^M=\active%
2475 \firstofone{\endgroup%
\DocInput 2476 \newcommand*\{\DocInput}[1]{\begingroup%
2479     \edef\gmd@inputname{\#1}% we'll use it in some notifications.
2481     \let\gmd@currentlabel@before=\@currentlabel% we store it because
          we'll do \xdefs of \@currentlabel to make proper references to the
          line numbers so we want to restore current \@currentlabel after our
          group.
2486     \gmd@setclubpenalty% we wrapped the assignment of \clubpenalty in
          a macro because we'll repeat it twice more.
2488     \@clubpenalty\clubpenalty\widowpenalty=3333% Most paragraphs
          of the code will be one-line most probably and many of the narration, too.

2493     \tolerance=1000% as in doc.
2496     \@xa\@makeother\csname\code@delim\endcsname%
2498     \gmd@resetlinecount% due to the option uresetlinecount we reset the
          line number counter or do nothing.
^M 2501     \QueerEOL% It has to be before the begin-input-hook to allow change by that
          hook.
2506     \@beginputhook% my first use of it is to redefine \maketitle just at this
          point not globally.
2508     \everypar=\@xa\@xa\codetonarrskip\the\everypar}%
2510     \edef\gmd@guardedinput{%
2511         \@nx\@input\#1\relax% \@nx is \noexpand, see gutils. \@in%
          put is the true TeX's \input.
2515     \gmd@ihook% cf. line 7340
2516     \@nx\EOFMark% to pretty finish the input, see line 2718.
2518     \@nx\CodeDelim\@xa\@nx\csname\code@delim\endcsname% to en-
          sure the code delimiter is the same as at the beginning of input.
2523     \@nx^M\code@delim%
2525     }% we add guardians after \inputting a file; somehow an error occurred with-
          out them.
2527     \catcode`\\=9% for doc-compatibility.
2528     \setcounter{CheckSum}{0}% we initialise the counter for the number of
          the escape chars (the assignment is \global).
2530     \everyeof{\relax}% \@nx moved not to spoil input of toc e.g.
2531     \@xa\@xa\@xa^M\gmd@guardedinput%
2532     \par%
2534     \@endinputhook% It's a hook to let postpone some stuff till the end of input.
          We use it e.g. for the doc-(not)likeness notifications.
2537     \glet\@currentlabel=\gmd@currentlabel@before% we restore value
          from before this group. In a very special case this could cause unex-
```

pected behaviour of cross-refs, but anyway we acted globally and so acts hyperref.

```
2541 \endgroup%
2542 }% end of \Doc@Input's definition.
2543 }% end of \firstofone's argument.
```

So, having the main macro outlined, let's fill in the details.

First, define the queer EOL. We define a macro that $\wedge\wedge M$ will be let to. $\backslash gmd@textEOL$ will be used also for checking the $\wedge\wedge M$ case ($\backslash @ifnextchar$ does $\backslash ifx$).

```
\gmd@textEOL 2553 \pdef \gmd@textEOL {\ % a space just like in normal TeX. We put it first to co-
                     operate with  $\wedge\wedge M$ 's  $\backslash expandafter$   $\backslash ignorespaces$ . It's no problem since
                     a space  $\sqcup_{10}$  doesn't drive TeX out of the vmode.
2557 \ifhmode \c@afternarrtrue \c@codeskipputfalse \fi% being in the hor-
                     izontal mode means we've just typeset some narration so we turn the re-
                     spective switches: the one bringing the message 'we are after narration' to
                     True ( $\c@afternarr$ ) and the 'we have put the code-narration glue' to False
                     ( $\c@codeskipput$ ). Since we are in a verbatim group and the information
                     should be brought outside it, we switch the switches globally (the letter g
                     in both).
2564 \c@newline=true% to \refstep the lines' counter at the proper point.
2566 \c@dsdir=true% to handle the DocStrip directives.
2567 \c@xa \c@trimandstore \the \everypar \c@trimandstore% we store the pre-
                     vious value of \everypar register to restore it at a proper point. See line
                     3407 for the details.
2570 \begingroup%
2576 \gmd@setclubpenalty% Most paragraphs will be one-line most probably. Since
                     some sectioning commands may change \clubpenalty, we set it again
                     here and also after this group.
2580 \aftergroup \gmd@setclubpenalty%
2581 \let \par \c@par% inside the verbatim group we wish \par to be genuine.
2583 \ttverbatim% it does \tt and makes specials other or \active-and-breakable.
                     to turn verbatim specials off in \scanverbs.
2587 \gmd@DoTeXCodeSpace%
2588 \c@makeother \c@% because \ttverbatim doesn't do that.
2589 \MakePrivateLetters% see line 3681.
2590 \c@xa \c@makeother \code@delim% we are almost sure the code comment char
                     is among the chars having been  $\sqcup_2$ ed already. For 'almost' see the \Index|
                     Input macro's definition.
```

So, we've opened a verbatim group and want to peek at the next character. If it's %, then we just continue narration, else we process the leading spaces supposed there are any and, if after them is a %, we just continue the commentary as in the previous case or else we typeset the TeX code.

```
2599 \texcode@hook% we add some special stuff, e.g. in gmdoc.cls we make star low.
2601 \c@xa \c@ifnextchar \c@xa { \code@delim } { %
2603   \gmd@continuenarration } %
2604   \gmd@dolspaces% it will launch \gmd@typesettexcode.
2605 }% end of \c@ifnextchar's else.
2606 }% end of \gmd@textEOL's definition.
2609 \emptify \texcode@hook
```

```
\gmd@setclubpenalty 2611 \def \gmd@setclubpenalty { \clubpenalty=3333 \ }
```

For convenient adding things to the begin- and endinput hooks:

```

\AtEndInput 2615 \def \AtEndInput {\g@addto@macro \@endinputhook}
\@endinputhook 2616 \def \@endinputhook {}

```

Simili modo

```

\AtBeginInput 2619 \def \AtBeginInput {\g@addto@macro \@begininputhook}
\@begininputhook 2620 \def \@begininputhook {}

```

For the index input hooking now declare a macro, we define it another way at line [7340](#).

```
2624 \emptyify \gmd@iihook
```

And let's use it instantly to avoid a disaster while reading in the table of contents.

```

\tableofcontents 2629 \AtBeginInput {\let \gmd@toc \tableofcontents
2630   \def \tableofcontents {%
2631     \@ifQueerEOL {\StraightEOL \gmd@toc \QueerEOL}%
2632     {\gmd@toc}{}}

```

As you'll learn from lines [3503](#) and [3490](#), we use those two strange declarations to change and restore the very special meaning of the line end. Without such changes `\tableofcontents` would cause a disaster (it did indeed). And to check the catcode of `^M` is the rôle of `\@ifEOLactive`:

```

\@ifEOLactive 2644 \def \@ifEOLactive {%
  % #1 what if end of line is active,
  % #2 what if not.
2649   \ifnum \catcode `\\^M=\active \xa \@firstoftwo \else \xa %
  % @secondoftwo \fi}

2651 \foone \obeylines {%
\@ifQueerEOL 2652   \def \@ifQueerEOL {%
  % #1 what if line end is 'queer',
  % #2 what if not 'queer'.
2658   \@ifEOLactive {%
2659     \ifx \\^M \gmd@textEOL \xa \@firstoftwo \else \xa %
  % @secondoftwo \fi}%
2660   {\@secondoftwo} }% of \@ifQueerEOL
2661 }% of \foone

```

A footnote for the 'queer' line ends scope.

```

\qfootnote 2664 \pdef \qfootnote {%
2665   \@ifQueerEOL
2666   {\begingroup \StraightEOL \qfootnote@}%
2667   {\footnote}}

```

```

\qfootnote@ 2669 \DeclareCommand \qfootnote@ {o>Lm} {%
2670   \endgroup % yes, we close the group: the arguments are already parsed and
  % passed to this macro.
2672   \edef \gmu@tempa {%
2673     \nx \footnote \IfValueT {\#1} {[#1]} }%
2674   \gmu@tempa {\#2}%
2675 }

```

An emphasis command for 'queer' line ends.

```

\qemph 2678 \pdef \qemph {%
2679   \@ifQueerEOL
2680   {\begingroup \StraightEOL \qemph@}%

```

```

2681 { \emph } }
\qemph@ \pdef \qemph@#1 { \endgroup \emph { #1 } }

```

The declaration below is useful if you wish to put sth. just in the nearest input/included file and no else: at the moment of putting the stuff it will erase it from the hook. You may declare several \AtBeginInputOnes, they add up.

```

\gmd@ABIOnce 2695 \@emptyify \gmd@ABIOnce
2696 \AtEndOfPackage { \AtBeginInput \gmd@ABIOnce }

\AtBeginInputOnce 2700 \long \def \AtBeginInputOnce#1 {%
2713   \gaddtomacro \gmd@ABIOnce { \g@emptyify \gmd@ABIOnce#1 } }

```

Many tries of finishing the input cleanly led me to setting the guardians as in line [2523](#) and to

```
\EOFMark 2718 \def \EOFMark { \<eof> }
```

Other solutions did print the last code delimiter or would require managing a special case for the macros typesetting TeX code to suppress the last line's numbering etc.

If you don't like it, see line [8312](#).

Due to the `codespacesblank` option in the line ?? we launch the macro defined below to change the meaning of a gmdoc-kernel macro.

```

\begin{obeyspaces}%
\gdef \CodeSpacesVisible {%
\def \gmd@DoTeXCodeSpace {%
\obeyspaces \let \ = \breakablevisspace } }%
\gdef \CodeSpacesBlank {%
\let \gmd@DoTeXCodeSpace \gmobeyspaces%
\let \gmd@texcodespace = \ }% the latter \let is for the \if...s.
\gdef \CodeSpacesSmall {%
\def \gmd@DoTeXCodeSpace {%
\obeyspaces \def \ { \, \hskip \z@ } }%
\def \gmd@texcodespace { \, \hskip \z@ } }%
\end{obeyspaces}
\gdef \CodeSpacesGrey {%
\CodeSpacesVisible
\VisSpacesGrey% defined in gmverb
}%

```

Note that \CodeSpacesVisible doesn't revert \CodeSpacesGrey.

```
2762 \CodeSpacesVisible
```

How the continuing of the narration should look like?

```

\def \gmd@continuenarration {%
\endgroup
\gmd@cpnarrline% see below.
\@xa \trimandstore \the \everypar \trimandstore
\everypar = \@xa { \@xa \codetonarrskip \the \everypar }%
\@xa \gmd@checkifEOL \gobble}

```

Simple, isn't it? (We gobble the 'other' code delimiter. Despite of \egroup it's [12](#) because it was touched by \futurelet contained in \ifnextchar in line [2601](#). And

in line 3019 it's been read as $_z$. That's why it works in spite of that % is of category 'ignored').

2778 \if@countalllines

If the countalllines option is in force, we get the count of lines from the \in|putlineno primitive. But if the option is countalllines*, we want to print the line number.

```
\gmd@countnarrline@ 2788 \def\gmd@countnarrline@{%
 2799 \gmd@grefstep{codelinenum}@\newlinefalse
 2800 \everypar=\@xa{%
 2801   \@xa\@codetonarrskip\the\gmd@preverypar}%
 2802   % el@line macro puts a hypertarget in a \raise i.e., drives TEX
 2803   % into the horizontal mode so \everypar shall be issued. Therefore
 2804   % we should restore it.
 2805 }% of \gmd@countnarrline@
\gmd@grefstep 2798 \def\gmd@grefstep#1{%
 2806   instead of diligent redefining all possible com-
 2807   mands and environments we just assign the current value of the respec-
 2808   tive TEX's primitive to the codelinenum counter. Note we decrease it
 2809   by -1 to get the proper value for the next line. (Well, I don't quite know
 2810   why, but it works.)
 2811   \ifnum\value{#1}<\inputlineno
 2812     \csname_c@#1\endcsname\numexpr\inputlineno-1\relax
 2813     \ifvmode\leavevmode\fi% this line is added 2008/08/10 after an
 2814     % all-night debugging;) that showed that at one point \gmd@grefstep
 2815     % was called in vmode which caused adding \penalty 10000 to
 2816     % the main vertical list and thus forbidding page break during entire
 2817     % oldmc.
 2818     \grefstepcounter{#1}%
 2819   \fi% We wrap stepping the counter in an \ifnum to avoid repetition of
 2820   % the same ref-value (what would result in the "multiply defined la-
 2821   % bels" warning).
```

The \grefstepcounter macro, defined in gmverb, is a global version of \ref|stepcounter, observing the redefinition made to \refstepcounter by hyperref.

2824 \if@printalllinenos% Note that checking this switch makes only sense
 2825 when countalllines is true.

```
\gmd@cpnarrline 2826 \def\gmd@cpnarrline{%
 2827   count and print narration line
 2828   \if@newline
 2829     \gmd@countnarrline@
 2830     \hyperlabel@line
 2831     {\LineNumFont\thecodelinenum},\ignorespaces}%
 2832   \fi}
 2833 \else% not printalllinenos
 2834   \emptyify\gmd@cpnarrline
 2835 \fi
```

```
\gmd@ctallsetup 2836 \def\gmd@ctallsetup{%
 2837   In the oldmc environments and with the \FileInfo
 2838   declaration (when countalllines option is in force) the code is
 2839   gobbled as an argument of a macro and then processed at one place
 2840   (at the end of oldmc e.g.) so if we used \inputlineno, we would
 2841   have got all the lines with the same number. But we only set the
 2842   counter not \refstep it to avoid putting a hypertarget.
```

```

2843 \setcounter{codelinenum}{\inputlineno}%
2844   it's global.  

2845 \let\gmd@grefstep\hhrefstepcounter}  

2846 \else% not countalllines (and therefore we won't print the narration lines'  

2847   numbers either)  

2848 \emptyify\gmd@cpnarrline  

2849 \let\gmd@grefstep\hhrefstepcounter% if we don't want to count all the  

2850   lines, we only \ref-increase the counter in the code layer.  

2851 \emptyify\gmd@ctallsetup  

2852 \fi% of \if@countalllines  

\skipelines 2853 \def\skipelines{\bgroup  

2854   \let\do\@makeother\dospecials% not \@sanitize because the latter  

2855     doesn't recatcode braces and we want all to be quieten.  

2856   \gmd@skipelines}  

2857 \edef\gmu@tempa{%
2858   \long\def\@nx\gmd@skipelines##1\bslash\endskipelines{%
2859     \egroup}}  

2860 \gmu@tempa

```

And typesetting the \TeX code?

```

\gmd@typesettexcode 2861 \foone\obeylines{%
2862   \def\gmd@typesettexcode{%
2863     \gmd@parfixclosingspace% it's to eat a space closing the paragraph, see
2864       below. It contains \par.

```

A verbatim group has already been opened by \ttverbatim and additional \catcode code.

```

2865 \everypar={\@settexcodehangi}%
2866   At first attempt we thought of giving
2867     the user a \toks list to insert at the beginning of every code line, but
2868     what for?  

2869 ^M 2869 \def^M{%
2870   \TeX code EOL
2871   @newlinetrue% to \refstep the counter in proper place.
2872   @dsdirgtrue% to handle the DocStrip directives.
2873   \global\gmd@closingspacewd=\z@% we don't wish to eat a closing
2874     space after a codeline, because there isn't any and a negative rigid
2875     \hskip added to \parfillskip would produce a blank line.
2876   \ifhmode\par\@codeskipputfalse\else%
2877     \if@codeskipput%
2878     \else\addvspace{\stanzaskip}\@codeskipputtrue%
2879     \fi% if we've just met a blank (code) line, we insert a \stanzaskip
2880       glue.
2881   \fi%
2882   \prevhmodefalse% we want to know later that now we are in the
2883     vmode.
2884   \@ifnextchar{\gmd@texcodespace}{%
2885     \@dsdirgfalse\gmd@dolspaces\{\gmd@charbychar\}%
2886   }% end of ^M's definition.
2887   \let\gmd@texcodeEOL=^M% for further checks inside \gmd@charbychar.
2888   \raggedright\leftskip=\CodeIndent%
2889   \if@aftercode%
2890     \gmd@nocodeskip1{iaC}%
2891   \else%

```

```

2907 \if@afternarr%
2908   \if@codeskipput \else%
2909     \gmd@codeskip1\@aftercodegfalse%
2910   \fi%
2911 \else\gmd@nocodeskip1 {NaN}%
2912   \fi%
2913 \fi% if now we are switching from the narration into the code, we insert
2914   a proper vertical space.
2915 \@aftercodegtrue\@afternarrgfalse%
2916 \ifdim\gmd@ldspaceswd>\z@% and here the leading spaces.
2917   \leavevmode\@dsdirgfalse%
2918   \if@newline\gmd@grefstep{codelinenum}\@newlinefalse%
2919   \fi%
2920   \printlinenumber% if we don't want the lines to be numbered, the re-
2921     spective option \lets this CS to \relax.
2922 \hyperlabel@line%
2923 \mark@envir% index and/or marginize an environment if there is some
2924   to be done so, see line 5043.
2925 \hskip\gmd@ldspaceswd%
2926 \advance\hangindent \by\gmd@ldspaceswd%
2927 \xdef\settexcodehangi{%
2928   \nx\hangindent=\the\hangindent% and also set the hanging in-
2929     dent setting for the same line comment case. BTW., this % or rather
2930     lack of it costed me five hours of debugging and rewriting. Active
2931     line ends require extreme caution.
2932   \nx\hangafter=1\space}%
2933 \else%
2934   \glet\settexcodehangi=\@@settexcodehangi%
2935   % \printlinenumber here produced line numbers for blank lines
2936     which is what we don't want.
2937 \fi% of \ifdim
2938 \gmd@ldspaceswd=\z@%
2939 \prevhmodegfalse% we have done \par so we are not in the hmode.
2940 \@aftercodegtrue% we want to know later that now we are typesetting
2941   a codeline.
2942 \if@ilgroup\aftergroup\egroup\@ilgroupfalse\fi% when we are
2943   in the in-line comment group (for ragged right or justified), we want to
2944   close it. But if we did it here, we would close the verbatim group for the
2945   code. But we set the switch false not to repeat \aftergroup\egroup.
2946 \gmd@charbychar% we'll eat the code char by char to scan all the macros and
2947   thus to deal properly with the case \% in which the % will be scanned and
2948   won't launch closing of the verbatim group.
2949 }% of \gmd@typesettexcode.
2950 }% of \foone\obeylines.

```

Now let's deal with the leading spaces once forever. We wish not to typeset \llcorner s but to add the width of every leading space to the paragraph's indent and to the hanging indent, but only if there'll be any code character not being % in this line (e.g., the end of line). If there'll be only %, we want just to continue the comment or start a new one. (We don't have to worry about whether we should \par or not.)

```

2951 \gmd@spacewd \newlength\gmd@spacewd% to store the width of a (leading) \llcorner12.
2952 \gmd@ldspaceswd \newlength\gmd@ldspaceswd% to store total length of gobbled leading spaces.

```

It costed me some time to reach that in my verbatim scope a space isn't ₁₂ but ₁₃, namely \let to \breakablevisspace. So let us \let for future:

```
\gmd@texcodespace 2983 \let \gmd@texcodespace=\breakablevisspace
```

And now let's try to deal with those spaces.

```
\gmd@dolspaces 2986 \def \gmd@dolspaces {%
2987   \ifx \gmd@texcodespace \let@token
2988     \@dsdirgfalse
2989     \afterfi{\settowidth{\gmd@spacewd}{\visible}{%}
2990       \gmd@ldspaceswd=\z@%
2991       \gmd@eatlspce}{%
2992     \else \afterfi{%
2993       about this smart macro and other of its family see gutils
2994       sec. 3.
2995       \if@afternarr \if@aftercode
2996         \ifilrr\bgroup\gmd@setilrr\fi
2997         \fi\fi
2998       \par% possibly after narration
2999       \if@afternarr \if@aftercode
3000         \ifilrr\egroup\fi
3001       \fi\fi
3002       \gmd@typesettextcode}{%
3003     \fi}
```

And now, the iterating inner macro that'll eat the leading spaces.

```
\gmd@eatlspce 3010 \def \gmd@eatlspce#1{%
3011   \ifx \gmd@texcodespace#1%
3012     \advance \gmd@ldspaceswd by \gmd@spacewd% we don't \advance
3013     it \globally because the current group may be closed iff we meet %
3014     and then we'll won't indent the line anyway.
3015     \afterfifi \gmd@eatlspce
3016   \else
3017     \if \code@delim \nx#1%
3018       \gmd@ldspaceswd=\z@
3019       \afterfifi{\gmd@continuenarration\narrationmark}{%
3020       \else \afterfifi{\gmd@typesettextcode#1}{%
3021         \fi
3022       \fi}{%
3023     \fi}{%
```

We want to know whether we were in hmode before reading current \code@delim. We'll need to switch the switch globally.

```
3028 \newgif \ifprevhmode
```

And the main iterating inner macro which eats every single char of verbatim text to check the end. The case \% should be excluded and it is indeed.

```
\gmd@charbychar 3036 \newcommand* \gmd@charbychar[1] {%
3037   \ifhmode \prevhmodegttrue
3038   \else \prevhmodegtfalse
3039   \fi
3040   \if \code@delim \nx#1%
3041     \def \next {%
3042       occurs when next a \hskip4.875pt is to be put
3043       \gmd@percenthack% to typeset % if a comment continues the codeline.
3044     \endgroup%
```

```

3047   \gmd@checkifEOLmixd}%
3048   \else% i.e., we've not met the code delimiter
3049     \ifx\relax\def\next{%
3050       \endgroup}%
3051       special case of end of file thanks to \everyeof.
3052     \else
3053       \if\code@escape@char\@nx#1%
3054         \@dsdirgfalse%
3055         yes, just here not before the whole \if because then
3056         we would discard checking for DocStrip directives doable by the
3057         active % at the 'old macrocode' setting.
3058         \def\next{%
3059           \gmd@counttheline#\@scantmacro}%
3060         \else
3061           \def\next{%
3062             \gmd@EOLorcharbychar#1}%
3063             \fi
3064           \fi
3065         \fi\next}
3066

\debug@special 3068 \def\debug@special#1{%
3069   \ifhmode\special{color\push\gray\o.\#1}%
3070   \else\special{color\push\gray\o.\#1000}\fi}

```

One more inner macro because $\wedge M$ in TeX code wants to peek at the next char and possibly launch `\gmd@charbychar`. We deal with counting the lines thoroughly. Increasing the counter is divided into cases and it's very low level in one case because `\refstepcounter` and `\stepcounter` added some stuff that caused blank lines, at least with hyperref package loaded.

```

\gmd@EOLorcharbychar 3078 \def\gmd@EOLorcharbychar#1{%
3079   \ifx\gmd@texcodeEOL#1%
3080     \if@newline
3081       \newlinefalse
3082     \fi
3083     \afterfi{#1}%
3084     here we print #1.
3085   \else% i.e., #1 is not a (very active) line end,
3086     \afterfi
3087     {%
3088   \gmd@counttheline#\gmd@charbychar}%
3089   or here we print #1. Here we would
3090   also possibly mark an environment but there's no need of it because declaring
3091   an environment to be marked requires a bit of commentary and here we are
3092   after a code  $\wedge M$  with no commentary.
3093   \fi}

\gmd@counttheline 3094 \def\gmd@counttheline{%
3095   \ifvmode
3096     \if@newline
3097       \leavevmode
3098       \gmd@grefstep{codelinenumber}\newlinefalse
3099       \hyperlabel@line
3100     \fi
3101     \printlinenumber
3102     \mark@envir
3103   \else% not vmode
3104     \if@newline
3105       \gmd@grefstep{codelinenumber}\newlinefalse

```

```

3114      \hyperlabel@line
3115      \fi
3116  \fi}

```

If before reading current % char we were in horizontal mode, then we wish to print % (or another code delimiter).

```

\gmd@percenthack 3121 \def\gmd@percenthack {%
3122   \ifprevhmode\aftergroup\narrationmark% We add a space after %, be-
      cause I think it looks better. It's done \aftergroup to make the spaces
      possible after the % not to be typeset.
3128   \else\aftergroup\gmd@narrcheckifds@ne% remember that \gmd@pre |
      centhack is only called when we've the code delimiter and soon we'll close
      the verbatim group and right after \endgroup there waits \gmd@checkifEOLmixd.
3132   \fi}

\gmd@narrcheckifds@ne 3134 \def\gmd@narrcheckifds@ne#1{%
3135   \@dsdirgfalse\@ifnextchar<{%
3136     \@xa\gmd@docstripdirective\@gobble}{#1}}

```

The macro below is used to look for the %^^M case to make a commented blank line make a new paragraph. Long searched and very simple at last.

```

\gmd@checkifEOL 3142 \def\gmd@checkifEOL{%
3143   \gmd@cpnarrline
3144   \everypar=\@xa{\@xa\@codetonarrskip% we add the macro that'll insert
      a vertical space if we leave the code and enter the narration.
3147   \the\gmd@preverypar}%
3148   \@ifnextchar{\gmd@textEOL}{%
3150     \@dsdirgfalse
3151     \par\ignorespaces}%
3152   \gmd@narrcheckifds}%

```

We check if it's %<, a DocStrip directive that is.

```

\gmd@narrcheckifds 3155 \def\gmd@narrcheckifds{%
3156   \@dsdirgfalse\@ifnextchar<{%
3157     \@xa\gmd@docstripdirective\@gobble}{\ignorespaces}}

```

In the 'mixed' line case it should be a bit more complex, though. On the other hand, there's no need to checking for DocStrip directives.

```

\gmd@checkifEOLmixd 3163 \def\gmd@checkifEOLmixd{%
3164   \gmd@cpnarrline
3165   \everypar=\@xa{\@xa\@codetonarrskip\the\gmd@preverypar}%
3168   \@afternarrgfalse\@aftercodegtrue
3169   \ifhmode\@codeskipputgfalse\fi
3170   \@ifnextchar{\gmd@textEOL}{%
3172     \raggedright\gmd@endpe\par}%
      without \raggedright this \par would
      be justified which is not appropriate for a long codeline that should be
      broken, e.g., 3165.
3176   \prevhmodegfalse
3177   \gmd@endpe\ignorespaces}%

```

If a codeline ends with % (prevhmode == True) first \gmd@endpe sets the parameters at the TeX code values and \par closes a paragraph and the latter \gmd@endpe sets the parameters at the narration values. In the other case both \gmd@endpes do the same and \par between them does nothing.

```

\par 3185 \def\par{%
3186   \ifhmode% (I added this \ifhmode as a result of a heavy debug.)
3187     \if@afternarr\if@aftercode
3188       \unless\if@ilgroup\bgroup\@ilgrouptrue\fi
3189       \ifilrr\gmd@setilrr\fi
3190     \fi\fi
3191   \@@par
3192   \if@afternarr
3193     \if@aftercode
3194       \if@ilgroup\egroup\fi% if we are both after code and after nar-
3195         ration it means we are after an in-line comment. Then we prob-
3196         ably end a group opened in line 3238
3197       \if@codeskipput\else\gmd@codeskip2%
3198         \@aftercodegfalse\fi
3199       \else\gmd@nocodeskip2{naC}%
3200         \fi
3201       \else\gmd@nocodeskip2{naN}%
3202         \fi
3203       \prevhmodegfalse\gmd@endpe% when taken out of \ifhmode, this
3204         line caused some codeline numbers were typeset with \leftskip =
3205         0.
3206       \everypar=\@xa{%
3207         \@xa\@codetonarrskip\the\gmd@preverypar}%
3208       \let\par\@@par%
3209     \fi}%
3210   \gmd@endpe\ignorespaces} }

```

As we announced, we play with \leftskip inside the verbatim group and therefore we wish to restore normal \leftskip when back to normal text i.e. the commentary. But, if normal text starts in the same line as the code, then we still wish to indent such a line.

```

\gmd@endpe 3219 \def\gmd@endpe{%
3220   \ifprevhmode
3221     \settexcodehangi% ndent
3222     \leftskip=\CodeIndent
3223   \else
3224     \leftskip=\TextIndent
3225     \hangindent=\z@
3226     \everypar=\@xa{%
3227       \@xa\@codetonarrskip\the\gmd@preverypar}%
3228     \fi}

```

Now a special treatment for an in-line comment:

```

\ifilrr 3234 \newif\ifilrr
\ilrr 3236 \def\ilrr{%
3237   \if@aftercode
3238     \unless\if@ilgroup\bgroup\@ilgrouptrue\fi% If we are 'aftercode',
3239       then we are in an in-line comment. Then we open a group to be able to
3240       declare e.g. \raggedright for that comment only. This group is closed
3241       in line 3195 or 2948.
3242     \ilrrtrue
3243   \fi}

```

```

\if@ilgroup 3246 \newif\if@ilgroup
\gmd@setilrr 3248 \def\gmd@setilrr{\rightskipoptplus\textwidth}
\ilju 3250 \def\ilju{%
  when in-line comments are ragged right in general but we want just
  this one to be justified.
  3252   \if@aftercode
  3253     \unless\if@ilgroup\bgroup\@ilgrouptrue\fi
  3254     \ilrrfalse
  3255   \fi}

\verbcodecorr 3257 \def\verbcodecorr{%
  a correction of vertical spaces between a \verb and
  code. We put also a \par to allow parindent in the next commentary.
  3261   \vskip-\lastskip\vskip-4\CodeTopsep\vskip3\CodeTopsep\par}

```

Numbering (or not) of the lines

Maybe you want codelines to be numbered and maybe you want to reset the counter within sections.

```

3269 \if@uresetlinecount% with uresetlinecount option...
3270   \relax\gmd@resetlinecount% ... we turn resetting the counter by \Doc {
  % Input off...
\resetlinecountwith 3272 \newcommand*\resetlinecountwith[1]{%
  codelinenum 3273   \newcounter{codelinenum}[#1]%
  ... and provide a new declaration of
  the counter.
3275 \else% With the option turned off...
  DocInputsCount 3276   \newcounter{DocInputsCount}%
  codelinenum 3277   \newcounter{codelinenum}[DocInputsCount]%
  ... we declare the \DocIn-
  puts' number counter and the codeline counter to be reset with stepping
  of it.
\gmd@resetlinecount 3283   \newcommand*\gmd@resetlinecount{\stepcounter{DocInputsCount}}%
  ... and let the \DocInput increment the \DocInputs number count and thus
  reset the codeline count. It's for unique naming of the hyperref labels.
3287 \fi

```

Let's define printing the line number as we did in gmvb package.

```

\printlinenumber 3291 \newcommand*\printlinenumber{%
  3292   \leavevmode\llap{\rlap{\LineNumFont\phantom{999}}\llap{%
    \thecodelinenum}}%
  3293   \hskip\leftskip}}
\LineNumFont 3295 \def\LineNumFont{\normalfont\tiny}
3297 \if@linesnotnum\relax\printlinenumber\fi
\hyperlabel@line 3299 \newcommand*\hyperlabel@line{%
  3300   \if@pageindex% It's good to be able to switch it any time not just define it once
    according to the value of the switch set by the option.
  3303   \else
  3304     \raisebox{2ex}{\scriptsize\z@\{}{\gmhypertarget{clnum.%}
  3305       \HLPrefix\arabic{codelinenum}\{\}}\%
  3306   \fi}

```

Spacing with \everypar

Last but not least, let's define the macro inserting a vertical space between the code and the narration. Its parameter is a relic of a very heavy debug of the automatic vspacing mechanism. Let it remain at least until this package is 2.0 version.

```
\gmd@codeskip 3316 \newcommand*{\gmd@codeskip}[1] {%
 3317   \@@par\addvspace\CodeTopsep
 3318   \codeskipputgtrue\@nostanzagfalse}
```

Sometimes we add the `\CodeTopsep` vertical space in `\everypar`. When this happens, first we remove the `\parindent` empty box, but this doesn't reverse putting `\parskip` to the main vertical list. And if `\parskip` is put, `\addvspace` shall see it not the 'true' last skip. Therefore we need a Boolean switch to keep the knowledge of putting similar vskip before `\parskip`.

```
\if@codeskipput 3329 \newgif\if@codeskipput
```

A switch to control `\nostanzas`:

```
3332 \newgif\if@nostanza
```

The below is another relic of the heavy debug of the automatic vspacing. Let's give it the same removal clause as [above](#).

```
\gmd@nocodeskip 3337 \newcommand*{\gmd@nocodeskip}[2] {}
```

And here is how the two relic macros looked like during the debug. As you see, they are disabled by a false `\if` (look at it closely ;-).

```
\gmd@codeskip 3342 \if1\if1
 3343   \renewcommand*{\gmd@codeskip}[1] {%
 3344     \hbox{\rule{1cm}{3pt}\#1!!!!}}
\gmd@nocodeskip 3345   \renewcommand*{\gmd@nocodeskip}[2] {%
 3346     \hbox{\rule{1cm}{0.5pt}\#1:\#2\#2}}
3347 \fi
```

We'll wish to execute `\gmd@codeskip` wherever a codeline (possibly with an inline comment) is followed by a homogeneous comment line or reverse. Let us dedicate a Boolean switch to this then.

```
\if@aftercode 3353 \newgif\if@aftercode
```

This switch will be set true in the moments when we are able to switch from the `\TeX` code into the narration and the below one when we are able to switch reversely.

```
\if@afternarr 3358 \newgif\if@afternarr
```

To insert vertical glue between the `\TeX` code and the narration we'll be playing with `\everypar`. More precisely, we'll add a macro that the `\parindent` box shall move and the glue shall put.

```
\@codetonarrskip 3363 \def\@codetonarrskip{%
 3364   \if@codeskipput\else
 3365     \if@afternarr\gmd@nocodeskip4{iaN}\else
 3366       \if@aftercode
```

We are at the beginning of `\everypar`, i.e., `\TeX` has just entered the hmode and put the `\parindent` box. Let's remove it then.

```
3369           {\setboxo=\lastbox}\%
```

Now we can put the vertical space and state we are not 'aftercode'.

```

3371           \gmd@codeskip4%
3373           \else\gmd@nocodeskip4{naC}%
3374           \fi
3375           \fi
3376           \fi
3377           \leftskip\TextIndent% this line is a patch against a bug-or-feature that in
            certain cases the narration \leftskip is left equal the code leftskip. (It
            happens when there are subsequent code lines after an in-line comment
            not ended with an explicit \par.) Before v0.99n it was just after line 3371.
3382           \@aftercodegfalse \@nostanzagtrue
3384 }

```

But we play with `\everypar` for other reasons too, and while restoring it, we don't want to add the `\@codetonarrskip` macro infinitely many times. So let us define a macro that'll check if `\everypar` begins with `\@codetonarrskip` and trim it if so. We'll use this macro with proper `\expandafter`ing in order to give it the contents of `\everypar`. The work should be done in two steps first of which will be checking whether `\everypar` is nonempty (we can't have two delimited parameters for a macro: if we define a two-parameter macro, the first is undelimited so it has to be nonempty; it costed me some one hour to understand it).

```

\@trimandstore
\@trimandstore@hash
3396 \long\def\@trimandstore#1\@trimandstore{%
3397   \def\@trimandstore@hash{\#1}%
3398   \ifx\@trimandstore@hash\@empty% we check if #1 is nonempty. The \if%
            % \relax#1\relax trick is not recommended here because using it we
            couldn't avoid expanding #1 if it'd be expandable.
3402   \gmd@preeverypar={ }%
3403   \else
3404     \afterfi{\@xa\@trimandstore@ne\the\everypar%
            \@trimandstore}%
3405   \fi}
3407 \long\def\@trimandstore@ne#1#2\@trimandstore{%
3408   \def\trimmed@everypar{\#2}%
3409   \ifx\@codetonarrskip{\#1}%
3410     \gmd@preeverypar=\@xa{\trimmed@everypar}%
3411   \else
3412     \gmd@preeverypar=\@xa{\the\everypar}%
3413   \fi}

```

We prefer not to repeat `#1` and `#2` within the `\ifs` and we even define an auxiliary macro because `\everypar` may contain some `\ifs` or `\fis`.

Life among queer EOLs

When I showed this package to my TeX Guru he commended it and immediately pointed some disadvantages in the comparison with the `doc` package.

One of them was an expected difficulty of breaking a moving argument (e.g., of a sectioning macro) in two lines. To work it around let's define a line-end eater:

```

3428 \catcode`^\^B=\active% note we re\catcode <char2> globally, for the entire
            document.
3430 \foone{\obeylines}%
3431 { \def\QueerCharTwo{%
3432   \protected\def^\^B##1^\^M{%

```

```

3434     \ifhmode\unskip\space\ignorespaces\fi} }% It shouldn't be \
3435         not
3436     }
3438 \QueerCharTwo
3440 \AtBeginInput {\@ifEOlactive {\catcode`\\=active} {} \QueerCharTwo}%
3441     We repeat redefinition of char2 at begin of the documenting input, be-
3442     cause doc.dtx suggests that some packages (namely inputenc) may re\cat-
3443     code such unusual characters.

```

As you see the `\\=active` char is defined to gobble everything since itself till the end of line and the very end of line. This is intended for harmless continuing a line. The price is affecting the line numbering when `countalllines` option is enabled.

I also liked the doc's idea of comment² i.e., the possibility of marking some text so that it doesn't appear nor in the working version neither in the documentation, got by making `^A` (i.e., *char1*) a comment char.

However, in this package such a trick would work another way: here the line ends are active, a comment char would disable them and that would cause disasters. So let's do it an `\active` way.

```

3462 \catcode`\\=active% note we re\catcode char1 globally, for the entire
3463     document.

```

```

3464 ^A \foone\obeylines{%
3465     \def\QueerCharOne{%
3466         \def^A{%
3467             \bgroup\let\do\@makeother\dospecials%
3468                 \gmd@gobbleuntilM} }%
3469     \def\gmd@gobbleuntilM#1^M{\egroup\ignorespaces^M}%
3470 }
3471 \QueerCharOne
3472 \AtBeginInput {\@ifEOlactive {\catcode`\\=active} {} \QueerCharOne}%
3473     see note after line
3440.

```

As I suggested in the users' guide, `\StraightEOL` and `\QueerEOL` are intended to cooperate in harmony for the user's good. They take care not only of redefining the line end but also these little things related to it.

One usefulness of `\StraightEOL` is allowing line-breaking of the command arguments. Another—making possible executing some code lines during the documentation pass.

```

\StraightEOL 3490 \def\StraightEOL{%
3491     \catcode`\\=5
3492     \catcode`\\=14
3493     \catcode`\\=14
3494     \def\\{\\}
3495
3496 \foone\obeylines{%
3497     \def\QueerEOL{%
3498         \catcode`\\=active%
3499         \let\\ \gmd@textEOL%
3500         \catcode`\\=active%
3501         \catcode`\\=active% I only re\catcode char1 and char2 hoping
3502             no one but me is that perverse to make them \active and (re)define.
3503             (Let me know if I'm wrong at this point.)}
3504
3505
3506
3507
3508
3509
3510
3511
3512
3513
3514
3515
3516
3517
3518
3519
3520
3521
3522
3523
3524
3525
3526
3527
3528
3529
3530
3531
3532
3533
3534
3535
3536
3537
3538
3539
3540
3541
3542
3543
3544
3545
3546
3547
3548
3549
3550
3551
3552
3553
3554
3555
3556
3557
3558
3559
3560
3561
3562
3563
3564
3565
3566
3567
3568
3569
3570
3571
3572
3573
3574
3575
3576
3577
3578
3579
3580
3581
3582
3583
3584
3585
3586
3587
3588
3589
3590
3591
3592
3593
3594
3595
3596
3597
3598
3599
3600
3601
3602
3603
3604
3605
3606
3607
3608
3609
3610
3611
3612
3613
3614
3615
3616
3617
3618
3619
3620
3621
3622
3623
3624
3625
3626
3627
3628
3629
3630
3631
3632
3633
3634
3635
3636
3637
3638
3639
3640
3641
3642
3643
3644
3645
3646
3647
3648
3649
3650
3651
3652
3653
3654
3655
3656
3657
3658
3659
3660
3661
3662
3663
3664
3665
3666
3667
3668
3669
3670
3671
3672
3673
3674
3675
3676
3677
3678
3679
3680
3681
3682
3683
3684
3685
3686
3687
3688
3689
3690
3691
3692
3693
3694
3695
3696
3697
3698
3699
3700
3701
3702
3703
3704
3705
3706
3707
3708
3709
3710
3711
3712
3713
3714
3715
3716
3717
3718
3719
3720
3721
3722
3723
3724
3725
3726
3727
3728
3729
3730
3731
3732
3733
3734
3735
3736
3737
3738
3739
3740
3741
3742
3743
3744
3745
3746
3747
3748
3749
3750
3751
3752
3753
3754
3755
3756
3757
3758
3759
3760
3761
3762
3763
3764
3765
3766
3767
3768
3769
3770
3771
3772
3773
3774
3775
3776
3777
3778
3779
3780
3781
3782
3783
3784
3785
3786
3787
3788
3789
3790
3791
3792
3793
3794
3795
3796
3797
3798
3799
3800
3801
3802
3803
3804
3805
3806
3807
3808
3809
3810
3811
3812
3813
3814
3815
3816
3817
3818
3819
3820
3821
3822
3823
3824
3825
3826
3827
3828
3829
3830
3831
3832
3833
3834
3835
3836
3837
3838
3839
3840
3841
3842
3843
3844
3845
3846
3847
3848
3849
3850
3851
3852
3853
3854
3855
3856
3857
3858
3859
3860
3861
3862
3863
3864
3865
3866
3867
3868
3869
3870
3871
3872
3873
3874
3875
3876
3877
3878
3879
3880
3881
3882
3883
3884
3885
3886
3887
3888
3889
3890
3891
3892
3893
3894
3895
3896
3897
3898
3899
3900
3901
3902
3903
3904
3905
3906
3907
3908
3909
3910
3911
3912
3913
3914
3915
3916
3917
3918
3919
3920
3921
3922
3923
3924
3925
3926
3927
3928
3929
3930
3931
3932
3933
3934
3935
3936
3937
3938
3939
3940
3941
3942
3943
3944
3945
3946
3947
3948
3949
3950
3951
3952
3953
3954
3955
3956
3957
3958
3959
3960
3961
3962
3963
3964
3965
3966
3967
3968
3969
3970
3971
3972
3973
3974
3975
3976
3977
3978
3979
3980
3981
3982
3983
3984
3985
3986
3987
3988
3989
3990
3991
3992
3993
3994
3995
3996
3997
3998
3999
3999
4000
4001
4002
4003
4004
4005
4006
4007
4008
4009
4010
4011
4012
4013
4014
4015
4016
4017
4018
4019
4020
4021
4022
4023
4024
4025
4026
4027
4028
4029
4030
4031
4032
4033
4034
4035
4036
4037
4038
4039
4040
4041
4042
4043
4044
4045
4046
4047
4048
4049
4050
4051
4052
4053
4054
4055
4056
4057
4058
4059
4060
4061
4062
4063
4064
4065
4066
4067
4068
4069
4070
4071
4072
4073
4074
4075
4076
4077
4078
4079
4080
4081
4082
4083
4084
4085
4086
4087
4088
4089
4090
4091
4092
4093
4094
4095
4096
4097
4098
4099
4099
4100
4101
4102
4103
4104
4105
4106
4107
4108
4109
4110
4111
4112
4113
4114
4115
4116
4117
4118
4119
4120
4121
4122
4123
4124
4125
4126
4127
4128
4129
4130
4131
4132
4133
4134
4135
4136
4137
4138
4139
4140
4141
4142
4143
4144
4145
4146
4147
4148
4149
4149
4150
4151
4152
4153
4154
4155
4156
4157
4158
4159
4160
4161
4162
4163
4164
4165
4166
4167
4168
4169
4170
4171
4172
4173
4174
4175
4176
4177
4178
4179
4180
4181
4182
4183
4184
4185
4186
4187
4188
4189
4190
4191
4192
4193
4194
4195
4196
4197
4198
4199
4199
4200
4201
4202
4203
4204
4205
4206
4207
4208
4209
4210
4211
4212
4213
4214
4215
4216
4217
4218
4219
4219
4220
4221
4222
4223
4224
4225
4226
4227
4228
4229
4229
4230
4231
4232
4233
4234
4235
4236
4237
4238
4239
4239
4240
4241
4242
4243
4244
4245
4246
4247
4248
4249
4249
4250
4251
4252
4253
4254
4255
4256
4257
4258
4259
4259
4260
4261
4262
4263
4264
4265
4266
4267
4268
4269
4269
4270
4271
4272
4273
4274
4275
4276
4277
4278
4279
4279
4280
4281
4282
4283
4284
4285
4286
4287
4288
4289
4289
4290
4291
4292
4293
4294
4295
4296
4297
4298
4299
4299
4300
4301
4302
4303
4304
4305
4306
4307
4308
4309
4309
4310
4311
4312
4313
4314
4315
4316
4317
4318
4319
4319
4320
4321
4322
4323
4324
4325
4326
4327
4328
4329
4329
4330
4331
4332
4333
4334
4335
4336
4337
4338
4339
4339
4340
4341
4342
4343
4344
4345
4346
4347
4348
4349
4349
4350
4351
4352
4353
4354
4355
4356
4357
4358
4359
4359
4360
4361
4362
4363
4364
4365
4366
4367
4368
4369
4369
4370
4371
4372
4373
4374
4375
4376
4377
4378
4379
4379
4380
4381
4382
4383
4384
4385
4386
4387
4388
4389
4389
4390
4391
4392
4393
4394
4395
4396
4397
4398
4399
4399
4400
4401
4402
4403
4404
4405
4406
4407
4408
4409
4409
4410
4411
4412
4413
4414
4415
4416
4417
4418
4419
4419
4420
4421
4422
4423
4424
4425
4426
4427
4428
4429
4429
4430
4431
4432
4433
4434
4435
4436
4437
4438
4439
4439
4440
4441
4442
4443
4444
4445
4446
4447
4448
4449
4449
4450
4451
4452
4453
4454
4455
4456
4457
4458
4459
4459
4460
4461
4462
4463
4464
4465
4466
4467
4468
4469
4469
4470
4471
4472
4473
4474
4475
4476
4477
4478
4479
4479
4480
4481
4482
4483
4484
4485
4486
4487
4488
4489
4489
4490
4491
4492
4493
4494
4495
4496
4497
4498
4498
4499
4499
4500
4501
4502
4503
4504
4505
4506
4507
4508
4509
4509
4510
4511
4512
4513
4514
4515
4516
4517
4518
4519
4519
4520
4521
4522
4523
4524
4525
4526
4527
4528
4529
4529
4530
4531
4532
4533
4534
4535
4536
4537
4538
4539
4539
4540
4541
4542
4543
4544
4545
4546
4547
4548
4549
4549
4550
4551
4552
4553
4554
4555
4556
4557
4558
4559
4559
4560
4561
4562
4563
4564
4565
4566
4567
4568
4569
4569
4570
4571
4572
4573
4574
4575
4576
4577
4578
4579
4579
4580
4581
4582
4583
4584
4585
4586
4587
4588
4589
4589
4590
4591
4592
4593
4594
4595
4596
4597
4598
4598
4599
4599
4600
4601
4602
4603
4604
4605
4606
4607
4608
4609
4609
4610
4611
4612
4613
4614
4615
4616
4617
4618
4619
4619
4620
4621
4622
4623
4624
4625
4626
4627
4628
4629
4629
4630
4631
4632
4633
4634
4635
4636
4637
4638
4639
4639
4640
4641
4642
4643
4644
4645
4646
4647
4648
4649
4649
4650
4651
4652
4653
4654
4655
4656
4657
4658
4659
4659
4660
4661
4662
4663
4664
4665
4666
4667
4668
4669
4669
4670
4671
4672
4673
4674
4675
4676
4677
4678
4679
4679
4680
4681
4682
4683
4684
4685
4686
4687
4688
4689
4689
4690
4691
4692
4693
4694
4695
4696
4697
4698
4698
4699
4699
4700
4701
4702
4703
4704
4705
4706
4707
4708
4709
4709
4710
4711
4712
4713
4714
4715
4716
4717
4718
4719
4719
4720
4721
4722
4723
4724
4725
4726
4727
4728
4729
4729
4730
4731
4732
4733
4734
4735
4736
4737
4738
4739
4739
4740
4741
4742
4743
4744
4745
4746
4747
4748
4749
4749
4750
4751
4752
4753
4754
4755
4756
4757
4758
4759
4759
4760
4761
4762
4763
4764
4765
4766
4767
4768
4769
4769
4770
4771
4772
4773
4774
4775
4776
4777
4778
4779
4779
4780
4781
4782
4783
4784
4785
4786
4787
4788
4789
4789
4790
4791
4792
4793
4794
4795
4796
4797
4798
4798
4799
4799
4800
4801
4802
4803
4804
4805
4806
4807
4808
4809
4809
4810
4811
4812
4813
4814
4815
4816
4817
4818
4819
4819
4820
4821
4822
4823
4824
4825
4826
4827
4828
4829
4829
4830
4831
4832
4833
4834
4835
4836
4837
4838
4839
4839
4840
4841
4842
4843
4844
4845
4846
4847
4848
4849
4849
4850
4851
4852
4853
4854
4855
4856
4857
4858
4859
4859
4860
4861
4862
4863
4864
4865
4866
4867
4868
4869
4869
4870
4871
4872
4873
4874
4875
4876
4877
4878
4879
4879
4880
4881
4882
4883
4884
4885
4886
4887
4888
4889
4889
4890
4891
4892
4893
4894
4895
4896
4897
4898
4898
4899
4899
4900
4901
4902
4903
4904
4905
4906
4907
4908
4909
4909
4910
4911
4912
4913
4914
4915
4916
4917
4918
4919
4919
4920
4921
4922
4923
4924
4925
4926
4927
4928
4929
4929
4930
4931
4932
4933
4934
4935
4936
4937
4938
4939
4939
4940
4941
4942
4943
4944
4945
4946
4947
4948
4949
4949
4950
4951
4952
4953
4954
4955
4956
4957
4958
4959
4959
4960
4961
4962
4963
4964
4965
4966
4967
4968
4969
4969
4970
4971
4972
4973
4974
4975
4976
4977
4978
4979
4979
4980
4981
4982
4983
4984
4985
4986
4987
4988
4989
4989
4990
4991
4992
4993
4994
4995
4996
4997
4998
4998
4999
4999
5000
5001
5002
5003
5004
5005
5006
5007
5008
5009
5009
5010
5011
5012
5013
5014
5015
5016
5017
5018
5019
5019
5020
5021
5022
5023
5024
5025
5026
5027
5028
5029
5029
5030
5031
5032
5033
5034
5035
5036
5037
5038
5039
5039
5040
5041
5042
5043
5044
5045
5046
5047
5048
5049
5049
5050
5051
5052
5053
5054
5055
5056
5057
5058
5059
5059
5060
5061
5062
5063
5064
5065
5066
5067
5068
5069
5069
5070
5071
5072
5073
5074
5075
5076
5077
5078
5079
5079
5080
5081
5082
5083
5084
5085
5086
5087
5088
5089
5089
5090
5091
5092
5093
5094
5095
5096
5097
5098
5098
5099
5099
5100
5101
5102
5103
5104
5105
5106
5107
5108
5109
5109
5110
5111
5112
5113
5114
5115
5116
5117
5118
5119
5119
5120
5121
5122
5123
5124
5125
5126
5127
5128
5129
5129
5130
5131
5132
5133
5134
5135
5136
5137
5138
5139
5139
5140
5141
5142
5143
5144
5145
5146
5147
5148
5149
5149
5150
5151
5152
5153
5154
5155
5156
5157
5158
5159
5159
5160
5161
5162
5163
5164
5165
5166
5167
5168
5169
5169
5170
5171
5172
5173
5174
5175
5176
5177
5178
5179
5179
5180
5181
5182
5183
5184
5185
5186
5187
5188
5189
5189
5190
5191
5192
5193
5194
5195
5196
5197
5198
5198
5199
5199
5200
5201
5202
5203
5204
5205
5206
5207
5208
5209
5209
5210
5211
5212
5213
5214
5215
5216
5217
5218
5219
5219
5220
5221
5222
5223
5224
5225
5226
5227
5228
5229
5229
5230
5231
5232
5233
5234
5235
5236
5237
5238
5239
5239
5240
5241
5242
5243
5244
5245
5246
5247
5248
5249
5249
5250
5251
5252
5253
5254
5255
5256
5257
5258
5259
5259
5260
5261
5262
5263
5264
5265
5266
5267
5268
5269
5269
5270
5271
5272
5273
5274
5275
5276
5277
5278
5279
5279
5280
5281
5282
5283
5284
5285
5286
5287
5288
5289
5289
5290
5291
5292
5293
5294
5295
5296
5297
5298
5298
5299
5299
5300
5301
5302
5303
5304
5305
5306
5307
5308
5309
5309
5310
5311
5312
5313
5314
5315
5316
5317
5318
5319
5319
5320
5321
5322
5323
5324
5325
5326
5327
5328
5329
5329
5330
5331
5332
5333
5334
5335
5336
5337
5338
5339
5339
5340
5341
5342
5343
5344
5345
5346
5347
5348
5349
5349
5350
5351
5352
5353
5354
5355
5356
5357
5358
5359
5359
5360
5361
5362
5363
5364
5365
5366
5367
5368
5369
5369
5370
5371
5372
5373
5374
5375
5376
5377
5378
5379
5379
5380
5381

```

```

3510      \let \^^M=\gmd@bslashEOL}%
3523 }

```

To make $\wedge\wedge M$ behave more like a ‘normal’ line end I command it to add a \wedge_{10} at first. It works but has one unwelcome feature: if the line has nearly textwidth , this closing space may cause line breaking and setting a blank line. To fix this I \backslash advance the \backslash parfillskip:

```

gmd@parfixclosingspace 3537 \def \gmd@parfixclosingspace { {%
3538     \advance \parfillskip by -\gmd@closingspacewd
3539     \if@aftercode \ifilrr \gmd@setilrr \fi \fi
3540     \par }%
3541     \if@ilgroup \aftergroup \egroup \ifilgroupfalse \fi% we are in the ver-
3542         batim group so we close the in-line comment group after it if the closing is
3543         not yet set.
3544 }

```

We’ll put it in a group surrounding \backslash par but we need to check if this \backslash par is executed after narration or after the code, i.e., whether the closing space was added or not.

```

\gmd@closingspacewd 3548 \newskip \gmd@closingspacewd
\gmd@setclosingspacewd 3549 \newcommand* \gmd@setclosingspacewd { %
3550     \global \gmd@closingspacewd=\fontdimen2\font%
3551     plus \fontdimen3\font minus \fontdimen4\font \relax}

```

See also line [2884](#) to see what we do in the codeline case when no closing space is added.

And one more detail:

```

3557 \foone \obeylines {%
3558     \if_1_1%
\gmd@bslashEOL 3559     \protected\def \gmd@bslashEOL { \xa \ignorespaces \wedge\wedge M }%
3560     }% of \foone. Note we interlace here \if with a group.
3561 \else%
\gmd@bslashEOL 3562     \protected\def \gmd@bslashEOL {%
3563         \ifhmode \unskip \fi \wedge\wedge M \ignorespaces }%
3565 \fi

```

The \backslash QueerEOL declaration will \backslash let it to $\wedge\wedge M$ to make $\wedge\wedge M$ behave properly. If this definition was omitted, $\wedge\wedge M$ would just expand to \wedge and thus not gobble the leading $\%$ of the next line leave alone typesetting the TeX code. I type \wedge etc. instead of just $\wedge\wedge M$ which adds a space itself because I take account of a possibility of redefining the \wedge CS by the user, just like in normal TeX.

We’ll need it for restoring queer definitions for doc-compatibility.

Adjustments of `verbatim` and `verb`

To make `verbatim`[\star] typeset its contents with the TeX code’s indentation:

```

\@verbatim 3588 \gaddtomacro \verbatim { \leftskip = \CodeIndent }

```

And a one more little definition to accommodate `verb` and pals for the lines commented out.

```

\check@percent 3592 \AtBeginInput { \long \def \check@percent #1 {%

```

```

3593 \gmd@cpnarrline% to count the verbatim lines and possibly print their
3594   numbers. This macro is used only by the verbatim end of line.
3595 \xa\ifx\code@delim#1\else\afterfi{#1}\fi}

```

We also redefine gmverb's \AddtoPrivateOthers that has been provided just with gmdoc's need in mind.

```

\AddtoPrivateOthers 3598 \def\AddtoPrivateOthers#1{%
3599   \xa\def\xadoprivateothers\@xa{%
3600     \doprivateothers\do#1}}%

```

We also redefine an internal \verb's macro \gm@verb@eol to put a proper line end if a line end char is met in a short verbatim: we have to check if we are in 'queer' or 'straight' EOLs area.

```

3611 \begingroup
3612 \obeylines%
\gm@verb@eol 3613 \AtBeginInput{\def\gm@verb@eol{\obeylines%
\verb@egroup 3614   \def^M{\verb@egroup\@latex@error{%
3615     \nx\verb@ended@by@end@of@line}%
3616     \@ifEOLactive{^M}{\@ehc}}}}%
3617 \endgroup

```

To distinguish the code typewriter from the narrative typewriter:

```

3620 \ampulexdef\@verbatim\ttverbatim\verbatim@specials
3621 {\ttverbatim\narrativett\verbatim@specials}
3623 \ampulexdef\verb\ttverbatim\verbatim@specials
3624 {\ttverbatim\narrativett\verbatim@specials}%
\texttt 3626 \pdef\texttt#1{{\narrativett#1}}

```

We rollback the \ampulexdef made to \verb in the index, see line 5895

Macros for marking of the macros

A great inspiration for this part was the doc package again. I take some macros from it, and some tasks I solve a different way, e.g., the \ (or another escape char) is not active, because anyway all the chars of code are scanned one by one. And exclusions from indexing are supported not with a list stored as \toks register but with separate control sequences for each excluded CS.

The doc package shows a very general approach to the indexing issue. It assumes using a special MakeIndex style and doesn't use explicit MakeIndex controls but provides specific macros to hide them. But here in gmdoc we prefer no special style for the index.

```

\actualchar 3653 \edef\actualchar{\string@}
\quotechar 3654 \edef\quotechar{\string"}
\encapchar 3655 \edef\encapchar{\xiclub}
\levelchar 3656 \edef\levelchar{\string!}

```

However, for the glossary, i.e., the change history, a special style is required, e.g., gm-glo.ist, and the above macros are redefined by the \changes command due to gm-glo.ist and gglo.ist settings.

Moreover, if you insist on using a special MakeIndex style, you may redefine the above four macros in the preamble. The \edefs that process them further are postponed till \begin{document}.

```

\CodeEscapeChar 3668 \def\CodeEscapeChar#1{%

```

```

3669  \begingroup
3670  \escapechar\m@ne
3671  \xdef\code@escape@char {\string#1}%
3672  \endgroup

```

As you see, to make a proper use of this macro you should give it a `\<one char>` CS as an argument. It's an invariant assertion that `\code@escape@char` stores 'other' version of the code layer escape char.

```
3678 \CodeEscapeChar \\
```

As mentioned in doc, someone may have some chars ₁₁ed.

```

3681 \@ifundefined{MakePrivateLetters} {%
3682   \def\MakePrivateLetters{\makeatletter\catcode`\\*=11\do}{}}

```

A tradition seems to exist to write about e.g., 'command `\section` and command `\section*`' and such an understanding also of 'macro' is noticeable in doc. Making the `*` a letter solves the problem of scanning starred commands.

And you may wish some special chars to be ₁₂.

```
3690 \def\MakePrivateOthers{\let\do=\@makeother\do\privateothers}
```

We use this macro to re`\catcode` the space for marking the environments' names and the caret for marking chars such as `^M`, see line [5233](#). So let's define the list:

```
3694 \def\doprivateothers{\do\do\do\^}
```

Two chars for the beginning, and also the `\MakeShortVerb` command shall this list enlarge with the char(s) declared. (There's no need to add the backslash to this list since all the relevant commands `\string` their argument whatever it is.)

Now the main macro indexing a macro's name. It would be a verbatim :-) copy of the doc's one if I didn't omit some lines irrelevant with my approach.

```

3708 \foone\obeylines{%
3709   \def\scan@macro#1{%
3710     \ifx#1^^M\@xa#1\else\afterfi{\scan@macro#1}\fi%
3711   }% of \scan@macro,
3712 }% of \foone.

```

```
\scan@macro@ 3715 \def\scan@macro#1{%
  we are sure to scan at least one token which is not the
  line end and therefore we define this macro as one-parameter.
```

Unlike in doc, here we have the escape char ₁₂ so we may just have it printed during main scan char by char, i.e., in the lines [3087](#) and [3091](#).

So, we step the checksum counter first,

```
3722 \step@checksum% (see line 6489 for details),
```

Then, unlike in doc, we do *not* check if the scanning is allowed, because here it's always allowed and required.

Of course, I can imagine horrible perversities, but I don't think they should really be taken into account. Giving the letter a `\catcode` other than ₁₁ surely would be one of those perversities. Therefore I feel safe to take the character `a` as a benchmark letter.

```

3731 \ifcat\do\@nx#1%
3732   \quote@char#1%
3733   \xdef\macro@iname{\gmd@maybequote#1}% global for symmetry with
                                             line 3751.

```

```
3735 \xdef\macro@pname{\string#1}%
      we'll print entire name of the macro
      later.
```

We `\string` it here and in the lines 3755 and 3767 to be sure it is whole 12 for easy testing for special index entry formats, see line 4655 etc. Here we are sure the result of `\string` is 12 since its argument is 11.

```
3742 \afterfi{\@ifnextcat{a}{\gmd@finishifstar#1}{%
      \finish@macroscan}}%
3743 \else% #1 is not a letter, so we have just scanned a one-char CS.
```

Another reasonable `\catcodes` assumption seems to be that the digits are 12. Then we don't have to type (%)`\expandafter\@gobble\string\a`. We do the `\uccode` trick to be sure that the char we write as the macro's name is 12.

```
3750 {\uccode`9=\#1%
3751   \uppercase{\xdef\macro@iname{9}}%
3752 }%
3753 \quote@char#1%
3754 \xdef\macro@iname{\gmd@maybequote\macro@iname}%
3755 \xdef\macro@pname{\xiistring#1}%
3756 \afterfi\finish@macroscan
3757 \fi}%
      of \scan@macro@. The \xiistring macro, provided by gmutils, is used
      instead of original \string because we wish to get 12('other' space).
```

Now, let's explain some details, i.e., let's define them. We call the following macro having known #1 to be 11.

```
\continue@macroscan 3764 \def\continue@macroscan#1{%
3765   \quote@char#1%
3766   \xdef\macro@iname{\macro@iname\gmd@maybequote#1}%
3767   \xdef\macro@pname{\macro@pname\string#1}%
      we know#1 to be 11, so
      we don't need \xiistring.
3770   \@ifnextcat{a}{\gmd@finishifstar#1}{\finish@macroscan}%
3771 }
```

As you may guess, `\@ifnextcat` is defined analogously to `\@ifnextchar` but the test it does is `\ifcat` (not `\ifx`). (Note it wouldn't work for an active char as the 'pattern'.)

We treat the star specially since in usual L^AT_EX it should finish the scanning of a CS name—we want to avoid scanning `\command\argum` as one CS.

```
\gmd@finishifstar 3780 \def\gmd@finishifstar#1{%
3781   \if*\@nx#1\afterfi\finish@macroscan%
      note we protect #1 against ex-
      pansion. In gmdoc verbatim scopes some chars are active (e.g. \).
3784   \else\afterfi\continue@macroscan
3785   \fi}
```

If someone *really* uses `*` as a letter please let me know.

```
\quote@char 3789 \def\quote@char#1{{\uccode`9=\#1%
      at first I took digit 1 for this \uc|
      codeing but then #1 meant #<#1> in \uppercase's argument, of course.
3792   \uppercase{%
3793     \@ifinmeaning{9}{\of}{\indexcontrols}
3794     {\glet\gmd@maybequote\quotechar}%
3795     {\gempty\gmd@maybequote}%
3796   }%
```

3797 } }

This macro is used for catching chars that are MakeIndex's controls. How does it work?

\quote@char sort of re\catcodes its argument through the \uccode trick: assigns the argument as the uppercase code of the digit 9 and does further work in the \uppercase's scope so the digit 9 (a benchmark 'other') is substituted by #1 but the \catcode remains so \gmd@ifinmeaning gets \quote@char's #1 'other'ed as the first argument.

In \quote@char the second argument for gutils \@ifinmeaning is \index| controls defined as the (expanded and 'other') sequence of the MakeIndex controls. \@ifinmeaning defines its inner macro \gmd@in@@ to take two parameters separated by the first and the second \@ifinmeaning's parameter, which are here the char investigated by \quote@char and the \indexcontrols list. The inner macro's parameter string is delimited by the macro itself, why not. \gmd@in@@ is put before a string consisting of \@ifinmeaning's second and first parameters (in such a reversed order) and \gmd@in@@ itself. In such a sequence it looks for something fitting its parameter pattern. \gmd@in@@ is sure to find the parameters delimiter (\gmd@in@@ itself) and the separator, \ifismember's #1 i.e., the investigated char, because they are just there. But the investigated char may be found not near the end, where we put it, but among the MakeIndex controls' list. Then the rest of this list and \ifismember's #1 put by us become the second argument of \gmd@in@@. What \gmd@in@@ does with its arguments, is just a check whether the second one is empty. This may happen *iff* the investigated char hasn't been found among the MakeIndex controls' list and then \gmd@in@@ shall expand to \iffalse, otherwise it'll expand to \iftrue. (The \after... macros are employed not to (mis)match just got \if... with the test's \fi.) "(Deep breath.) You got that?" If not, try doc's explanation of \ifnot@excluded, pp. 36–37 of the v2.1b dated 2004/02/09 documentation, where a similar construction is attributed to Michael Spivak.

Since version 0.99g \@ifinmeaning is used also in testing whether a detector is already present in the carrier in the mechanism of automatic detection of definitions (line 4002).

And now let's take care of the MakeIndex control characters. We'll define a list of them to check whether we should quote a char or not. But we'll do it at \begin{document} to allow the user to use some special MakeIndex style and in such a case to redefine the four MakeIndex controls' macros. We enrich this list with the backslash because sometimes MakeIndex didn't like it unquoted.

```
\indexcontrols 3851 \AtBeginDocument {\xdef\indexcontrols{%
3852     \bslash\levelchar\encapchar\actualchar\quotechar}}
\ifgmd@glosscs 3856 \newif\ifgmd@glosscs% we use this switch to keep the information whether
                  a history entry is a CS or not.
\finish@macroscan 3860 \newcommand*\finish@macroscan{%
```

First we check if the current CS is not just being defined. The switch may be set true in line 3899

```
3863 \ifgmd@adef@cshook% if so, we throw it into marginpar and index as a def
                  entry...
3865 \gm@ifundefined{gmd/iexcl/\macro@pname\space}{% ... if it's not
                  excluded from indexing.
3867 \xa\Code@MarginizeMacro\x{ \macro@pname}%
3868 \xa\defentryze\x{ \macro@pname}{1}{ }% here we declare the
                  kind of index entry and define \last@defmark used by \changes
```

```

3870      \global\gmd@adef@cshookfalse% we falsify the hook that was set true
            just for this CS.
3872  \fi

```

We have the CS's name for indexing in `\macro@iname` and for print in `\macro@pname`. So we index it. We do it a bit counter-crank way because we wish to use more general indexing macro.

```

3877  \if\verbatimchar\macro@pname% it's important that \verbatimchar comes
            before the macro's name: when it was reverse, the \tt CS turned this test
            true and left the \verbatimchar what resulted with '\+tt' typeset. Note
            that this test should turn true iff the scanned macro name shows to be the
            default \verb's delimiter. In such a case we give \verb another delimiter,
            namely $:

```

```

\im@firstpar 3884  \def\im@firstpar {[ $%
3885          ] }%
\im@firstpar 3886  \else\def\im@firstpar {}%
3887  \fi
3888  \@xa\index@macro\im@firstpar\macro@iname\macro@pname
3889  \maybe@marginpar\macro@pname
3890  \if\xiispace\macro@pname\relax\gmd@texcodespace
3891  \else
3892      {\noverbatimspecials\RestoreMacro\verb
3893          \@xa\scanverb\@xa{\macro@pname}}% we typeset scanned CS.
3894  \fi
3895  \let\next\gmd@charbychar
3896  \gmd@detectors% for automatic detection of definitions. Defined and ex-
            plained in the next section. It redefines \next if detects a definition com-
            mand and thus sets the switch of line 3860 true.
3897  \next
3898 }

```

Now, the macro that checks whether the just scanned macro should be put into a marginpar: it checks the meaning of a very special CS: whose name consists of `gmd/2marpar/` and of the examined macro's name.

```

\maybe@marginpar 3912 \def\maybe@marginpar#1{%
3913     \gm@ifundefined{gmd/2marpar/\@xa\detokenize\@xa{#1}}{}{%
3914         \edef\gmu@tempa{%
3915             \unexpanded{\Text@Marginize*}%
3916             {\bslash\@xa\unexpanded\@xa{#1}}%
3917         }\gmu@tempa
3918
3919         \macro@pname, which will be the only possible argument to \maybe@margin@
3920             % inpar, contains the macro's name without the escape char so we
3921             added it here.
3922
3923         \@xa\g@relaxen
3924         \csname gmd/2marpar/\@xa\detokenize\@xa{#1}\endcsname% we re-
            set the switch.
3925     }
}

```

Since version 0.99g we introduce automatic detection of definitions, it will be implemented in the next section. The details of indexing CSes are implemented in the section after it.

Automatic detection of definitions

To begin with, let's introduce a general declaration of a defining command. `\DeclareDefining` comes in two flavours: 'sauté', and with star. The 'sauté' version without an optional argument declares a defining command of the kind of `\def` and `\newcommand`: whether wrapped in braces or not, its main argument is a CS. The star version without the optional argument declares a defining command of the kind of `\newenvironment` and `\DeclareOption`: whose main mandatory argument is text. Both versions provide an optional argument in which you can set the keys. Probably the most important key is `star`. It determines whether the starred version of a defining command should be taken into account. For example, `\newcommand` should be declared with `[star=true]` while `\def` with `[star=false]`. You can also write just `[star]` instead of `[star=true]`. It's the default if the `star` key is omitted.

Another key is `type`. Its possible values are the (backslashless) names of the defining commands, see below.

We provide now more keys for the `xkeyvalish` definitions: `KVpref` (the key prefix) and `KVfam` (the key family). If not set by the user, they are assigned the default values as in `xkeyval`: `KVpref` letters `KV` and `KVfam` the input file name. The latter assignment is done only for the `\DeclareOptionX` defining command because in other `xkeyval` definitions (`\define@{...}key`) the family is mandatory.

`\DeclareDefining` and the detectors

Note that the main argument of the next declaration should be a CS *without star*, unless you wish to declare only the starred version of a command. The effect of this command is always global.

```
\DeclareDefining 3970 \outer\def\DeclareDefining{\begingroup
3971   \MakePrivateLetters
3972   \gma@ifstar
3973     {\gdef\gmd@adef@defaulttype{text}\Declare@Dfng}%
3974     {\gdef\gmd@adef@defaulttype{cs}\Declare@Dfng}%
3975 }
```

The keys except `star` depend of `\gmd@adef@currdef`, therefore we set them having known both arguments

```
\Declare@Dfng 3979 \newcommand*\Declare@Dfng[2][]{%
3980   \endgroup
3981   \Declare@Dfng@inner{\#1}{\#2}%
3982   \ifgmd@adef@star% this switch may be set false in first \Declare@Dfng@inner
3983     (it's the star key).
3984     \Declare@Dfng@inner{\#1}{\#2*}%
3985     The catcode of * doesn't matter since
3986     it's in \csname...\endcsname everywhere.
3988 }
```

```
\Declare@Dfng@inner 3991 \def\Declare@Dfng@inner#1#2{%
3992   \edef\gmd@resa{%
3993     \nx\setkeys[gmd]{adef}{type=\gmd@adef@defaulttype}}%
3994   \gmd@resa
3995   {\escapechar\m@ne
3996     \xdef\gmd@adef@currdef{\string#2}%
3998 }%
3999 \gmd@adef@setkeysdefault
4000 \setkeys[gmd]{adef}{#1}%
4001 \xa@ifinmeaning
```

```

4002 \csname_gmd@detect@\gmd@adef@currdef\endcsname
4004 \of\gmd@detectors{}{%
4005   \xa\gaddtomacro\x\gmd@detectors\@xa{%
4006     \csname_gmd@detect@\gmd@adef@currdef\endcsname} }% we add
4007     a CS
4008     % \gmd@detect@<def name> (a detector) to the meaning of the
4009     detectors' carrier. And we define it to detect the #2 command.
4010   \xa\xdef\csname_gmd@detectname@\gmd@adef@currdef%
4011     \endcsname{%
4012   \gmd@adef@currdef}%
4013   \edef\gmu@tempa{%
4014     this \edef is to expand \gmd@adef@TYPE.
4015     \global\@nx\@namedef{\gmd@detect@\gmd@adef@currdef}{%
4016       \@nx\ifx
4017         \xa\@nx\csname_gmd@detectname@\gmd@adef@currdef%
4018           \endcsname
4019         \@nx\macro@pname
4020         \@nx\n@melet{next}{\gmd@adef@\gmd@adef@TYPE}%
4021         \@nx\n@melet{\gmd@adef@currdef}{\gmd@detectname@%
4022           \gmd@adef@currdef}%
4023         \@nx\fi} }%
4024   \gmu@tempa
4025   \SMglobal\StoreMacro*{\gmd@detect@\gmd@adef@currdef}%
4026     we store the
4027     CS to allow its temporary discarding later.
4028 }

nd@adef@setkeysdefault 4029 \def\gmd@adef@setkeysdefault{%
4030   \setkeys[gmd]{adef}{star,prefix,KVpref}}}

Note we don't set KVfam. We do not so because for \define@key-likes family is
a mandatory argument and for \DeclareOptionX the default family is set to the input
file name in line 4200.

star 4033 \define@boolkey[gmd]{adef}{star}[true]{}

The prefix@<command> key-value will be used to create additional index entry for
detected definiendum (a definiendum is the thing defined, e.g. in \newenvironment{%
foo} the env. foo). For instance, \newcounter is declared with [prefix=\bslash_c@] in line 4463 and therefore \newcounter{foo} occurring in the code will index
both foo and \c@foo (as definition entries).

prefix 4042 \define@key[gmd]{adef}{prefix}[]{%
4043   \edef\gmd@resa{%
4044     \def\x\@nx\csname_gmd@adef@prefix@\gmd@adef@currdef\endcsname{%
4045       #1} }%
4046   \gmd@resa}

\gmd@KVprefdefault 4049 \def\gmd@KVprefdefault{KV}%
in a separate macro because we'll need it in
\ifx.

A macro \gmd@adef@KVprefixset@<command> if defined, will falsify an \ifnum
test that will decide whether create additional index entry together with the tests for
prefix<command> and

KVpref 4057 \define@key[gmd]{adef}{KVpref}[\gmd@KVprefdefault]{%
4058   \edef\gmd@resa{#1}%
4059   \ifx\gmd@resa\gmd@KVprefdefault

```

```

4060 \else
4061   @namedef{gmd@adef@KVprefixset@\gmd@adef@currdef}{1}%
4062   \gmd@adef@setKV% whenever the KVprefix is set (not default), the de-
        clared command is assumed to be keyvalish.
4064 \fi
4065 \edef\gmd@resa{\#1}% because \gmd@adef@setKV redefined it.
4066 \edef\gmd@resa{%
4067   \def\@xa\@nx\csname\gmd@adef@KVpref@\gmd@adef@currdef%
        \endcsname{%
4068     \ifx\gmd@resa\empty
4069     \else\#1@ \fi} }% as in xkeyval, if the KV prefix is not empty, we add @ to
        it.
4071 \gmd@resa}

```

Analogously to KVpref, KVfam declared in \DeclareDefining will override the family scanned from the code and, in \DeclareOptionX case, the default family which is the input file name (only for the command being declared).

```

KVfam 4078 \define@key[gmd]{adef}{KVfam}[]{%
4079   \edef\gmd@resa{\#1}%
4080   @namedef{gmd@adef@KVfamset@\gmd@adef@currdef}{1}%
4081   \edef\gmd@resa{%
4082     \def\@xa\@nx\csname\gmd@adef@KVfam@\gmd@adef@currdef%
        \endcsname{%
4083       \ifx\gmd@resa\empty
4084       \else\#1@ \fi} }%
4085   \gmd@resa
4086   \gmd@adef@setKV }% whenever the KVfamily is set, the declared command is
        assumed to be keyvalish.

```

```

type 4090 \define@choicekey[gmd]{adef}{type}
4091   [ \gmd@adef@typevals\gmd@adef@typenr]
4092   { % the list of possible types of defining commands
4093     def,
4094     newcommand,
4095     cs, % equivalent to the two above, covers all the cases of defining a CS, includ-
        ing the PLAIN TeX \new... and LATEX \newlength.
4096     newenvironment,
4097     text, % equivalent to the one above, covers all the commands defining its first
        mandatory argument that should be text, \DeclareOption e.g.
4098     define@key, % special case of more arguments important; covers the xkeyval
        defining commands.
4099     dk, % a shorthand for the one above.
4100     DeclareOptionX, % another case of special arguments configuration, covers
        the xkeyval homonym.
4101     dox, % a shorthand for the one above.
4102     kvo% one of option defining commands of the kvoptions package by Heiko
        Oberdiek (a package available o CTAN in the oberdiek bundle).
4103   }
4104   { % In fact we collapse all the types just to four so far:
4105     \ifcase\gmd@adef@typenr% if def
4106       \gmd@adef@settype{cs}{o}%
4107     \or% when newcommand
4108       \gmd@adef@settype{cs}{o}%

```

```

4117   \or% when cs
4118     \gmd@adef@settype{cs}{o}%
4119   \or% when newenvironment
4120     \gmd@adef@settype{text}{o}%
4121   \or% when text
4122     \gmd@adef@settype{text}{o}%
4123   \or% when define@key
4124     \gmd@adef@settype{dk}{1}%
4125   \or% when dk
4126     \gmd@adef@settype{dk}{1}%
4127   \or% when DeclareOptionX
4128     \gmd@adef@settype{dox}{1}%
4129   \or% when dox
4130     \gmd@adef@settype{dox}{1}%
4131   \or% when kvo
4132     \gmd@adef@settype{text}{1}%
4133       The kvoptions option definitions take
4134         first mandatory argument as the option name and they define a key-
4135         val key whose macro's name begins with the prefix/family, either de-
4136         fault or explicitly declared. The kvoptions prefix/family is supported
4137         in gmdoc with [KVpref=, □KVfam=<family>].
4138   \fi}
4139 \gmd@adef@settype
4140   \def\gmd@adef@settype#1#2{%
4141     \def\gmd@adef@TYPE{#1}%
4142     \ifnum1=#2%
4143       now we define (or not) a quasi-switch that fires for the keyvalish
4144       definition commands.
4145     \gmd@adef@setKV
4146   \fi}
4147 \gmd@adef@setKV
4148   \edef\gmd@resa{%
4149     \def\@xa\@nx\csname\gmd@adef@KV@\gmd@adef@currdef%
4150       \endcsname{1}%
4151   }%
4152   \gmd@resa}

```

We initialise the carrier of detectors:

```
4155 \emptyify\gmd@detectors
```

The definiendum of a command of the cs type is the next control sequence. Therefore we only need a self-relaxing hook in \finish@macroscan.

```
\ifgmd@adef@cshook
4161 \newif\ifgmd@adef@cshook
4163 \def\gmd@adef@cs{\global\gmd@adef@cshooktrue\gmd@charbychar}
```

For other kinds of definitions we'll employ active chars of their arguments' opening braces, brackets and sergeants. In gmdoc code layer scopes the left brace is active so we only add a hook to its meaning (see line 371 in gmverb) and here we switch it according to the type of detected definition.

```
\gmd@adef@text
4171 \def\gmd@adef@text{\gdef\gmd@lbracecase{1}\gmd@charbychar}
4172 \foone{%
4173   \catcode`\\[\active
4174   \catcode`\\<\active}
4175 {%
4176 }
```

The detector of xkeyval \define@ [...] key:

```
\gmd@adef@dk 4179 \def\gmd@adef@dk {%
4180   \let[\gmd@adef@scanKVpref
4181   \catcode`\[\active
4182   \gdef\gmd@lbracecase{2}%
4183   \gmd@adef@dfKVpref\gmd@KVprefdefault% We set the default value of
4184     the xkeyval prefix. Each time again because an assignment
4185     in \gmd@adef@dfKVpref is global.
4186   \gmd@adef@checklbracket}
```

The detector of xkeyval \DeclareOptionX:

```
\gmd@adef@dox 4190 \def\gmd@adef@dox {%
4191   \let[\gmd@adef@scanKVpref
4192   \let<\gmd@adef@scanDOXfam
4193   \catcode`[\active
4194   \catcode`<\active
4195   \gdef\gmd@lbracecase{1}%
4196   \gmd@adef@dfKVpref\gmd@KVprefdefault% We set the default values of
4197     the xkeyval prefix...
4198   \edef\gmd@adef@fam{\gmd@inputname}% ... and family.
4199   \gmd@adef@dofam
4200   \gmd@adef@checkDOXopts}%
4201 }
```

The case when the right bracket is next to us is special because it is already touched by \futurelet (of CSes scanning macro's \ifnextcat), therefore we need a 'future' test.

```
\gmd@adef@checklbracket 4208 \def\gmd@adef@checklbracket {%
4209   \ifnextchar[%
4210     \gmd@adef@scanKVpref\gmd@charbychar}% note that the prefix scanning
        macro gobbles its first argument (undelimited) which in this case is [.
```

After a \DeclareOptionX-like defining command not only the prefix in square brackets may occur but also the family in sergeants. Therefore we have to test presence of both of them.

```
\gmd@adef@checkDOXopts 4218 \def\gmd@adef@checkDOXopts {%
4219   \ifnextchar[\gmd@adef@scanKVpref%
4220     {\ifnextchar<\gmd@adef@scanDOXfam\gmd@charbychar}}
```

```
\gmd@adef@scanKVpref 4224 \def\gmd@adef@scanKVpref#1#2 {%
4225   \gmd@adef@dfKVpref{#2}%
4226   [#2]\gmd@charbychar}
```

```
\gmd@adef@dfKVpref 4229 \def\gmd@adef@dfKVpref#1 {%
4230   \ifnum1=0\csname\gmd@adef@KVprefixset@\gmd@adef@currdef%
        \endcsname
      \relax
    \else
      \edef\gmu@resa{%
        \gdef\@xa\@nx
        \csname\gmd@adef@KVpref@\gmd@adef@currdef\endcsname{%
          \ifx\relax#1\relax
            \else#1@%
```

```

4238      \fi}}%
4239      \gmu@resa
4240      \fi}
\gmd@adef@scanDOXfam 4243 \def\gmd@adef@scanDOXfam{%
4244   \ifnum12=\catcode`\
4245     \relax
4246   \let\next\gmd@adef@scansfamoth
4247   \else
4248     \ifnum13=\catcode`\
4249       \relax
4250       \let\next\gmd@adef@scansfamact
4251   \else
4252     \PackageError{gmdoc}{> neither `other' nor `active'!}%
4253     \MakeUit
4254     `other' with \bslash AddtoPrivateOthers \bslash >.%}
4255   \fi
4256   \fi
4257   \next}

\gmd@adef@scansfamoth 4258 \def\gmd@adef@scansfamoth#1>{%
4259   \edef\gmd@adef@fam{\@gobble#1}% there is always \gmd@charbychar
4260   first.
4261   \gmd@adef@dofam
4262   <\gmd@adef@fam>%
4263   \gmd@charbychar}

\gmd@adef@scansfamact 4264 \foone{\catcode`\
4265   \active}{%
4266   \def\gmd@adef@scansfamact#1>{%
4267     \edef\gmd@adef@fam{\@gobble#1}% there is always \gmd@charbychar
4268     first.
4269     \gmd@adef@dofam
4270     <\gmd@adef@fam>%
4271     \gmd@charbychar}%
4272 }

The hook of the left brace consists of \ifcase that logically consists of three sub-cases:
0 —the default: do nothing in particular;
1 —the detected defining command has one mandatory argument (is of the text type, including koptions option definition);
2–3 —we are after detection of a \define@key-like command so we have to scan two mandatory arguments (case 2 is for the family, case 3 for the key name).

```

```

\gm@lbracehook 4285 \def\gm@lbracehook{%
4286   \ifcase\gmd@lbracecase\relax
4287   \or% when 1
4288     \afterfi{%
4289       \gdef\gmd@lbracecase{0}%
4290       \gmd@adef@scanname}%
4291   \or% when 2—the first mandatory argument of two (\define@...key)
4292     \afterfi{%
4293       \gdef\gmd@lbracecase{3}%
4294       \gmd@adef@scansDKfam}%
4295   \or% when 3—the second mandatory argument of two (the key name).
4296     \afterfi{%
4297       \gdef\gmd@lbracecase{0}%

```

```

4298      \gmd@adef@scancode}%
4299      \fi}
\gmd@lbracecase 4301 \def\gmd@lbracecase{o}{ we initialise the hook caser.

```

And we define the inner left brace macros:

```

4306 \foone{\catcode`[\@gobble\catcode`\]_@\catcode`\}\_@\}
4307 [% Note that till line 4330 the square brackets are grouping and the right brace is
     'other'.

```

Define the macro that reads and processes the `\define@key` family argument. It has the parameter delimited with 'other' right brace. An active left brace that has launched this macro had been passed through iterating `\gmd@charbychar` that now stands next right to us.

```

\gmd@adef@scanDKfam 4314 \def\gmd@adef@scanDKfam#1[%%
4315   \edef\gmd@adef@fam[\@gobble#1]{ there is always \gmd@charbychar
        first.
4317   \gmd@adef@dofam
4318   \gmd@adef@fam}%
4319 \gmd@charbychar]

\gmd@adef@scancode 4322 \def\gmd@adef@scancode#1[%%
4323   \@makeother\[@%
4324   \@makeother\<%

```

The scanned name begins with `\gmd@charbychar`, we have to be careful.

```

4327   \gmd@adef@deftext[#1]%
4328   \@gobble#1}%
4329   \gmd@charbychar]
4330 ]

\gmd@adef@dofam 4333 \def\gmd@adef@dofam{%
4334   \ifnum1=0\csname\gmd@adef@KVfamset@\gmd@adef@currdef%
        \endcsname
4335   \relax% a family declared with \DeclareDefining overrides the one cur-
        rently scanned.
4337   \else
4338     \edef\gmu@resa{%
4339       \gdef\@xa\@nx
4340       \csname\gmd@adef@KVfam@\gmd@adef@currdef\endcsname
4341       {\ifx\gmd@adef@fam\empty
4342         \else\gmd@adef@fam\@%
4343         \fi}%
4344       \gmu@resa
4345     \fi}

\gmd@adef@deftext 4347 \def\gmd@adef@deftext#1{%
4348   \@xa\def\@xa\macro@pname\@xa{\@gobble#1}{ we gobble \gmd@charbychar,
        cf. above.
4349   \edef\macro@pname{\@xa\detokenize\@xa{\macro@pname}}% note the
        space at the end.
4351   \edef\macro@pname{\@xa\@xiispaces\macro@pname\@nil}%
4352   \@xa\Text@Marginize\@xa{\macro@pname}%
4353   \gmd@adef@indextext
4354   \edef\gmd@adef@altindex{%

```

```

4355      \csname_gmd@adef@prefix@\gmd@adef@currdef_\endcsname}%
and we add the xkeyval header if we are in xkeyval definition.

4358  \ifnum1=0 \csname_gmd@adef@KV@\gmd@adef@currdef_\endcsname}%
      \relax% The
      CS \gmd@adef@KV@def. command is defined {1} (so \ifnum gets
      1=0\relax—true) iff def. command is a keyval definition. In
      that case we check for the KVprefix and KVfamily. (Otherwise
      \gmd@adef@KV@def. command is undefined so \ifnum gets
      1=0\relax—false.)
4364  \edef\gmd@adef@altindex{%
4365    \gmd@adef@altindex
4366    \csname_gmd@adef@KVpref@\gmd@adef@currdef_\endcsname}%
4367  \edef\gmd@adef@altindex{%
4368    \gmd@adef@altindex
4369    \csname_gmd@adef@KVfam@\gmd@adef@currdef_\endcsname}%
4370  \fi
4371  \ifx\gmd@adef@altindex\empty
4372  \else% we make another index entry of the definiendum with prefix/KVheader.
        \edef\macro@pname{\gmd@adef@altindex\macro@pname}%
        \gmd@adef@indextext
4375  \fi}
4377 \def\gmd@adef@indextext{%
4378  \@xa\@defentryze\@xa{\macro@pname}{0}%
      declare the definiendum has
      to have a definition entry and should appear without backslash in the
      changes history.
4381  \gmd@doindexingtext% redefine \do to an indexing macro.
4383  \@xa\do\@xa{\macro@pname}}

```

So we have implemented automatic detection of definitions. Let's now introduce some.

Default defining commands

Some commands are easy to declare as defining:

```

4397 \DeclareDefining[star=false]\def
\pdef 4398 \DeclareDefining[star=false]\pdef% it's a gutils' shorthand for \protected
      % \def.
\provide 4399 \DeclareDefining[star=false]\provide% a gutils' conditional \def.
\pprovide 4400 \DeclareDefining[star=false]\pprovide% a gutils' conditional \pdef.

```

But \def definitely *not always* defines an important macro. Sometimes it's just a scratch assignment. Therefore we define the next declaration. It turns the next occurrence of \def off (only the next one).

```

\UnDef 4408 \def\UnDef{{%
4412   \gmd@adef@selfrestore\def
4413   } }
\UnPdef 4416 \def\UnPdef{{\gmd@adef@selfrestore\pdef}}
4418 \StoreMacro\UnDef% because the 'hiding' commands relax it.
\HideDef 4420 \def\HideDef{%
\relaxen 4422   \gm@ifstar\UnDef{\HideDefining\def\relaxen\UnDef}}

```

```

\ResumeDef 4424 \def \ResumeDef { \ResumeDefining \def \RestoreMacro \UnDef }
\RestoreMacro

Note that I don't declare \gdef, \edef neither \xdef. In my opinion their use as
'real' definition is very rare and then you may use \Define implemented later.

\newcount 4431 \DeclareDefining [star=false] \newcount
\newdimen 4432 \DeclareDefining [star=false] \newdimen
\newskip 4433 \DeclareDefining [star=false] \newskip
\newif 4434 \DeclareDefining [star=false] \newif
\newtoks 4435 \DeclareDefining [star=false] \newtoks
\newbox 4436 \DeclareDefining [star=false] \newbox
\newread 4437 \DeclareDefining [star=false] \newread
\newwrite 4438 \DeclareDefining [star=false] \newwrite
\newlength 4439 \DeclareDefining [star=false] \newlength
\DeclareDocumentCommand 4440 \DeclareDefining [star=false] \DeclareDocumentCommand
\DeclareCommand 4441 \DeclareDefining [star=false] \DeclareCommand

\renewcommand 4445 \DeclareDefining \renewcommand
\ProvideDocumentCommand 4446 \DeclareDefining \renewcommand
\ProvideTextCommand 4447 \DeclareDefining \providecommand
\DeclareRobustCommand 4448 \DeclareDefining \DeclareRobustCommand
\DeclareTextCommand 4449 \DeclareDefining \DeclareTextCommand
\DeclareTextCommandDefault 4450 \DeclareDefining \DeclareTextCommandDefault

\newenvironment 4452 \DeclareDefining* \newenvironment
\renewenvironment 4453 \DeclareDefining* \renewenvironment
\DeclareOption 4454 \DeclareDefining* [star=false] \DeclareOption

% \DeclareDefining* \@namedef

\newcounter 4463 \DeclareDefining* [prefix=\bslash_c@] \newcounter% this prefix provides
               indexing also \c@<counter>.

\define@key 4466 \DeclareDefining [type=dk, \prefix=\bslash] \define@key
\define@boolkey 4467 \DeclareDefining [type=dk, \prefix=\bslash_if] \define@boolkey% the
               alternate index entry will be \if<KVpref>@\<KVfam>@\<key name>
\define@choicekey 4470 \DeclareDefining [type=dk, \prefix=\bslash] \define@choicekey

\DeclareOptionX 4472 \DeclareDefining [type=dox, \prefix=\bslash] \DeclareOptionX% the al-
               ternate index entry will be \<KVpref>@\<KVfam>@\<option name>.

For \DeclareOptionX the default KVfamily is the input file name. If the source
file name differs from the name of the goal file (you TEX a .dtx not .sty e.g.), there is the
next declaration. It takes one optional and one mandatory argument. The optional is
the KVpref, the mandatory the KVfam.
```

```

\DeclareDOXHead 4481 \newcommand* \DeclareDOXHead [2] [ \gmd@KVprefdefault ] {%
4482   \csname \DeclareDefining \endcsname
4483   [type=dox, \prefix=\bslash, \KVpref=#1, \KVfam=#2]%
\DeclareOptionX 4484   \DeclareOptionX
4485 }
```

An example:

```
4491 \DeclareOptionX[Berg]<Lulu>{EvelynLear} {}
```

Check in the index for EvelynLear and \Berg@\Lulu@\EvelynLear. Now we set
in the comment layer \DeclareDOXHead[Webern]{Lieder} and

```
ChneOelze 4496 \DeclareOptionX<AntonW>{ChneOelze}
```

The latter example shows also overriding the option header by declaring the default. By the way, both the example options are not declared in the code actually.

Now the Heiko Oberdiek's `kvoptions` package option definitions:

```

\DeclareStringOption
    \DeclareStringOption
\DeclareBoolOption
    \DeclareBoolOption
\DeclareComplementaryOption
    \DeclareComplementaryOption
\DeclareVoidOption
    \DeclareVoidOption

```

The `kvoptions` option definitions allow setting the default family/prefix for all definitions forth so let's provide analogon:

```

\def \DeclareKVOFam#1 {%
  \def \do##1 {%
    \csname \ DeclareDefining \endcsname
    [type=kvo, prefix=\bslash, KVpref=, KVfam=#1]##1}%
  \do \DeclareStringOption
  \do \DeclareBoolOption
  \do \DeclareComplementaryOption
  \do \DeclareVoidOption
}

```

As a nice exercise I recommend to think why this list of declarations had to be preceded (in the comment layer) with `\HideAllDefining` and for which declarations of the above `\DeclareDefining``\DeclareDefining` did not work. (The answers are commented out in the source file.)

One remark more: if you define (in the code) a new defining command (I did: a shorthand for `\DeclareOptionX[gmcc] <>`), declare it as defining (in the commentary) *after* it is defined. Otherwise its first occurrence shall fire the detector and mark next CS or worse, shall make the detector expect some arguments that it won't find.

Suspending ('hiding') and resuming detection

Sometimes we want to suspend automatic detection of definitions. For `\def` we defined suspending and resuming declarations in the previous section. Now let's take care of detection more generally.

The next command has no arguments and suspends entire detection of definitions.

```

\HideAllDefining
\def \HideAllDefining {%
  \ifnumo=0 \csname gmd@adef@allstored \endcsname
  \SMglobal \StoreMacro \gmd@detectors
  \global \namedef{gmd@adef@allstored}{1}%
}
\global \emptyify \gmd@detectors}%
we make the carrier \empty not \relax
to be able to declare new defining command in the scope of \HideAll...

```

The `\ResumeAllDefining` command takes no arguments and restores the meaning of the detectors' carrier stored with `\HideAllDefining`

```

\ResumeAllDefining
\def \ResumeAllDefining {%
  \ifnumi=0 \csname gmd@adef@allstored \endcsname \relax
}

```

```

4570   \SMglobal\RestoreMacro\gmd@detectors
4571   \SMglobal\RestoreMacro\UnDef
4572   \global\@namedef{gmd@adef@allstored}{0}%
4573 \fi}

```

Note that `\ResumeAllDefining` discards the effect of any `\DeclareDefining` that could have occurred between `\HideAllDefining` and itself.

The `\HideDefining` command takes one argument which should be a defining command (always without star). `\HideDefining` suspends detection of this command (also of its starred version) until `\ResumeDefining` of the same command or `\ResumeAllDefining`.

```

\HideDefining 4585 \def\HideDefining{\begingroup
4588   \MakePrivateLetters
4589   \gmd@ifstar\Hide@DfngOnce\Hide@Dfng}

\Hide@Dfng 4591 \def\Hide@Dfng#1{%
4592   \escapechar\m@ne
4593   \gn@melet{gmd@detect@\string#1}{relax}%
4594   \gn@melet{gmd@detect@\string#1*}{relax}%
4595   \ifx\def#1\global\relaxen\UnDef\fi
4596 \endgroup}

\Hide@DfngOnce 4598 \def\Hide@DfngOnce#1{%
4599   \gmd@adef@selfrestore#1%
4600 \endgroup

4602 \def\gmd@adef@selfrestore#1{%
4603   \escapechar\m@ne
4604   \@ifundefined{gmd@detect@\string#1}{%
4605     \SMglobal@xa\StoreMacro
4606     \csname\gmd@detect@\string#1\endcsname}{}%
4608   \global\@nameedef{gmd@detect@\string#1}{%
4609     \@nx\ifx\@xa\@nx\csname\gmd@detectname@\string#1%
4610       \endcsname
4611     \@nx\macro@pname
4612     \def\@nx\next{%
4613       \SMglobal\RestoreMacro% this \next will be executed in line 3904.
4614       \@xa\@nx\csname\gmd@detect@\string#1\endcsname
4615       \@nx\gmd@charbychar}%
4616     \@nx\fi}%
4617   }%
4618 }%
4619 }%
4620 }%
4621 }%
4622 }%
4623 }%
4624 }%
4625 }%
4626 }%
4627 }%
4628 }%
4629 }%
4630 }%
4631 }%
4632 }%
4633 }%
4634 }%
4635 }%
4636 }%
4637 }%
4638 }%
4639 }%

```

The `\ResumeDefining` command takes a defining command as the argument and resumes its automatic detection. Note that it restores also the possibly undefined detectors of starred version of the argument but that is harmless I suppose until we have millions of CSes.

```

\ResumeDefining 4631 \def\ResumeDefining{\begingroup
4632   \MakePrivateLetters
4633   \gmd@ResumeDfng}

\gmd@ResumeDfng 4635 \def\gmd@ResumeDfng#1{%
4636   \escapechar\m@ne
4637   \SMglobal\RestoreMacro*{gmd@detect@\string#1}%
4638   \SMglobal\RestoreMacro*{gmd@detect@\string#1*}%
4639 \endgroup}

```

Indexing of CSes

The inner macro indexing macro. #1 is the \verb's delimiter; #2 is assumed to be the macro's name with MakeIndex-control chars quoted. #3 is a macro storing the ₁₂ macro's name, usually \macro@pname, built with \stringing every char in lines 3735, 3755 and 3767. #3 is used only to test if the entry should be specially formatted.

```

\index@macro 4651 \newcommand* \index@macro [3] [ \verb+verbatimchar+] { {%
4652   \gm@ifundefined{gmd/iexcl/} {\xa\detokenize\@xa{\#3}} %
4653   { % #3 is not excluded from index
4654     \gm@ifundefined{gmd/defentry/} {\xa\detokenize\@xa{\#3}} %
4655     { % #3 is not def entry
4656       \gm@ifundefined{gmd/usgentry/} {\xa\detokenize\@xa{\#3}} %
4657       { % #3 is not usg. entry
4658         \edef\kind@fentry{\CommonEntryCmd} %
4659         { % #3 is usg. entry
4660           \def\kind@fentry{UsgEntry} %
4661           \un@usgentryze{\#3} %
4662         } %
4663       } %
4664       { % #3 is def entry
4665         \def\kind@fentry{DefEntry} %
4666         \un@defentryze{\#3} %
4667       } %
4668       } % of gmd/defentry/ test's 'else'
\if@pageindex \@pageinindexfalse \fi% should it be here or there?
        Definitely here because we'll wish to switch the switch with a decla-
        ration.
\if@pageinindex
  \edef\gmu@tempa{gmdindexpagecs{\HLPrefix} %
    \kind@fentry{\EntryPrefix}} %
\else
  \edef\gmu@tempa{gmdindexrefcs{\HLPrefix} %
    \kind@fentry{\EntryPrefix}} %
\fi
\edef\gmu@tempa{\IndexPrefix#2\actualchar%
  \quotechar\bslash_\verb+#+\quoted@eschar#+} % The last macro
        in this line usually means the first two, but in some cases it's re-
        defined to be empty (when we use \index@macro to index not
        a CS).
\encapchar\gmu@tempa} %
\@xa\special@index\@xa{\gmu@tempa} % We give the indexing macro
        the argument expanded so that hyperref may see the explicit encap-
        char in order not to add its own encapsulation of \hyperpage when
        the (default) hyperindex=true option is in force. (After this setting
        the \edefs in the above may be changed to \defs.)
} {} % closing of gmd/iexcl/ test.
} }

\un@defentryze 4699 \def\un@defentryze#1{ %
4700   \ifcsname\gmd/defentry/\@xa\detokenize\@xa{\#1}\endcsname
4701   \@xa\g@relaxen\csname\gmd/defentry/\@xa\detokenize\@xa{\#1}%
4702   \endcsname
4703   \fi
4704   \ifx\gmd@detectors\empty
4705     \g@relaxen\last@defmark

```

4705 \fi} \% the last macro (assuming \fi is not a macro :) is only used by \changes.
 If we are in the scope of automatic detection of definitions, we want to
 be able not to use \Define but write \changes after a definition and
 get proper entry. Note that in case of automatic detection of definitions
 \last@defmark's value keeps until the next definition.

```
\un@usgentryze 4712 \def\un@usgentryze#1{%
 4713   \ifcsname_gmd/usgentry/\@xa\detokenize\@xa{\#1}\endcsname
 4714   \@xa\g@relaxen\csname_gmd/usgentry/\@xa\detokenize\@xa{\#1}%
 4715   }\endcsname
 4716 \fi}
 4717 \emptyify\EntryPrefix% this macro seems to be obsolete now (vo.98d).
```

For the case of page-indexing a macro in the commentary when codeline index option is on:

```
\if@pageinindex 4722 \newif\if@pageinindex
\quoted@eschar 4724 \newcommand*\quoted@eschar{\quotecchar\bslash}% we'll redefine it when
               indexing an environment.
```

Let's initialise \IndexPrefix

```
\IndexPrefix 4728 \def\IndexPrefix{}
```

The \IndexPrefix and \HLPrefix ('HyperLabel Prefix') macros are given with account of a possibility of documenting several files in(to) one document. In such case the user may for each file \def\IndexPrefix{\textcolor{red}{<package name>}!} for instance and it will work as main level index entry and \def\HLPrefix{\textcolor{red}{<package name>}} as a prefix in hypertargets in the codelines. They are redefined by \DocInclude e.g.

```
4737 \if@linesnotnum\@pageindextrue\fi
4738 \AtBeginDocument{%
 4739   \if@pageindex
 4740     \def\gmdindexrefcs#1#2#3#4{\csname#2\endcsname{%
 4741       \hyperpage{\#4}}}% in the page case we gobble the third argument
 4742       that is supposed to be the entry prefix.
 4743     \let\gmdindexpagecs=\gmdindexrefcs
 4744   \else
 4745     \def\gmdindexrefcs#1#2#3#4{\gmiiflink[clnum.\#4]{%
 4746       \csname#2\endcsname{\#4}}}
 4747     \def\gmdindexpagecs#1#2#3#4{\hyperlink{page.\#4}{%
 4748       \csname#2\endcsname{\gmd@revprefix{\#3}\#4}}}
 4749   \gmd@revprefix
 4750     \def\gmd@revprefix#1{%
 4751       \def\gmu@tempa{\#1}%
 4752       \ifx\gmu@tempa\empty\p.\,,\fi}
 4753     \def\gmu@tempa{\#1}%
 4754   \fi
 4755 \HLPrefix 4756   \providecommand*\HLPrefix{}% it'll be the hypertargets names' pre-
 4756   fix in multi-docs. Moreover, it showed that if it was empty, hyper-
 4757   ref saw duplicates of the hyper destinations, which was perfectly un-
 4758   derstandable (\codelinenumber{123} made by \refstepcounter and
 4759   \codelinenumber{123} made by \gmhypertarget). But since vo.98 it
 4760   is not a problem anymore because during the automatic \hypertar-
 4761   getting the lines are labelled clnum.\textcolor{red}{<number>}. When \HLPrefix was
 4762   defined as dot, MakeIndex rejected the entries as 'illegal page number'.
 4763   \fi}
```

The definition is postponed till `\begin{document}` because of the `\PageIndex` declaration (added for doc-compatibility), see line 7964.

I design the index to contain hyperlinking numbers whether they are the line numbers or page numbers. In both cases the last parameter is the number, the one before the last is the name of a formatting macro and in line number case the first parameter is a prefix for proper reference in multi-doc.

I take account of three kinds of formatting the numbers: 1. the ‘def’ entry, 2. a ‘usage’ entry, 3. a common entry. As in doc, let them be underlined, italic and upright respectively.

```
\DefEntry 4783 \def \DefEntry#1 {\underline{\#1}}
\UsgEntry 4784 \def \UsgEntry#1 {\textit{\#1}}
```

The third option will be just `\relax` by default:

```
\CommonEntryCmd 4786 \def \CommonEntryCmd {\relax}
```

In line 4659 it’s `\edef`ed to allow an ‘unmöglich’ situation that the user wants to have the common index entries specially formatted. I use this to make *all* the index entries of the driver part to be ‘usage’, see the source of chapter 1928.

Now let’s `\def` the macros declaring a CS to be indexed special way. Each declaration puts the `_ed` name of the macro given it as the argument into proper macro to be `\ifx`ed in lines 4655 and 4657 respectively.

Now we are ready to define a couple of commands. The `*` versions of them are for marking environments and *implicit* CSes.

```
\DefIndex 4802 \outer \def \DefIndex {\begingroup
4803   \MakePrivateLetters
4804   \gm@ifstar
4805   { \@sanitize \MakePrivateOthers %
4806     \Code@DefIndexStar %
4807   \Code@DefIndex }
```



```
\Code@DefIndex 4812 \long \def \Code@DefIndex#1 {\endgroup %
4813   \escapechar \m@ne% because we will compare the macro's name with a string
   without the backslash.
4815   \@defentryze {\#1} {1} }
```



```
\Code@DefIndexStar 4819 \long \def \Code@DefIndexStar#1 %
4820   \endgroup %
4821   \addto@estointer {\#1} %
4822   \@defentryze {\#1} {0} %
4823 }
```



```
\gmd@justadot 4825 \def \gmd@justadot {.}
\@defentryze 4827 \long \def \@defentryze#1#2 %
4828   \xa \glet \csname \gmd@defentry / \detokenize {\#1} \endcsname %
   \gmd@justadot % The
   LATEX \namedef macro could not be used since it's not 'long'. The space
   to sound with the checker.
4832   \ifcat \relax \xa \nx \firstofmany {\#1} \nil
if we meet a CS, then maybe it's a CS to be 'defentryzed' or maybe it's a 'verbatim
special' CS. The only way to distinguish those cases is to assume there shouldn't
be a verbatim containing only a 'verbatim special' CS.
4837   \xa \def \xa \gmu@tempa \xa { \allbutfirstof {\#1} \nil } %
```

```

4838 \ifx\gmu@tempa\empty
4839   \afterfifi{@firstoftwo} if #1 is a single CS, we \xiistring it.
        Otherwise we \detokenize it.
4841 \else\afterfifi{@secondoftwo}
4842   \fi
4843 \else@xa@secondoftwo
4844   \fi
\last@defmark 4845 {\xdef\last@defmark{\xiistring#1}}% we \string the argument just
               in case it's a control sequence. But when it can be a CS, we \defentryze
               in a scope of \escapechar=-1, so there will never be a backslash at the
               beginning of \last@defmark's meaning (unless we \defentryze \\).
4850 {\xdef\last@defmark{\detokenize{#1}}}%
4851 @xa\gdef\csname_gmd/isaCS/\last@defmark\endcsname{#2}% #2 is
               either 0 or 1. It is the information whether this entry is a CS or not.
4854 }% of \defentryze.

@\usgentryze 4856 \long\def@\usgentryze#1{%
4857   @xa\let\csname_gmd/usgentry/\detokenize{#1}\endcsname%
               \gmd@justadot}

      Initialise \envirs@toindex

4860 @emptyify\envirs@toindex

```

Now we'll do the same for the 'usage' entries:

```

\CodeUsgIndex 4863 \outer\def\CodeUsgIndex{\begingroup
4864   \MakePrivateLetters
4865   \gm@ifstar
4866   {@sanitize\MakePrivateOthers%
4867     \Code@UsgIndexStar}%
4868   {\Code@UsgIndex}}

```

The * possibility is for marking environments etc.

```

\Code@UsgIndex 4871 \long\def\Code@UsgIndex#1{%
4872   \endgroup{%
4873     \escapechar\m@ne
4874     \global@\usgentryze{#1}}}

```

```

\Code@UsgIndexStar 4877 \long\def\Code@UsgIndexStar#1{%
4878   \endgroup
4879   {%
4880     \addto@est@index{#1}%
4881     @usgentryze{#1}}%
4882 }

```

For the symmetry, if we want to mark a control sequence or an environment's name to be indexed as a 'normal' entry, let's have:

```

\CodeCommonIndex 4886 \outer\def\CodeCommonIndex{\begingroup
4887   \MakePrivateLetters
4888   \gm@ifstar
4889   {@makeprivateothers@sanitize\Code@CommonIndexStar}%
4890   {\Code@CommonIndex}}

```

```

\Code@CommonIndex 4893 \long\def\Code@CommonIndex#1{\endgroup}

```

```

\Code@CommonIndexStar 4896 \long\def\Code@CommonIndexStar#1{%

```

```
4897 \endgroup\addto@estoindex{\#1}
```

And now let's define commands to index the control sequences and environments occurring in the narrative.

```
\text@indexmacro 4902 \long\def\text@indexmacro#1{%
4903   {\escapechar\m@ne\xdef\macro@pname{\xiistring#1}}%
4905   \@xa\quote@mname\macro@pname\relax% we process the CS's name char
      by char and quote MakeIndex controls. \relax is the iterating macro's
      stopper. The scanned CS's quoted name shall be the expansion of \macro@iname.
4909   \if\verbatimchar\macro@pname
4910     \def\im@firstpar{[$]}%
4911   \else\def\im@firstpar{}%
4912   \fi
4913   {\do@properindex% see line 5332.
4914     \@xa\index@macro\im@firstpar\macro@iname\macro@pname}}
```

The macro defined below (and the next one) are executed only before a $_{12}$ macro's name i.e. a nonempty sequence of $_{12}$ character(s). This sequence is delimited (guarded) by \relax.

```
\quote@mname 4919 \def\quote@mname{%
4920   \def\macro@iname{}%
4921   \quote@charbychar}

\quote@charbychar 4923 \def\quote@charbychar#1{%
4924   \ifx\relax#1% finish quoting when you meet \relax or:
4925   \else
4926     \ifnumo\ifcat\@nx#1\@nx~1\fi\ifcat\@nx#1\relax\@fi>0% we
        can meet active char and/or control sequences (made by) verbatim spe-
        cials, therefore we check whether #1 is an active char and if it is a CS.
4930   \afterfifi% we can meet an active char or a CS iff we use verbatim
        specials.
4932   \ifdef\verbatim@specials@list
4933   \afterfi{%
4934     \begingroup
4935       \escapechar\@xa\@xa\@xa`\'\@xa\@firstofmany%
4936         \verbatim@specials@list\@nil
4937       \@xa\endgroup
4938       \@xa\quote@charbychar\detokenize{\#1}% for a CS \detok-
          enize adds a space but if so, it will be ignored by the ar-
          gument scanner.
4940   }% of \afterfi.
4941   \else\PackageError{gmdoc}{Please report a space-
        bug in
4942     \bslash\quote@charbychar in line 4934}%
4943     \fi% of \ifdef\verbatim@specials@list.
4944   }% of \afterfifi.
4945   \else
4946     \quote@char#1%
4947     \xdef\macro@iname{\macro@iname\gmd@maybequote#1}%
4948     \afterfifi\quote@charbychar
4949     \fi
4950 }
```

The next command will take one argument, which in plain version should be a control sequence and in the starred version also a sequence of chars allowed in environment names or made other by `\MakePrivateOthers` macro, taken in the curly braces.

```

\TextUsgIndex 4956 \def\TextUsgIndex{\begingroup
4957   \MakePrivateLetters
4958   \gm@ifstar{\MakePrivateOthers\Text@UsgIndexStar}{%
4959     \Text@UsgIndex}}
\Text@UsgIndex 4961 \long\def\Text@UsgIndex#1{%
4962   \endgroup\@usgentryze#1%
4963   \text@indexmacro#1}
\Text@UsgIndexStar 4966 \long\def\Text@UsgIndexStar#1{\endgroup\@usgentryze{#1}%
4967   \text@indexenvir{#1}}
\text@indexenvir 4969 \long\def\text@indexenvir#1{%
4970   \verbatim@specials
4971   \edef\macro@pname{\xiistring#1}%
4972   \if\bslash@\xa@\firstofmany\macro@pname\@nil% if \stringed #1
4973     begins with a backslash, we will gobble it to make MakeIndex not see it.
4974     \edef\gmu@tempa{\xa\@gobble\macro@pname}%
4975     \@tempswatrue
4976   \else
4977     \let\gmu@tempa\macro@pname
4978     \@tempswfalset
4979   \fi
4980   \@xa\quote@mname\gmu@tempa\relax% we process \stringed #1 char by
4981   char and quote MakeIndex controls. \relax is the iterating macro's stop-
4982   per. The quoted \stringed #1 shall be the meaning of \macro@iname.
4983   \if@tempswa
4984     \def\quoted@eschar{\quotechar\bslash}%
4985     \else\@emptyify\quoted@eschar\fi% we won't print any backslash be-
4986     fore an environment's name, but we will before a CS's name.
4987     \do@properindex% see line 5332.
4988     \index@macro\macro@iname\macro@pname} }
\TextCommonIndex 4993 \def\TextCommonIndex{\begingroup
4994   \MakePrivateLetters
4995   \gm@ifstar{\MakePrivateOthers\Text@CommonIndexStar}{%
4996     \Text@CommonIndex}}
\Text@CommonIndex 4998 \long\def\Text@CommonIndex#1{\endgroup
5000   \text@indexmacro#1}
\Text@CommonIndexStar 5002 \long\def\Text@CommonIndexStar#1{\endgroup
5003   \text@indexenvir{#1}}

```

As you see in the lines 4666 and 4662, the markers of special formatting are reset after first use.

But we wish the CSes not only to be indexed special way but also to be put in marginpars. So:

```

\CodeMarginize 5010 \outer\def\CodeMarginize{\begingroup
5011   \MakePrivateLetters
5012   \gm@ifstar
5013     {\MakePrivateOthers\egCode@MarginizeEnvir}

```

```

5014      { \egCode@MarginizeMacro} }

One more expansion level because we wish \Code@MarginizeMacro not to begin
with \endgroup because in the subsequent macros it's used after ending the re\cat |
codeing group.

```

```

\egCode@MarginizeMacro 5020 \long\def\egCode@MarginizeMacro#1 {\endgroup
5021   \Code@MarginizeMacro#1}

\Code@MarginizeMacro 5024 \long\def\Code@MarginizeMacro#1 {{% #1 is always a CS.
5027   \escapechar\m@ne
5028   \xa\glet\csname\gmd\zmarpar/\xiistring#1\endcsname%
5029   \gmd@justadot
5030   }}

\egCode@MarginizeEnvir 5033 \long\def\egCode@MarginizeEnvir#1 {\endgroup
5034   \Code@MarginizeEnvir{#1}}

\Code@MarginizeEnvir 5037 \long\def\Code@MarginizeEnvir#1 {\addto\estomarginpar{#1}}

```

And a macro really putting the environment's name in a marginpar shall be triggered at the beginning of the nearest codeline.

Here it is:

```

\mark@envir 5043 \def\mark@envir{%
5044   \ifx\envirs@tomarginpar\empty
5045   \else
5046     \def\do{\Text@Marginize*}%
5047     \envirs@tomarginpar%
5048     \g@emptyify\envirs@tomarginpar%
5049   \fi
5050   \ifx\envirs@toindex\empty
5051   \else
5052     {\verbatim@specials
5053       \gmd@doindexingtext
5054       \envirs@toindex
5055       \g@emptyify\envirs@toindex}%
5056   \fi}

\gmd@doindexingtext 5058 \def\gmd@doindexingtext{%
5059   \def\do##1{%
5060     the \envirs@toindex list contains \stringed macros or environments' names in braces and each preceded with \do. We extract the definition because we use it also in line 4381.
5061     \if\bslash\@firstofmany##1\@nil%
5062       if ##1 begins with a backslash,
5063       we will gobble it for MakeIndex not see it.
5064     \edef\gmd@resa{\@gobble##1}%
5065     \tempswatrue
5066     \else
5067       \edef\gmd@resa{##1}\tempswafalse
5068     \fi
5069     \xa\quote@mname\gmd@resa\relax% see line 4982 & subs. for com-
5070     mmentary.
5071     \if@tempswa
5072       \def\quoted@eschar{\quotetchar\bslash}%
5073       \else\g@emptyify\quoted@eschar
5074     \fi
5075     \index@macro\macro@iname{##1}}%
5076   }
5077 }
```

```
5078 }
```

One very important thing: initialisation of the list macros:

```
5082 \@emptyify\envirs@tomarginpar  
5083 \@emptyify\envirs@toindex
```

For convenience we'll make the 'private letters' first not to bother ourselves with `\makeatletter` for instance when we want mark some CS. And `\MakePrivateOthers` for the environment and other string case.

```
\Define 5090 \outer\def\Define{%
  note that since it's \outer, it doesn't have to be \protected.
  5091 \begingroup
  5092   \MakePrivateLetters
```

We do `\MakePrivateLetters` before `\gm@ifstar` in order to avoid a situation that TeX sees a control sequence with improper name (another CS than we wished) (because `\gm@ifstar` establishes the `\catcodes` for the next token):

```
5098   \gm@ifstar{\@sanitize%
  5099     \Code@DefEnvir}{\Code@DefMacro}
\CodeUsage 5101 \outer\def\CodeUsage{\begingroup
  5102   \MakePrivateLetters
  5103   \gm@ifstar{%
  5104     \@sanitize%
  5105     \MakePrivateOthers
  5106     \Code@UsgEnvir}{\Code@UsgMacro}}
```

And then we launch the macros that close the group and do the work.

```
\Code@DefMacro 5109 \DeclareCommand\Code@DefMacro\long{om}{%
  5110   \Code@DefIndex#2% we use the internal macro; it'll close the group.
  5111   \IfValueTF{\#1}{%
  5112     {\Code@MarginizeMacro\#1}{%
  5113     {\Code@MarginizeMacro\#2}{%
  5114   }}
```

```
\Code@UsgMacro 5118 \DeclareCommand\Code@UsgMacro\long{om}{%
  5119   \Code@UsgIndex#2% here also the internal macro; it'll close the group
  5120   \IfValueTF{\#1}{%
  5121     {\Code@MarginizeMacro\#1}{%
  5122     {\Code@MarginizeMacro\#2}{%
  5124 }}
```

The next macro is taken verbatim ;-) from doc and the subsequent `\lets`, too.

```
\codeline@wrindex 5129 \def\codeline@wrindex#1{\if@filesw
  5130   \immediate\write\@indexfile
  5131   {\string\indexentry{\#1}%
  5132     {\HLPrefix\number\c@codelinenum}}\fi}
```

```
\codeline@glossary 5136 \def\codeline@glossary#1{%
  It doesn't need to establish a group since it is
  always called in a group.
  5138   \if@pageinclistindex
  5139     \edef\gmu@tempa{\gmdindexpagecs{\HLPrefix}{relax}{%
  \EntryPrefix}}%
  5140   \else
```

```

5141   \edef\gmu@tempa{gmdindexrefcs{\HLPrefix}{relax}%
5142     \EntryPrefix}%
5143   relax stands for the formatting command. But
5144   we don't want to do anything special with the change history entries.
5145   \fi
5146   \protected@edef\gmu@tempa{%
5147     \nx\protected@write\nx@glossaryfile{}%
5148     {\string\glossaryentry{\#1\encapchar\gmu@tempa}%
5149      {\HLPrefix\number\c@codelinenum}}%
5150   \gmu@tempa
5151 }

```

We initialise it due to the option (or lack of the option):

```

5156 \AtBeginDocument{%
5157   \if@pageindex
5158     \let\special@index=\index
5159     \let\gmd@glossary=glossary
5160   \else
5161     \let\special@index=\codeline@wrindex
5162     \let\gmd@glossary=codeline@glossary
5163   \fi}%
5164 postponed till \begin{document} with respect of doc-like declarations.

```

And in case we don't want to index:

```

\gag@index 5169 \def\gag@index{\let\index=\@gobble
5170   \let\codeline@wrindex=\@gobble}

```

We'll use it in one more place or two. And we'll wish to be able to undo it so let's copy the original meanings:

```
5176 \StoreMacros{\index\codeline@wrindex}
```

```
\ungag@index 5178 \def\ungag@index{\RestoreMacros{\index\@codeline@wrindex}}
```

Our next task is to define macros that'll mark and index an environment or other string in the code. Because of lack of a backslash, no environment's name is scanned so we have to proceed different way. But we wish the user to have symmetric tools, i.e., the 'def' or 'usage' use of an environment should be declared before the line where the environment occurs. Note the slight difference between these and the commands to declare a CS marking: the latter do not require to be used *immediately* before the line containing the CS to be marked. We separate indexing from marginizing to leave a possibility of doing only one of those things.

```
\Code@DefEnvir 5194 \DeclareCommand\Code@DefEnvir\long{\om}{%
5195   \endgroup
5196   {%
5197     \IfValueTF{\#1}{%
5198       {\addto@estomarginpar{\#1}}%
5199       {\addto@estomarginpar{\#2}}%
5200       \addto@estoindex{\#2}%
5201       \defentryze{\#2}{0}}}}

```

```
\Code@UsgEnvir 5205 \DeclareCommand\Code@UsgEnvir\long{\om}{%
5206   \endgroup
5207   {%
5208     \IfValueTF{\#1}{%
5209       {\addto@estomarginpar{\#1}}%
5210       {\addto@estomarginpar{\#2}}%}
5211 }}
```

```

5211      \addto@estoindex{\#2}%
5212      \@usgentryze{\#2}}}

\addto@estomarginpar 5215 \long\def\addto@estomarginpar#1{%
5220   \gaddtomacro\envirs@tomarginpar{\do{\#1}}}

\addto@estoindex 5222 \long\def\addto@estoindex#1{%
5226   \gaddtomacro\envirs@toindex{\do{\#1}}}

```

And now a command to mark a ‘usage’ occurrence of a CS, environment or another string in the commentary. As the ‘code’ commands this also has plain and starred version, first for CSes appearing explicitly and the latter for the strings and CSes appearing implicitly.

```

\TextUsage 5233 \def\TextUsage{\begingroup
5237   \MakePrivateLetters
5238   \gm@ifstar{\@sanitize\MakePrivateOthers
5240     \Text@UsgEnvir}{\Text@UsgMacro}}

\Text@UsgMacro 5243 \DeclareCommand\Text@UsgMacro\long{\om}{%
5244   \endgroup
5248   \IfValueTF{\#1}{%
5249     {\Text@Marginize*{\#1}{\scanverb*{\#1}}}{%
5250     {\Text@Marginize*{\#2}{\scanverb*{\#2}}}{%
5251       \begingroup\Code@UsgIndex{\#2} we declare the kind of formatting of the
5252         entry.
5252       \text@indexmacro{\#2}}}

\Text@UsgEnvir 5255 \DeclareCommand\Text@UsgEnvir\long{\om}{%
5256   \endgroup
5259   \IfValueTF{\#1}{%
5260     {\Text@Marginize*{\#1}{\scanverb*{\#1}}}{%
5261     {\Text@Marginize*{\#2}{\scanverb*{\#2}}}{%
5262       \@usgentryze{\#2} we declare the ‘usage’ kind of formatting of the entry and
5263         index the sequence \#1.
5264       \text@indexenvir{\#2}}}

```

We don’t provide commands to mark a macro’s or environment’s definition present within the narrative because we think there won’t be any: one defines macros and environments in the code not in the commentary.

```

\TextMarginize 5270 \pdef\TextMarginize{@bsphack\begingroup
5273   \MakePrivateLetters
5274   \gm@ifstar{%
5275     \MakePrivateOthers\egText@MarginizeEnv}{%
5276       \egText@MarginizeCS}}}

\egText@MarginizeEnv 5278 \long\def\egText@MarginizeEnv#1{\endgroup
5279   \Text@Marginize*{\#1}{%
5281     @esphack}

\egText@MarginizeCS 5283 \long\def\egText@MarginizeCS#1{%
5284   \endgroup
5285   \Text@Marginize*{\#1}{%
5286   }}

```

We check whether the margin pars are enabled and proceed respectively in either case.

```

5290 \if@marginparsused
5291   \reversemarginpar
5292   \marginparpush{z}
5293   \marginparwidth8pc\relax

```

You may wish to put not only macros and environments to a marginpar.

```

\gmdmarginpar 5298 \long\def\gmdmarginpar#1{%
5299   \marginpar{\raggedleft\strut
5300     \hskipoptplus1ooptminus1oopt%
5301   #1}}%
5303 \else
\gmdmarginpar 5304 \long\def\gmdmarginpar#1{}%
5305 \fi
5307 \let\gmu@tempa\all@stars
5308 \xa\addtomacro\xat\gmu@tempa\xat{\all@unders}
5309 \xa\DeclareCommand\xat\Text@Marginize\xat!%
5310 \xa{\xa\Q\xat{\gmu@tempa}m}{%
5311   \gmdmarginpar{%
5312     \addtomacro\verb@lasthook{\marginpartt}{%
5313       \IfValueTF{#1}{\scanverb#1}{\scanverb}{#2}}{%
5314 }% of \Text@Marginize.

```

Note that the above command will just gobble its arguments if the marginpars are disabled.

It may be advisable to choose a condensed typewriter font for the marginpars, if there is any. (The Latin Modern font family provides a light condensed typewriter font, it's set in gmdoc class.)

```
5322 \let\marginpartt\narrativett
```

If we print also the narration lines' numbers, then the index entries for CSes and environments marked in the commentary should have codeline numbers not page numbers and that is \let in line 5163. On the other hand, if we don't print narration lines' numbers, then a macro or an environment marked in the commentary should have page number not codeline number. This we declare here, among others we add the letter p before the page number.

```
\do@properindex 5332 \def\do@properindex{%
5333   \if@printalllinenos\else
5334     \pageincliindextrue
5335     \let\special@index=\index
5336   \fi}
```

In doc all the 'working' TeX code should be braced in(to) the macrocode environments. Here another solutions are taken so to be doc-compatible we only should nearly-ignore macrocode[*]s with their Percent and The Four Spaces Preceding ;). I.e., to ensure the line ends are 'queer'. And that the DocStrip directives will be typeset as the DocStrip directives. And that the usual code escape char will be restored at \end{macrocode}. And to add the vertical spaces.

If you know doc conventions, note that gmdoc *does not* require \end{macrocode} to be preceded with any particular number of any char :-).

```
macrocode* 5356 \newenvironment*{macrocode*}{%
5357   \if@codeskipput\else\par\addvspace\CodeTopsep%
   \fi}
```

```
5358 \QueerEOL}%
5359 {\par\addvspace\CodeTopsep\CodeEscapeChar\\}
```

Let's remind that the starred version makes \square visible, which is the default in gmdoc outside macrocode.

So we should make the spaces *invisible* for the unstarred version.

```
macrocode 5367 \newenvironment*{macrocode}{%
 5368   \if@codeskipput\else\par\addvspace\CodeTopsep%
 5369     \codeline{%
 5370       \par\addvspace\CodeTopsep\CodeEscapeChar\\}}
```

Note that at the end of both the above environments the \’s rôle as the code escape char is restored. This is crafted for the `\SpecialEscapechar` macro’s compatibility: this macro influences only the first `macrocode` environment. The situation that the user wants some queer escape char in general and in a particular `macrocode` yet another seems to me “unmöglich, Prinzessin”¹⁰.

Since the first .dtx I tried to compile after the first published version of gmdoc uses a lot of commented out code in `macrocodes`, it seems to me necessary to add a possibility to typeset `macrocodes` as if they were a kind of `verbatim`, that is to leave the code layer and narration layer philosophy.

```
oldmc 5389 \let\oldmc\macrocode
      5390 \let\endoldmc\endmacrocode
oldmc* 5392 \n@melet{\oldmc*}{\macrocode*}
       5393 \n@melet{\endoldmc*}{\endmacrocode*}
```

Now we arm `oldmc` and `olmc*` with the macro looking for `\end{<envir name>}`.

```
5397 \addtomacro{\oldmc}{\@oldmacrocode@launch}%
5398 \expandafter\addtomacro\csname\oldmc*\endcsname{%
5399   \@oldmacrocode@launch}
5402 \def\@oldmacrocode@launch{%
5403   \empty\gmd@textEOL% to disable it in \gmd@docstripdirective launched
                     within the code.
5405   \gmd@ctallsetup
5406   \glet\stored@code@delim\code@delim
5407   \makeother\^\^B\CodeDelim*\^\^B%
5408   \ttverbatim\gmd@DoTeXCodeSpace
5409   \makeother\|\% because \ttverbatim doesn't do that.
5410   \MakePrivateLetters% see line 3681.
5412   \docstrip@percent\@makeother\>%
```

sine qua non of the automatic delimiting is replacing possible \star_{12} in the environment's name with \star_{11} . Not to complicate assume \star may occur at most once and only at the end. We also assume the environment's name consists only of character tokens whose catcodes (except of \star) will be the same in the verbatim text.

```
5419     \@xa\gmd@currenvxistar\@currenvir*\relax
5420     \oldmacrocode}
5421 \foone{\catcode`*11\relax}
5422 {\def\gm@xistar{*}}
5423 \def\gmd@currenvxistar#1*#2\relax{%
```

¹⁰ Richard Strauss after Oscar Wilde, *Salomé*.

```

5426 \edef\currenvir{\#1\if*\#2\gm@xistar\fi}}
The trick is that #2 may be either *12 or empty. If it's *, the test is satisfied and
\if... \fi expands to \gm@xistar. If #2 is empty, the test is also satisfied since
\gm@xistar expands to * but there's nothing to expand to. So, if the environment's
name ends with *12, it'll be substituted with *11 or else nothing will be added. (Note that
a * not at the end of env. name would cause a disaster.)

5436 \foone{%
5437 \catcode`[=1\catcode`]=2
5438 \catcode`\{=\active\@makeother\}
5439 \@makeother\^\^B
5440 \catcode`=/=0\catcode`\=\active
5441 \catcode`\&=14\catcode`\*=11
5442 \catcode`\%=\active\obeyspaces}\&%
5443 [& here the \foone's second pseudo-argument begins

\oldmacrocode
5445 /def/\oldmacrocode[&
5446 /bgroup/let_=relax& to avoid writing /\nx_ four times.
5447 /xdef/\oldmc@def[&
5448 /def/\nx/\oldmc@end####1/\nx%\u\u\u\u/\nx\end&
5449 /nx{/@currenvir} [&
5450 ####1^\^B/\nx/gmd@\oldmcfinis]]&
5451 /egroup& now \oldmc@edef is defined to have one parameter delimited with
      & \end{\textcolor{red}{<current env.'s name>}}
5453 /oldmc@def&
5454 /oldmc@end]&
5455 ]

5457 \def\gmd@\oldmcfinis{%
5458   \def\gmu@tempa{\end{\currenvir}}%
5459   \xa\gmu@tempa\x\def\x\gmd@lastenvir\x\currenvir{%
      \currenvir}%
5460   \xa\CodeDelim\x\*stored@code@delim
5461   \gmd@mchook}% see line 7600

5463 \def\OldMacrocodes{%
5465   \let\macrocode\oldmc
5466   \n@melet{\macrocode*}{\oldmc*}}

```

To handle DocStrip directives in the code (in the old macrocodes case that is).

```

5474 \foone{\catcode`\%\active}
5475 {\def\docstrips@percent{\catcode`\%\active
5476   \let%\gmd@codecheckifds}}

```

The point is, the active % will be expanded when just after it is the \gmd@charbychar cs token and next is some char, the ^^B code delimiter at least. So, if that char is <, we wish to launch DocStrip directive typesetting. (Thanks to \ttverb@im all the < are 'other'.)

```

\gmd@codecheckifds \def\gmd@codecheckifds#1#2{%
  note that #1 is just to gobble \gmd@charbychar
  token.
5477 \if@dsdir\@dsdirgfalse
5478   \if\@nx<\@nx#2\afterfifi\gmd@docstripdirective
5479   \else\afterfifi{\xiipercent#1#2}%
5490   \fi

```

```

5491   \else\afterfi{\xiipercent#1#2}%
5492   \fi}

macro Almost the same we do with the macro[*] environments, stating only their argument to be processed as the 'def' entry. Of course, we should re\catcode it first.

macro 5499 \newenvironment{macro}{%
5500   \tempskipa=\MacroTopsep
5501   \if@codeskipput\advance\tempskipa by-\CodeTopsep\fi
5502   \par\addvspace{\tempskipa}\@codeskipputtrue
5503   \begingroup\MakePrivateLetters\MakePrivateOthers% we make also
      the 'private others' to cover the case of other sequence in the argument.
      (We'll use the \macro macro also in the environment for describing and
      defining environments.)
5507   \gmd@ifoneton\Hybrid@DefMacro\Hybrid@DefEnvir}%
5508   endmacro
5509   {\par\addvspace\MacroTopsep\@codeskipputtrue}

```

It came out that the doc's author(s) give the `macro` environment also starred versions of commands as argument. It's OK since (the default version of) `\MakePrivateLetters` makes `*` a letter and therefore such a starred version is just one CS. However, in `doc.dtx` occur macros that mark *implicit* definitions i.e., such that the defined CS is not scanned in the subsequent code.

`macro*` And for those who want to use this environment for marking implicit definitions, define the star version:

```

5522 \@namedef{macro*}{\let\gmd@ifoneton\@secondoftwo\macro}
5523   endmacro*
5524 \@xa\let\csname\endmacro*\endcsname\endmacro

```

Note that `macro` and `macro*` have the same effect for more-than-one-token arguments thanks to `\gmd@ifoneton`'s meaning inside unstarring `macro` (it checks whether the argument is one-token and if it isn't, `\gmd@ifoneton` switches execution to 'other sequence' path).

The two environments behave different only with a one-token argument: `macro` postpones indexing it till the first scanned occurrence while `macro*` till the first code line met.

Now, let's complete the details. First define an `\if`-like macro that turns true when the string given to it consists of just one token (or one `{<text>}`, to tell the whole truth).

```

\gmd@ifsingle 5542 \def\gmd@ifsingle#1#2\@nil{%
5543   \def\gmu@tempa{#2}%
5544   \ifx\gmu@tempa\empty}

```

Note it expands to an open `\if...` test (unbalanced with `\fi`) so it has to be used as all the `\ifs`, with optional `\else` and obligatory `\fi`. And cannot be used in the possibly skipped branches of other `\if...`s (then it would result with 'extra `\fi`/extra `\else`' errors). But the below usage is safe since both `\gmd@ifsingle` and its `\else` and `\fi` are hidden in a macro (that will not be `\expandafter`ed).

Note also that giving `\gmd@ifsingle` an `\if...` or so as the first token of the argument will not confuse TeX since the first token is just gobbled. The possibility of occurrence of `\if...` or so as a not-first token seems to be negligible.

```

\gmd@ifoneton 5557 \def\gmd@ifoneton#1#2#3{%

```

```

5558 \def \gmu@tempb {\#3} % We hide #3 from TeX in case it's \if... or
      so. \gmu@tempa is used in \gmd@ifsingle.
5560 \gmd@ifsingle{\#3\@nil
5561   \afterfi{\@xa\#1\gmu@tempb}%
5562 \else
5563   \edef\gmu@tempa{\@xa\string\gmu@tempb}%
5564   \afterfi{\@xa\#2\@xa{\gmu@tempa}}%
5565 \fi}

```

Now, define the mysterious `\Hybrid@DefMacro` and `\Hybrid@DefEnvir` macros. They mark their argument with a certain subtlety: they put it in a marginpar at the point where they are and postpone indexing it till the first scanned occurrence or just the first code line met.

```

\Hybrid@DefMacro 5570 \long\def\Hybrid@DefMacro#1{%
5571   \Code@DefIndex{\#1}%
5572   \Text@MarginizeNext{*{\#1}}}

```

```

\Hybrid@DefEnvir 5574 \long\def\Hybrid@DefEnvir#1{%
5575   \Code@DefIndexStar{\#1}%
5576   \Text@MarginizeNext{*{\#1}}}

```

```

\Text@MarginizeNext 5579 \long\def\Text@MarginizeNext#1{%
5580   \gmd@evpaddonce{\Text@Marginize#1\ignorespaces}}

```

The following macro adds its argument to `\everypar` using an auxiliary macro to wrap the stuff in. The auxiliary macro has a self-destructor built in so it `\relaxes` itself after first use.

```

\gmd@evpaddonce 5586 \long\def\gmd@evpaddonce#1{%
5587   \global\advance\gmd@oncenum\@ne
5588   \@xa\long\@xa\edef%
5589     \csname\gmd/evp/NeuroOncer\the\gmd@oncenum\endcsname{%
5590       \@nx\g@relaxen
5591       \csname\gmd/evp/NeuroOncer\the\gmd@oncenum%
5592         \endcsname}%
5593       Why does it work despite it shouldn't? Because
5594       when the CS got with \csname ... \endcsname is undefined, it's
5595       equivalent to \relax and therefore unexpandable. That's why it
5596       passes \edef and is able to be assigned.
5597   \@xa\addtomacro\csname\gmd/evp/NeuroOncer\the\gmd@oncenum%
5598     \endcsname{\#1}%
5599   \@xa\addto@hook\@xa\everypar\@xa{%
5600     \csname\gmd/evp/NeuroOncer\the\gmd@oncenum\endcsname}%
5601 }

```

```
\gmd@oncenum 5601 \newcount\gmd@oncenum
```

`environment` Wrapping a description and definition of an environment in a `macro` environment would look inappropriate ('zgrzytało by' in Polish) although there's no `TeX`nical obstacle to do so. Therefore we define the `environment`, because of æ sthetic and psychological reasons.

```

5612 \@xa\let\@xa\environment\csname\macro*\endcsname
5613 \@xa\let\@xa\endenvironment\csname\endmacro*\endcsname

```

Index exclude list

We want some CSes not to be indexed, e.g., the `LATeX` internals and `TeX` primitives.

doc takes `\index@excludelist` to be a `\toks` register to store the list of expelled CSes. Here we'll deal another way. For each CS to be excluded we'll make (`\let`, to be precise) a control sequence and then we'll be checking if it's undefined (`\ifx`-equivalent to `\relax`).¹¹

```

\DoNotIndex 5628 \def \DoNotIndex {\bgroup \MakePrivateLetters \DoNot@Index}
\DoNot@Index 5636 \long \def \DoNot@Index#1 {\egroup% we close the group,
5637   \let \gmd@iedir \gmd@justadot% we declare the direction of the
      <?>cluding to be excluding. We act this way to be able to reverse the
      exclusions easily later.
5640   \dont@index#1.}

\dont@index 5643 \long \def \dont@index#1 {%
5644   \def \gmu@tempa {\@nx#1}% My TeX Guru's trick to deal with \fi and such,
      i.e., to hide from TeX when it is processing a test's branch without expanding.
5647   \if \gmu@tempa.% a dot finishes expelling
5648   \else
5649     \if \gmu@tempa,% The list this macro is put before may contain commas and
       that's O.K., we just continue the work.
5650     \afterfifi \dont@index
5651   \else% what is else shall off the Index be expelled.
5652     \escapechar \m@ne
5653     \xdef \gmu@tempa {\string#1}% its to sound with \detokenizes
5654     in tests.
5655   \xa \let%
5656   \csname \gmd@iedir / \gmu@tempa \endcsname= \gmd@iedir% In the
5657     default case explained e.g. by the macro's name, the last macro's
     meaning is such that the test in line 4652 will turn false and the subject
     CS shall not be indexed. We \let not \def to spare TeX's memory.
5660   \afterfifi \dont@index
5661   \fi
5662   \fi}

```

Let's now give the exclude list copied ~verbatim ;-) from doc.dtx. I give it in the code layer because I suppose one will document not L^AT_EX source but normal packages.

```

5673 \DoNotIndex{\DoNotIndex}%
  the index entries of these two CSes would be
  rejected by MakeIndex anyway.

```

```

5676 \begin{MakePrivateLetters}%
  Yes, \DoNotIndex does \MakePrivateLetters on its own but No, it won't have any effect if it's given in another macro's
  \def.

```

```

DefaultIndexExclusions 5680 \gdef \DefaultIndexExclusions {%
5681   \DoNotIndex {\@ \@@par \begin{parpenalty} \empty }%
5682   \DoNotIndex {\@flushglue \gobble \input }%
5683   \DoNotIndex {\makefnmark \makeother \maketitle }%
5684   \DoNotIndex {\namedef \ne \spaces \tempa }%
5685   \DoNotIndex {\tempb \tempswafalse \tempswatrue }%
5686   \DoNotIndex {\thanks \thefnmark \topnum }%
5687   \DoNotIndex {\@ \elt \forloop \fortmp \gtempa
      \totalleftmargin }%
5688   \DoNotIndex {" \ / \ifundefined \nil \verbatim
      \vobeyspaces }%

```

¹¹ This idea comes from Marcin Woliński.

```

5689 \DoNotIndex{\| \~\active \advance \aftergroup \begingroup
5690   \bgroup}%
5691 \DoNotIndex{\mathcal \csname \def \documentstyle \dospecials
5692   \edef}%
5693 \DoNotIndex{\egroup}%
5694 \DoNotIndex{\else \endcsname \endgroup \endinput
5695   \endtrivlist}%
5696 \DoNotIndex{\expandafter \fi \fnsymbol \futurelet \gdef
5697   \global}%
5698 \DoNotIndex{\hbox \hss \if \if@inlabel \if@tempswa
5699   \if@twocolumn}%
5700 \DoNotIndex{\ifcase}%
5701 \DoNotIndex{\ifcat \iffalse \ifx \ignorespaces \index \input
5702   \item}%
5703 \DoNotIndex{\jobname \kern \leavevmode \leftskip \let \llap
5704   \lower}%
5705 \DoNotIndex{\m@ne \next \newpage \nobreak \noexpand
5706   \nonfrenchspacing}%
5707 \DoNotIndex{\obeylines \or \protect \raggedleft \rightskip
5708   \rm \sc}%
5709 \DoNotIndex{\setbox \setcounter \small \space \string
5710   \strut}%
5711 \DoNotIndex{\strutbox}%
5712 \DoNotIndex{\thefootnote \thispagestyle \topmargin
5713   \trivlist \tt}%
5714 \DoNotIndex{\twocolumn \typeout \vss \vtop \xdef \z@}%
5715 \DoNotIndex{\, \@bsphack \@esphack \@noligs \@vobeyspaces
5716   \@xverbatim}%
5717 \DoNotIndex{\` \catcode \end \escapechar \frenchspacing
5718   \glossary}%
5719 \DoNotIndex{\hangindent \hfil \hfill \hskip \hspace \ht \it
5720   \langle}%
5721 \DoNotIndex{\leaders \long \makelabel \marginpar \markboth
5722   \mathcode}%
5723 \DoNotIndex{\mathsurround \mbox} % \newcount \newdimen \newskip
5724 \DoNotIndex{\nopagebreak}%
5725 \DoNotIndex{\parfillskip \parindent \parskip \penalty \raise
5726   \rangle}%
5727 \DoNotIndex{\section \setlength \TeX \topsep \underline
5728   \unskip}%
5729 \DoNotIndex{\vskip \vspace \widetilde \\ \% \@date
5730   \@defpar}%
5731 \DoNotIndex{\[ \]}% see line 5673.
5732 \DoNotIndex{\count@ \ifnum \loop \today \uppercase
5733   \uccode}%
5734 \DoNotIndex{\baselineskip \begin \tw@}%
5735 \DoNotIndex{\a \b \c \d \e \f \g \h \i \j \k \l \m \n \o \p \q}%
5736 \DoNotIndex{\r \s \t \u \v \w \x \y \z \A \B \C \D \E \F \G \H}%
5737 \DoNotIndex{\I \J \K \L \M \N \O \P \Q \R \S \T \U \V \W \X \Y
5738   \Z}%
5739 \DoNotIndex{\_1 \_2 \_3 \_4 \_5 \_6 \_7 \_8 \_9 \_o}%
5740 \DoNotIndex{\_! \# \$_ \& \_ \(\) . \_ \: \; \_ \< \= \_ \? \_}%
5741   \+ seems
5742   to be so rarely used that it may be advisable to index it.

```

```

5722 \DoNotIndex{\discretionary \immediate \makeatletter
5723   \makeatother}%
5724 \DoNotIndex{\meaning \newenvironment \par \relax
5725   \renewenvironment}%
5726 \DoNotIndex{\repeat \scriptsize \selectfont \the
5727   \undefined}%
5728 \DoNotIndex{\arabic \do \makeindex \null \number \show \write
5729   \@ehc}%
5730 \DoNotIndex{\@author \@ehc \@ifstar \@sanitize \@title}%
5731 \DoNotIndex{\if@minipage \if@restonecol \ifeof \ifmmode}%
5732 \DoNotIndex{\lccode \% \% \newtoks
5733   \onecolumn \openin \p@ \SelfDocumenting}%
5734 \DoNotIndex{\settowidth \resetonecoltrue
5735   \resetonecolfalse \bf}%
5736 \DoNotIndex{\clearpage \closein \lowercase
5737   \@inlabelfalse}%
5738 \DoNotIndex{\selectfont \mathcode \newmathalphabet
5739   \rmdefault}%
5740 \DoNotIndex{\bfdefault}%

```

From the above list I removed some `\new...` declarations because I think it may be useful to see gathered the special `\new...`s of each kind. For the same reason I would not recommend excluding from the index such declarations as `\AtBeginDocument`, `\AtEndDocument`, `\AtEndOfPackage`, `\DeclareOption`, `\DeclareRobustCommand` etc. But the common definitions, such as `\(new|provide)command` and `\(e|g|x)defs`, as the most common, in my opinion excluded should be.

And some my exclusions:

```

5746 \DoNotIndex{\@@input \@auxout \@currentlabel \@dblarg}%
5747 \DoNotIndex{\@ifdefinable \@ifnextchar \@ifpackageloaded}%
5748 \DoNotIndex{\@indexfile \@let@token \@sptoken \^}%
  the latter comes
  from CSes like \^\^M, see sec. 8228.
5749 \DoNotIndex{\addto@hook \addvspace}%
5750 \DoNotIndex{\CurrentOption}%
5751 \DoNotIndex{\emph \empty \firstofone}%
5752 \DoNotIndex{\font \fontdimen \hangindent \hangafter}%
5753 \DoNotIndex{\hyperpage \hyperlink \hypertarget}%
5754 \DoNotIndex{\ifdim \ifhmode \iftrue \ifvmode
5755   \medskipamount}%
5756 \DoNotIndex{\message}%
5757 \DoNotIndex{\NeedsTeXFormat \newcommand \newif}%
5758 \DoNotIndex{\newlabel}%
5759 \DoNotIndex{\of}%
5760 \DoNotIndex{\phantom \ProcessOptions \protected@edef}%
5761 \DoNotIndex{\protected@xdef \protected@write}%
5762 \DoNotIndex{\ProvidesPackage \providecommand}%
5763 \DoNotIndex{\raggedright}%
5764 \DoNotIndex{\raisebox \refstepcounter \ref \rlap}%
5765 \DoNotIndex{\reserved@a \reserved@b \reserved@c
5766   \reserved@d}%
5767 \DoNotIndex{\stepcounter \subsection \textit \textsf
5768   \thepage \tiny}%
5769 \DoNotIndex{\copyright \footnote \label \LaTeX}%

```

```

5771 \DoNotIndex{\@eha \@endparenv \if@endpe \@endpefalse
5772   \@endptrue}%
5773 \DoNotIndex{\@evenfoot \@oddfoot \firstoftwo
5774   \secondoftwo}%
5775 \DoNotIndex{\@for \gobbletwo \@idxitem \ifclassloaded}%
5776 \DoNotIndex{\@ignorefalse \ignoretrue \ifignore}%
5777 \DoNotIndex{\@input@ \input}%
5778 \DoNotIndex{\@latex@error \mainaux \nameuse}%
5779 \DoNotIndex{\@nomath \@oddfoot}%
5780   \%@\onlypreamble should be in-
5781   dexed IMHO.
5782 \DoNotIndex{\@outerparskip \partaux \partlist \plus}%
5783 \DoNotIndex{\@sverb \sxverbatim}%
5784 \DoNotIndex{\@tempcnta \tempcntb \tempskipa \tempskipb}%
5785   I think the layout parameters even the kernel, should not be excluded:
5786   \%@\topsep \topsepadd \abovedisplayskip \clubpenalty etc.
5787 \DoNotIndex{\@writeckpt}%
5788 \DoNotIndex{\bfseries \chapter \part \section \subsection}%
5789 \DoNotIndex{\subsubsection}%
5790 \DoNotIndex{\char \check@mathfonts \closeout}%
5791 \DoNotIndex{\fontsize \footnotemark \footnotetext
5792   \footnotesize}%
5793 \DoNotIndex{\g@addto@macro \hfilneg \Huge \huge}%
5794 \DoNotIndex{\hyphenchar \if@partsw \IfFileExists }%
5795 \DoNotIndex{\include \includeonly \indexspace}%
5796 \DoNotIndex{\itshape \language \LARGE \Large \large}%
5797 \DoNotIndex{\lastbox \lastskip \m@th \makeglossary}%
5798 \DoNotIndex{\maketitle \math@fontsfalse \math@fontstrue
5799   \mathsf}%
5800 \DoNotIndex{\MessageBreak \noindent \normalfont
5801   \normalsize}%
5802 \DoNotIndex{\on@line \openout \outer}%
5803 \DoNotIndex{\parbox \part \rmfamily \rule \sbox}%
5804 \DoNotIndex{\sf@size \sffamily \skip}%
5805 \DoNotIndex{\textsc \textup \toks@ \ttfamily \vbox}%
5806 \% \DoNotIndex{\begin*} maybe in the future, if the idea gets popular...
5807 \DoNotIndex{\hspace* \newcommand* \newenvironment*
5808   \providecommand*}%
5809 \DoNotIndex{\renewenvironment* \section* \chapter*}%
5810 }% of \DefaultIndexExclusions.

```

I put all the expellings into a macro because I want them to be optional.

```
5811 \end{MakePrivateLetters}
```

And we execute it due to the (lack of) counter-corresponding option:

```

5815 \if@indexallmacros\else
5816   \DefaultIndexExclusions
5817 \fi

```

If we expelled so many CSes, someone may like it in general but he/she may need one or two expelled to be indexed back. So

```

\DoIndex 5823 \def\DoIndex{\bgroup\MakePrivateLetters\Do@Index}
\Do@Index 5830 \long\def\Do@Index#1{\egroup\relaxen\gmd@iedir\dont@index#1.}% note

```

we only redefine an auxiliary CS and launch also \dont@index inner macro.

And if a user wants here make default exclusions and there do not make them, they may use the \DefaultIndexExclusions declaration themself. This declaration OCSR, but anyway let's provide the counterpart. It OCSR, too.

```
DefaultIndexExclusions 5839 \def\UndoDefaultIndexExclusions{%
5840   \StoreMacro\DoNotIndex
5842   \let\DoNotIndex\DoIndex
5844   \DefaultIndexExclusions
5846   \RestoreMacro\DoNotIndex}
```

Index parameters

"The \IndexPrologue macro is used to place a short message into the document above the index. It is implemented by redefining \index@prologue, a macro which holds the default text. We'd better make it a \long macro to allow \par commands in its argument."

```
\IndexPrologue 5858 \long\def\IndexPrologue#1{\@bsphack\def\index@prologue{#1}%
\index@prologue      \@esphack}

\indexdiv 5861 \def\indexdiv{\@ifundefined{chapter}{\section*}{\chapter*}{}}

\index@prologue 5865 \@ifundefined{\index@prologue}{\def\index@prologue{%
  \indexdiv{Index}%
  \markboth{Index}{Index}%
  Numbers written in italic refer to the \if@pageindex%
  pages \else
  code lines \fi where the
  corresponding entry is described; numbers underlined%
  refer to the
  \if@pageindex\else code line of the \fi definition;%
  numbers in
  roman refer to the \if@pageindex pages \else code lines %
  \fi where
  the entry is used.
  \if@pageindex\else
    \ifx\HLPrefix\empty
      The numbers preceded with 'p.' are page numbers.
    \else
      The numbers with no prefix are page numbers.
    \fi\fi
  \ifx\IndexLinksBlack\relax\else
    All the numbers are hyperlinks.
  \fi
  \gmd@dip@hook% this hook is intended to let a user add something without
  redefining the entire prologue, see below.
  }}{}}

5885 }
```

During the preparation of this package for publishing I needed only to add something at the end of the default index prologue. So

```
\AtDIPrologue 5890 \@emptyify\gmd@dip@hook
5891 \long\def\AtDIPrologue#1{\g@addto@macro\gmd@dip@hook{#1}}
```

Now we can rollback the \ampulexdef made to \verb:

```
5895 \AtDIPrologue{%
```

```

5896 \ampulexdef\verb\ttverbatim{\verb+verbatim@specials}
5897 {\ttverbatim\verb+verbatim@specials+}}


The Author(s) of doc assume multicol is known not to everybody. My assumption is the other so


5903 \RequirePackage{multicol}


"If multicol is in use, when the index is started we compute the remaining space on the current page; if it is greater than \IndexMin, the first part of the index will then be placed in the available space. The number of columns set is controlled by the counter \c@IndexColumns which can be changed with a \setcounter declaration."


\IndexMin 5912 \newdimen\IndexMin\IndexMin=133pt\relax% originally it was set 80 pt,
           but with my default prologue there's at least 4.7 cm needed to place the pro-
          logue and some index entries on the same page.
\c@IndexColumns 5915 \newcount\c@IndexColumns\c@IndexColumns=3
\theindex 5916 \renewenvironment{\theindex}{%
5917 {\begin{multicols}\c@IndexColumns[\index@prologue][%
           \IndexMin]%
           \IndexLinksBlack
           \IndexParms\let\item@\idxitem\ignorespaces}%
5920 {\end{multicols}}}
\IndexLinksBlack 5922 \def\IndexLinksBlack{\hypersetup{linkcolor=black}}% To make Adobe
                  Reader work faster.
\IndexParms 5925 \@ifundefined{\IndexParms}{%
5926 {\def\IndexParms{%
           \parindent\z@
           \columnsep15pt
           \parskip\opt\plus\onept
           \rightskip\z@
           \mathsurround\z@
           \parfillskip=-15pt\plus\onefil}% doc defines this parameter rigid
           but that's because of the stretchable space (more precisely, a \dotfill)
           between the item and the entries. But in gmdoc we define no
           such special delimiters, so we add an infinite stretch.
5928 \small
5929 \def\@idxitem{\par\hangindent\z@}
5930 \def\subitem{\@idxitem\hspace*{15pt}}%
5931 \def\subsubitem{\@idxitem\hspace*{25pt}}%
5932 \def\indexspace{\par\vspace{10pt}\plus\z@minus\z@}%
5933 \ifx\EntryPrefix\empty\else\raggedright\fi% long (actually,
           a quite short but nonempty entry prefix) made space stretches so terri-
           bly large in the justified paragraphs that we should make \raggedright
           rather.
5934 \ifnum\c@IndexColumns>\tw@\raggedright\fi% the numbers in
           narrow columns look better when they are \raggedright in my
           opinion.
5935 } } } }
\PrintIndex 5951 \def\PrintIndex{%
           we ensure the standard meaning of the line end character
           not to cause a disaster.
5953 \@ifQueerEOL{\StraightEOL\printindex\QueerEOL}%

```

```
5954 { \printindex }
```

Remember that if you want to change not all the parameters, you don't have to redefine the entire `\IndexParms` macro but you may use a very nice L^AT_EX command `\g@addto@macro` (it has `\global` effect, also with an apeless name (`\gaddtomacro`) provided by `gutils`. (It adds its second argument at the end of definition of its first argument provided the first argument is a no-argument macro.) Moreover, `gutils` provides also `\addtomacro` that has the same effect except it's not `\global`.

The DocStrip directives

```
6026 \foone {\@makeother\<\@makeother\>
6027   \glet \sgtleftxii=<}
6028 {
6029   \def \gmd@docstripdirective {%
6030     \begingroup \let \do=\@makeother
6031     \do\*\do\ /\do\+\do\-\do\,, \do\&\do\|\do\!\do\(\do\)\do\>%
6032       \do\<%
6033     \@ifnextchar {<} {%
6034       \let \do=\@makeother \dospecials
6035       \gmd@docstripverb}
6036     { \gmd@docstripinner }{%
6037   }
6038   \def \gmd@docstripinner#1> {%
6039     \endgroup
6040   }
6041   \def \gmd@modulehashone {%
6042     \Module{#1}\space
6043     \@afternarrgfalse \@aftercodegrue \@codeskipputgfalse }%
6044   \gmd@textEOL \gmd@modulehashone}
```

A word of explanation: first of all, we close the group for changed `\catcodes`; the directive's text has its `\catcodes` fixed. Then we put the directive's text wrapped with the formatting macro into one macro in order to give just one token the `gmdoc`'s T_EX code scanner. Then launch this big T_EX code scanning machinery by calling `\gmd@textEOL` which is an alias for the 'narrative' meaning of the line end. This macro opens the verbatim group and launches the char-by-char scanner. That is this scanner because of what we encapsulated the directive's text with the formatting into one macro: to let it pass the scanner.

That's why in the 'old' macrocodes case the active `%` closes the group before launching `\gmd@docstripdirective`.

The 'verbatim' directive macro works very similarly.

```
6068 }
6069 \foone {\@makeother\<\@makeother\>
6070   \glet \sgtleftxii=<
6071   \catcode`\\^M=\active}{%
6072 }
6073 {
6074   \def \gmd@docstripverb<#1^M{%
6075     \endgroup%
6076     \def \gmd@modulehashone {%
6077       \ModuleVerb{#1} \@afternarrgfalse \@aftercodegrue %
6078       \@codeskipputgfalse }%
6079     \gmd@docstripshook%
6080     \gmd@textEOL \gmd@modulehashone^M }%
```

```

6081 }

(~Verbatim ;-) from doc:)

\Module 6084 \providecommand*\Module[1] {{\mod@math@codes$\langle\mathsf{\%}
#1}\rangle\$}}
\ModuleVerb 6086 \providecommand*\ModuleVerb[1] {{\mod@math@codes$\langle\mathsf{\%
\langle\mathsf{\#1}\$}}\$}}
\mod@math@codes 6088 \def\mod@math@codes {\mathcode`\\ |="226A_\mathcode`\\ &="2026_}

```

The changes history

The contents of this section was copied ~verbatim from the doc's documentation, with only smallest necessary changes. Then my additions were added :-)).

"To provide a change history log, the \changes command has been introduced. This takes [one optional and] three [mandatory] arguments, respectively, [the macro that'll become the entry's second level,] the version number of the file, the date of the change, and some detail regarding what change has been made [i.e., the description of the change]. The [second] of these arguments is otherwise ignored, but the others are written out and may be used to generate a history of changes, to be printed at the end of the document. [... I omit an obsolete remark about then-older MakeIndex's versions.]

The output of the \changes command goes into the [⟨Glossary File⟩](#) and therefore uses the normal \glossaryentry commands. Thus MakeIndex or a similar program can be used to process the output into a sorted "glossary". The \changes command commences by taking the usual measures to hide its spacing, and then redefines \protect for use within the argument of the generated \indexentry command. We re-code nearly all chars found in \@sanitize to letter since the use of special package which make some characters active might upset the \changes command when writing its entries to the file. However we have to leave % as comment and _ as [⟨space⟩](#) otherwise chaos will happen. And, of course the \ should be available as escape character."

We put the definition inside a macro that will be executed by (the first use of \RecordChanges. And we provide the default definition of \changes as a macro just gobbling its arguments. We do this to provide no changes' writing out if \RecordChanges is not used.

```

\gmd@DefineChanges 6134 \def\gmd@DefineChanges {%
\changes 6135   \outer\long\def\changes {%
6136     \gmd@changes@init
6137     \changes@{}}

\gmd@changes@init 6139 \def\gmd@changes@init {%
6140   \@bsphack\begin{group}\@sanitize
6141   \catcode`\\z@\catcode`\_10_\MakePercentIgnore
6142   \MakePrivateLetters_\StraightEOL
6143   \MakeGlossaryControls}

\changes 6145 \newcommand\changes[4][] {\PackageWarningNoLine{gmdoc} {%
6146   ^^JThe_\bslash_changes_command_used_on@line
6147   ^^Jwith_no_\string\RecordChanges_space_declared.
6148   ^^JI_shall_not_warn_you_again_about_it}%
\changes 6150   \renewcommand\changes[4][] {%
6151   } }

\MakeGlossaryControls 6153 \def\MakeGlossaryControls {%
6154   \edef\actualchar{\string=}\edef\quotechar{\string!}%

```

```

6155  \edef\levelchar{\string>} \edef\encapchar{\xiiclus} % for the glossary the 'actual', the 'quote' and the 'level' chars are respectively =, ! and >, the 'encap' char remains untouched. I decided to preserve the doc's settings for the compatibility.

\changes@ 6161 \newcommand\changes@[4][\generalname]{%
6164   \if@RecentChange{#3}%
6165     if the date is later than the one stored in \c@Chang %
6166     % esStartDate,
6167     \@tempswafalse
6168     \ifx\generalname#1% then we check whether a CS-entry is given in the
6169       optional first argument or is it unchanged.
6170     \ifx\last@defmark\relax\else%
6171       if no particular CS is specified in
6172         #1, we check whether \last@defmark contains something and if
6173         so, we put it into \gmu@tempb scratch macro.
6174     \@tempswatrue
6175     \edef\gmu@tempb{%
6176       it's a bug fix: while typesetting traditional .
6177       dtxes, \last@defmark came out with \ at the beginning (which
6178       resulted with \\<name> in the change log) but while typesetting
6179       the 'new' way, it occurred without the bslash. So we gobble the
6180       bslash if it's present and two lines below we handle the exception
6181       of \last@defmark = {} (what would happen if a definition
6182       of \\ was marked in new way gmdocing).
6183     \if\bslash\last@defmark\else\last@defmark\fi%
6184     \ifx\last@defmark\bslash\let\gmu@tempb\last@defmark%
6185       \fi%
6186     \n@melet{gmd@glossCStest}{gmd/isaCS/\last@defmark}%
6187     \fi
6188   \else% the first argument isx not \generalname i.e., a particular CS is spec-
6189     ified by it (if some day one wishes to \changes \generalname, they
6190     should type \changes[\generalname] ...)
6191   \@tempswatrue
6192   {\escapechar\m@ne
6193     \xdef\gmu@tempb{\string#1}%
6194   \if\bslash\@xa\@firstofmany\string#1\relax\@nil%
6195     we
6196     check whether #1 is a CS...
6197     \def\gmd@glossCStest{1}%
6198     ... and tell the glossary if so.
6199     \fi
6200   \fi
6201   \@ifundefined{gmd@glossCStest}{\def\gmd@glossCStest{o}}{%
6202   }%
6203   \protected@edef\gmu@tempa{\@nx\gmd@glossary{%
6204     \if\relax\GeneralName\relax\else
6205       \GeneralName%
6206       it's for the \DocInclude case to precede every
6207       \changes of the same file with the file name, cf. line 6696.
6208     \fi
6209     #2\levelchar%
6210     \if@tempswa%
6211       If the macro \last@defmark doesn't contain any CS
6212       name (i.e., is empty) nor #1 specifies a CS, the current changes
6213       entry was done at top-level. In this case we precede it by \gen-
6214       eralname.
6215       \gmu@tempb
6216       \actualchar\bslash_\verb+%
6217       \if\verbatimchar\gmu@tempb$%
6218         \else\verbatimchar\fi

```

```

6213          \if\gmd@glossCStest\quotechar\bslash\fi%
6214          \gmu@tempb
6215          \if\verbatimchar\gmu@tempb$\else\verbatimchar\fi
6216          \else
6217          \space\actualchar\generalname
6218          \fi
6219          :\levelchar%
6220          #4%
6221          } }%
6222          \gmu@tempa
6223          \grelaxen\gmd@glossCStest
6224          \fi% of \if@recentchange
6225          \endgroup\@esphack}

```

Let's initialise `\last@defmark` and `\GeneralName`.

```

6228 \@relaxen\last@defmark
6229 \@emptyify\GeneralName

```

`\ChangesGeneral` 6231 `\def\ChangesGeneral{\grelaxen\last@defmark}%` If automatic detection of definitions is on, the default entry of `\changes` is the meaning of `\last@defmark`, the last detected definiendum that is. The declaration defined here serves to start a scope of 'general' `\changes` entries.

```

6237 \AtBeginInput{\ChangesGeneral}

```

Let's explain `\if@RecentChange`. We wish to check whether the change's date is later than date declared (if any limit date *was* declared). First of all, let's establish a counter to store the declared date. The untouched counters are equal so if no date is declared there'll be no problem. The date will have the `<YYYYMMDD>` shape both to be easily compared and readable.

```

\c@ChangesStartDate 6245 \newcount\c@ChangesStartDate
\if@RecentChange 6248 \def\if@RecentChange#1{%
6249   \gmd@setChDate#1\@nil\@tempcnta
6250   \ifnum\@tempcnta>\c@ChangesStartDate}
\gmd@setChDate 6252 \def\gmd@setChDate#1/#2/#3\@nil#4{%
  the last parameter will be a \count register.
6254   #4=#1\relax
6255   \multiply#4\by\@M
6256   \count8=#2\relax% I know it's a bit messy not to check whether the #4
   \count is \count8 but I know this macro will only be used with \counto
   (\@tempcnta) and some higher (not a scratch) one.
6258   \multiply\count8\by100%
6259   \advance#4\by\count8\count8=\z@
6260   \advance#4\by#3\relax}

```

Having the test defined, let's define the command setting the date counter. `#1` is to be the version and `#2` the date `{<year>/<month>/<day>}`.

```

\ChangesStart 6268 \def\ChangesStart#1#2{%
6269   \gmd@setChDate#2\@nil\c@ChangesStartDate
6270   \typeout{^^JPackage\gmdoc\info: ^^JChanges' start date #1%
   memorised
6272   as\string<\the\c@ChangesStartDate\string>\on@line.^^J}

```

```

6274 \advance\c@ChangesStartDate\m@ne% we shall show the changes at the
6275 specified day and later.
6276 \ifnum\c@ChangesStartDate>19820900 $\relax$ 12 see below.
6277 \edef\gmu@tempa{%
6278   \nx\g@addto@macro\nx\glossary@prologue{%
6279     The\changes
6280     \if\relax\GeneralName\relax\else\of\GeneralName%
6281       \space\fi
6282     earlier\than
6283     #1\if\relax#1\relax\#2\else(#2)\fi\space\are\not%
6284     shown.} }%
6285   \gmu@tempa
6286 }
6287 
```

(Explanation to line 6276.) My *TEX Guru* has remarked that the change history tool should be used for documenting the changes that may be significant for the users not only for the author and talking of what may be significant to the user, no changes should be hidden since the first published version. However, the changes' start date may be used to provide hiding the author's 'personal' notes: they should only date the 'public' changes with the four digit year and the 'personal' ones with two digit year and set \ChangesStart {} {1000/0/0} or so.

In line 6276 I establish a test value that corresponds to a date earlier than any *TEX* stuff and is not too small (early) to ensure that hiding the two digit year changes shall not be mentioned in the changes prologue.

"The entries [of a given version number] are sorted for convenience by the name of [the macro explicitly specified as the first argument or] the most recently introduced macro name (i.e., that in the most recent \begin{macro} command [or \Define]). We therefore provide [\last@defmark] to record that argument, and provide a default definition in case \changes is used outside a *macro* environment. (This is a wicked hack to get such entries at the beginning of the sorted list! It works providing no macro names start with ! or ".)

This macro holds the string placed before changes entries on top-level."

```
\generalname 6325 \def\generalname{General}
```

"To cause the changes to be written (to a .glo) file, we define \RecordChanges to invoke LATEX's usual \makeglossary command."

I add to it also the \writeing definition of the \changes macro to ensure no changes are written out without \RecordChanges.

```
\RecordChanges 6337 \def\RecordChanges{\makeglossary\gmd@DefineChanges
6338   \relaxen\RecordChanges}
```

"The remaining macros are all analogues of those used for the theindex environment. When the glossary is started we compute the space which remains at the bottom of the current page; if this is greater than \GlossaryMin then the first part of the glossary will be placed in the available space. The number of columns set [is] controlled by the counter \c@GlossaryColumns which can be changed with a \setcounter declaration."

```
\GlossaryMin 6350 \newdimen\GlossaryMin           \GlossaryMin      = 8pt
              c@GlossaryColumns
```

```
\c@GlossaryColumns 6352 \newcount\c@GlossaryColumns \c@GlossaryColumns = 2
```

¹² DEK writes in *TEX, The Program* of September 1982 as the date of *TEX* Version 0.

“The environment `theglossary` is defined in the same manner as the `theindex` environment.”

```
theglossary 6358 \newenvironment{theglossary}{%
6360   \begin{multicols}{\c@GlossaryColumns}
6361   [\glossary@prologue][\GlossaryMin]{%
6362     \GlossaryParms\IndexLinksBlack
6363     \let\item\@idxitem\ignorespaces}%
6364   \end{multicols}}
```

Here is the `MakeIndex` style definition:

```
6369 </ package>
6370 <+gmglo> preamble
6371 <+gmglo> "\n\begin{theglossary}\n"
6372 <+gmglo> \\makeatletter\n"
6373 <+gmglo> postamble
6374 <+gmglo> "\n\n\end{theglossary}\n"
6375 <+gmglo> keyword\glossaryentry"
6376 <+gmglo> actual='
6377 <+gmglo> quote'!'
6378 <+gmglo> level'>'
```

6379 <+gmglo>

The `MakeIndex` shell command for the glossary should look as follows:

`makeindex -r -s gmglo.ist -o <myfile>.gls <myfile>.glo`

where `-r` commands `MakeIndex` not to make implicit page ranges, `-s` commands `MakeIndex` to use the style stated next not the default settings and the `-o` option with the subsequent filename defines the name of the output.

“The `\GlossaryPrologue` macro is used to place a short message above the glossary into the document. It is implemented by redefining `\glossary@prologue`, a macro which holds the default text. We better make it a long macro to allow `\par` commands in its argument.”

```
\GlossaryPrologue 6398 \long\def\GlossaryPrologue#1{\@bsphack
\glossary@prologue 6399 \def\glossary@prologue{#1}%
6400 \@esphack}
```

“Now we test whether the default is already defined by another package file. If not we define it.”

```
\glossary@prologue 6405 \@ifundefined{glossary@prologue}
6406   {\def\glossary@prologue{\indexdiv{{Change_History}}\%
6407     \markboth{{Change_History}}{{Change_History}}\%
6408   }{}}
```

“Unless the user specifies otherwise, we set the change history using the same parameters as for the index.”

```
\GlossaryParms 6412 \AtBeginDocument{%
6413   \@ifundefined{GlossaryParms}{\let\GlossaryParms%
\IndexParms}{}{}}
```

“To read in and print the sorted change history, just put the `\PrintChanges` command as the last (commented-out, and thus executed during the documentation pass through the file) command in your package file. Alternatively, this command may form one of the arguments of the `\StopEventually` command, although a change history

is probably not required if only the description is being printed. The command assumes that MakeIndex or some other program has processed the .glo file to generate a sorted .gls file."

```
\PrintChanges 6425 \def\PrintChanges {%
  to avoid a disaster among queer EOLs:
  6426   \ifQueerEOL
  6427     {\StraightEOL\@input{\jobname.gls}\QueerEOL}%
  6428     {\@input{\jobname.gls}}%
  6429   \g@emptyify\PrintChanges}

\toCTAN 6431 \pdef\toCTAN {%
  % #1 <year/month/day>,<version number>
  6432   \gmd@changes@init
  6433   \gmd@toCTAN@}

\gmd@toCTAN@ 6441 \def\gmd@toCTAN@#1 {%
  6442   \edef\gmu@tempa {\gmd@chgs@parse#1\@nil}%
  6443   \edef\gmu@tempa {%
    \unexpanded{\changes@[\generalname]}%
    {\@xa\@firstofthree\gmu@tempa}%
    {\@xa\@secondofthree\gmu@tempa}%
    {put_to_\acro{CTAN}_on_\@xa\@secondofthree\gmu@tempa}}%
  6444   \gmu@tempa}
```

To make writing changes easier, to allow copying the date & version string from the \ProvidesPackage/Class optional argument.

```
\chgs 6453 \outer\pdef\chgs {\gmd@changes@init\gmd@chgs}

\gmd@chgs 6456 \DeclareCommand\gmd@chgs{o>!m}{%
  \IfValueTF{#1}{%
    \edef\gmu@tempa {\@nx\changes@[\unexpanded{#1}]%
      \@xa\unexpanded\@xa{\gmd@chgs@parse#2\@nil}}%
    \edef\gmu@tempa {\@nx\changes@%
      \@xa\unexpanded\@xa{\gmd@chgs@parse#2\@nil}}%
    \gmu@tempa}

\gmd@chgs@parse 6467 \long\def\gmd@chgs@parse#1#2#3\@nil{{#2}{#1}{#3}}%
```

The checksum

doc provides a checksum mechanism that counts the backslashes in the scanned code. Let's do almost the same.

At the beginning of the source file you may put the \CheckSum macro with a number (in one of TeX's formats) as its argument and TeX with gmdoc shall count the number of the *escape chars* in the source file and tell you in the .log file (and on the terminal) whether you have typed the right number. If you don't type \CheckSum, TeX anyway will tell you how much it is.

```
\check@sum 6485 \newcount\check@sum

\CheckSum 6487 \def\CheckSum#1{\@bsphack\global\check@sum#1\relax\@esphack}

CheckSum 6489 \newcounter{CheckSum}

\step@checksum 6492 \newcommand*\step@checksum{\stepcounter{CheckSum}}
```

And we'll use it in the line [3722](#) (\stepcounter is \global). See also the \chschange declaration, l.[6594](#).

However, the check sum mechanism in gmdoc behaves slightly different than in doc which is nicely visible while gmdocing doc: doc states its check sum to be 2171 and our count counts 2126. The mystery lies in the fact that doc's CheckSum mechanism counts the code's backslashes no matter what they mean and the gmdoc's the escape chars so, among others, `\`` at the default settings increases doc's CheckSum by 2 while the gmdoc's by 1. (There are 38 occurrences of `\`` in doc.dtx macrocodes, I counted myself.)¹³

"But \Finale will be called at the very end of a file. This is exactly the point where we want to know if the file is uncorrupted. Therefore we also call \check@checksum at this point."

In gmdoc we have the `\AtEndInput` hook.

6519 \AtEndInput {\check@checksum}

Based on the lines 723–741 of doc.dtx.

13 My opinion is that nowadays a check sum is not necessary for checking the completeness of a file but I like it as a marker of file development and this more than that is its rôle in gmdoc.

to be able to change it when we don't want \TeX to finish with Code 1 what usually breaks make.

As I mentioned above, I use the check sum mechanism to mark the file growth. Therefore I provide a macro that produces a line on the terminal to be put somewhere at the beginning of the source file's commentary for instance.

```
\gmd@chschangeline 6570 \def\gmd@chschangeline {%
6571   \xiipercent\space\string\chschange
6572   {\@ifundefined{fileversion}{v??}{\fileversion}}%
6573   {\the\year/\the\month/\the\day}%
6574   {\the\c@CheckSum}^^J%
6575   \xiipercent\space\string\chschange
6576   {\@ifundefined{fileversion}{v??}{\fileversion}}%
6577   {\@xa\@gobbletwo\the\year/\the\month/\the\day}%
6578   { with two digit year in case you use \ChangesStart .
6579   \the\c@CheckSum}^^J}
```

And here the meaning of such a line is defined:

```
\chschange 6582 \outer\pdef\chschange {%
6583   % #1 m file version,
6584   % #2 m date,
6585   % (#3) c hecksum,
6586   % [#4] o the reason of check sum change, possibly short.
6587   \@ifQueerEOL
6588   {\def\EOLwasQueer{11}}{\def\EOLwasQueer{10}}%
\EOlwasQueer 6590 \gmd@changes@init
\EOlwasQueer 6591 \chschange@}

\chschange@ 6594 \DeclareCommand\chschange@{mmmo}{%
6595   \changes@{#1}{#2}{\CheckSum_{#3}}
6596   \IfValueT{#4}{because_of_{#4}}%
6597 }% \csname... because \changes is \outer.
6598 \CheckSum_{#3}%
6599 \IfValueF{#4}{%
6600   \if\EOLwasQueer
6601     \afterfi{%
6602       \@ifnextchar\par{%
6603         \@xa\gmd@textEOL\gobble}%
6604       {}%
6605     }%
6606   }% of \afterfi,
6607   \fi}%
6608 }% of \chschange@.
```

It will make a 'General' entry in the change history unless used in some \Define's scope or inside a macro environment. It's intended to be put somewhere at the beginning of the documented file.

Macros from ltxdoc

I'm not sure whether this package still remains 'minimal' but I liked the macros provided by ltxdoc.cls so much...

The next page setup declaration is intended to be used with the article's default Letter paper size. But since

```
\ltxPageLayout 6630 \newcommand*\ltxPageLayout {%
```

“Increase the text width slightly so that width the standard fonts 72 columns of code may appear in a `macrocode` environment.”

```
6634 \setlength{\textwidth}{355pt}%
```

“Increase the marginpar width slightly, for long command names. And increase the left margin by a similar amount.”

To make these settings independent from the defaults (changed e.g. in `gmdoc.cls`) we replace the original `\addtolengths` with `\setlengths`.

```
6644 \setlength\marginparwidth{95pt}%
6645 \setlength\oddsidemargin{82pt}%
6646 \setlength\evensidemargin{82pt} }
```

`\DocInclude` and the `ltxdoc`-like setup

Let’s provide a command for including multiple files into one document. In the `ltxdoc` class such a command is defined to include files as parts. But we prefer to include them as chapters in the classes that provide `\chapter`. We’ll redefine `\maketitle` so that it make a chapter or a part heading *unlike* in `ltxdoc` where the file parts have their title pages with only the filename and article-like titles made by `\maketitle`.

But we will also provide a possibility of typesetting multiple files exactly like with the `ltxdoc` class.

`\DocInclude`

- So, define the `\DocInclude` command, that acts “more or less exactly the same as `\include`, but uses `\DocInput` on a `dtx` [or `.fdd`] file, not `\input` on a `tex` file.”
- Our version will accept also `.sty`, `.cls`, and `.tex` files.

`\DocInclude`

```
6678 \DeclareCommand\DocInclude{0{}m0{} }{%
    % [#1] o path (with closing slash), will not be printed
    % #2 m file name without extension, will be printed
    % [#3] o file extension (with dot) if not .sty, .cls, .tex, .dtx nor .fdd
originally it took just one argument. Here we make it take two, first of which is intended to be the path (with the closing /). This is intended not to print the path in the page footers only the filename.
```

`\HLPrefix`

```
6690 \gdef\HLPrefix{\filesep}%
6691 \gdef\EntryPrefix{\filesep}% we define two rather kernel parameters to expand to the file marker. The first will bring the information to one of the default \IndexPrologue's \ifs. Therefore the definition is global. The latter is such for symmetry.
```

`\GeneralName`

```
6696 \def\GeneralName{#2\actualchar\pk{#2}\relax}% for the changes' history main level entry.
```

Now we check whether we try to include ourselves and if so—we’ll (create and) read an `.auxx` file instead of (the main) `.aux` to avoid an infinite recursion of `\inputs`.

`\gmd@auxext`

```
6703 \edef\gmd@jobname{\jobname}%
6704 \edef\gmd@difilename{%
    we want the filename all ‘other’, just as in \job name.
    \@xa\@xa\@xa\@gobble\@xa\string\csname#2\endcsname}%
6706 \ifx\gmd@jobname\gmd@difilename
6707     \def\gmd@auxext{auxx}%
6708 \else
6709     \def\gmd@auxext{aux}%
6710 \fi
6711 \relax
```

```

\currentfile 6714 \clearpage
6715 \gmd@docincludeaux \def\currentfile{%
6716   gmdoc-IncludeFileNotFound.ooo}%
6717 \let\fullcurrentfile\currentfile
6718 \@ifnonempty{\#3}{%
6719 {%
6720   \unless\if.\@firstofmany{\#3}\relax\@nil
6721     \PackageError{gmdoc}{Optional \xiihash_3 of
6722       \string\DocInclude\space
6723       if present has to begin with a dot (.)}%
6724   \fi
6725   \edef\currentfile{\#2\#3}{%
6726     \IfFileExists{\#1\currentfile}{}{%
6727       \PackageError{gmdoc}{\string\DocInclude\space\_file
6728         \currentfile\space\_not\_found}%
6729     }% of if extension given.
6730     \% if extension not given:
6731     \IfFileExists{\#1\#2.fdd}{\edef\currentfile{\#2.fdd}}% it's not
6732       .fdd,
6733     \IfFileExists{\#1\#2.dtx}{\edef\currentfile{\#2.dtx}}% it's not
6734       .dtx either,
6735     \IfFileExists{\#1\#2.sty}{\edef\currentfile{\#2.sty}}% it's
6736       not .sty,
6737     \IfFileExists{\#1\#2.cls}{\edef\currentfile{\#2.cls}}% it's
6738       not .cls,
6739     \IfFileExists{\#1\#2.tex}{\edef\currentfile{\#2.tex}}% it's
6740       not .tex,
6741     \IfFileExists{\#1\#2.fd}{\edef\currentfile{\#2.fd}}% so it must be .fd or error.
6742     \PackageError{gmdoc}{\string\DocInclude\space\_file
6743       \#1\#2.fdd/dtx/sty/cls/tex/fd\_not\_found.}{}}
6744   }% of if no extension given
6745   \edef\currentfile{\@xa\detokenize\@xa{\currentfile}}%
6746   \edef\fullcurrentfile{\#1\currentfile}%
6747   \ifnum\@auxout=\@partaux
6748     \@latexerr{\string\DocInclude\space cannot be nested}%
6749     \@eha
6750   \else\@docinclude{\#1\#2\#3}\fi% Why is #2 delimited with _ not braced
6751   as we are used to, one may ask.
6752
6753 \def\@docinclude{\def\@docinclude{\#1\#2}%
6754   To match the macro's parameter string, is an answer.
6755   But why is \@docinclude defined so? Originally, in ltxdoc it takes one argument and it's delimited with a space probably in resemblance to the true
6756   \input (\@input in LATEX).
6757   \clearpage
6758   \if@filesw \gmd@writemauxinpaux{\#2.\gmd@auxext}\fi% this strange
6759   macro with a long name is another spurious thing to allow _ in the file-
6760   names (see line 6829). which are allowed anyway unless active or 14.
6761   \tempswattrue
6762   \if@partsw \tempswafalse\edef\gmu@tempb{\#2}%
6763     \for\gmu@tempa:=\partlist\do{\ifx\gmu@tempa%
6764

```

```

6771   \gmu@tempb \@tempsw@true \fi}%
6771   \fi
6772   \if@tempsw@ the file is on \partlist
6773     \let \auxout \partaux
6774   \if@filesw
6775     \immediate\openout \partaux\#2.\gmd@auxext\relax% Yes, only
6775       #2. It's to create and process the partial .aux(x) files always in the
6775       main document's (driver's) directory.
6780     \immediate\write\partaux{\relax}%
6781   \fi

```

"We need to save (and later restore) various index-related commands which might be changed by the included file."

```

6788   \StoringAndRelaxingDo\gmd@doIndexRelated
6789   \if@ltxDocInclude\part {\currentfile}% In the ltxdoc-like setup we
6789     make a part title page with only the filename and the file's \maketitle
6790     will typeset an article-like title.
6792   \else\let \maketitle=\InclMaketitle
6793   \fi% In the default setup we redefine \maketitle to typeset a common
6793     chapter or part heading.
6795   \if@ltxDocInclude\xdef@filekey\fi
6796   \GetFileInfo{\currentfile}% it's my (GM) addition with the account
6796     of using file info in the included files' title/heading etc.
6798   \incl@DocInput {\fullcurrentfile}% originally just \currentfile.
6799   \if@ltxDocInclude\else\xdef@filekey\fi% in the default case we
6799     add new file to the file key after the input because in this case it's the
6799       files own \maketitle what launches the sectioning command that in-
6799         creases the counter.

```

And here is the moment to restore the index-related commands.

```

6805   \RestoringDo\gmd@doIndexRelated
6807   \clearpage
6809   \gmd@writeckpt {\#1\#2}%
6810   \if@filesw\immediate\closeout \partaux\fi
6811   \else% the file isn't on \partlist
6812     \nameuse{cp@\#1\#2}%
6813     \g@emptyify\gmd@ABIOnce
6814   \fi
6815   \let \auxout \mainaux% end of \docinclude.

```

(Two is a sufficient number of iterations to define a macro for.)

```

\xdef@filekey 6819 \def\xdef@filekey {{\relax\nn@{\narrativett}}% This assignment is very trick-
6819   \relax\def\nn@{\narrativett}{\filekey}\expandafter\def\filekey{\narrativett}%
6819     unexpandable not only the one added in this step.
6823   \xdef\filekey{\filekey,\nn@{\thefilediv={\narrativett}}%
6823     {\currentfile}}}

```

To allow `_` in the filenames we must assure `_` will be `_12` while reading the filename. Therefore define

```

\gmd@writemauxinpaux 6829 \def\gmd@writemauxinpaux#1 {%
6829   this name comes from 'write out to main .aux
6829   to input partial .aux'.

```

We wrap `\input {\partial .aux}` in a `_12` hacked scope. This hack is especially recommended here since the .aux file may contain a non-\global stuff that should not

be localised by a group that we would have to establish if we didn't use the hack. (Hope you understand it. If not, notify me and for now I'll only give a hint: "Look at it with the \TeX 's eyes". More uses of this hack are to be seen in `gutils` where they are a bit more explained.)

```

6841   \immediate\write\@mainaux {%
6842     \unexpanded{%
6843       \bgroup
6844         \@makeother\_{\_} to allow underscore
6845         \@makeother\~{\_} to allow paths beginning with ~/
6846         \firstofone}{\egroup
6847       \string\@input{\#1}}}}

```

We also slightly modify a \LaTeX kernel macro `\@writeckpt` to allow `_` in the file name.

```

\gmd@\writeckpt 6854 \def\gmd@\writeckpt#1{%
6855   \immediate\write\@partaux {%
6856     \unexpanded{%
6857       \bgroup
6858         \@makeother\_{\_}
6859         \@makeother\~{\_}
6860         \firstofone}{\char1b\egroup}\%
6861       \@writeckpt{\#1}\%
6862     \immediate\write\@partaux {\@charrb}\}
6863
\gmd@doIndexRelated 6864 \def\gmd@doIndexRelated{%
6865   \do\tableofcontents\do\makeindex\do@EnableCrossrefs
6866   \do\PrintIndex\do\printindex\do\RecordChanges\do%
6867     \PrintChanges
6868   \do\theglossary\do\endtheglossary}
6869
6870 \emptyify\filesep

```

The `ltxdoc` class establishes a special number format for multiple file documentation numbering needed to document the \LaTeX sources. I like it too, so

```

\aaalph 6874 \def\aaalph#1{\aaalph{\csname c@#1\endcsname}}
\@aaalph 6875 \def\@aaalph#1{%
6876   \ifcase#1\or\aa\or\ab\or\ac\or\ad\or\ae\or\af\or\ag\or\ah\or\ai%
6877     \or
6878       \j\or\k\or\l\or\m\or\n\or\o\or\p\or\q\or\r\or\s%
6879       \or
6880       \t\or\u\or\v\or\w\or\x\or\y\or\z\or\A\or\B\or\C%
6881       \or
6882       \D\or\E\or\F\or\G\or\H\or\I\or\J\or\K\or\L\or\M%
6883       \or
6884       \N\or\O\or\P\or\Q\or\R\or\S\or\T\or\U\or\V\or\W%
6885       \or
6886       \X\or\Y\or\Z\else\@ctrerr\fi}

```

A macro that initialises things for `\DocInclude`.

```
\gmd@docincludeaux 6884 \def\gmd@docincludeaux{%

```

We set the things for including the files only once.

```
6886   \global\@relax\gmd@docincludeaux
```

By default, we will include multiple files into one document as chapters in the classes that provide \chapter and as parts elsewhere.

```

6890  \ifx\filediv\relax
6891    \ifx\filedivname\relax% (nor \filediv neither \filedivname is
6892      defined by the user)
6893      \@ifundefined{chapter}{%
6894        \SetFileDiv{part}}%
6895      {\SetFileDiv{chapter}}%
6896    \else% (\filedivname is defined by the user, \filediv is not)
6897      \SetFileDiv{\filedivname}% why not? Inside is \edef so it'll work.
6898    \fi
6899  \else% (\filediv is defined by the user
6900    \ifx\filedivname\relax% and \filedivname is not)
6901      \PackageError{gmdoc}{You've redefined \string\filediv%
6902        \space
6903        without redefining \string\filedivname. }{Please
6904          redefine the
6905          two macros accordingly. You may use \string%
6906            \SetFileDiv{name
6907              without \bslash}.}%
6908    \fi
6909  \fi
6910  \fi
6911 \fi
6912 \def\thefilediv{\aalph{\filedivname}}% The files will be numbered
6913   with letters, lowercase first.
6914 \xa\let\csname\the\filedivname\endcsname=\thefilediv% This li-
6915   ne lets \the<chapter> etc. equal \thefilediv.
\filesep 6916 \def\filesep{\thefilediv-}% File separator (identifier) for the index.
6917 \let\filekey=\@gobble
6918 \g@addto@macro\index@prologue{%
6919   \gdef\@oddfoot{\parbox{\textwidth}{\strut\footnotesize
6920     \raggedright\bfseries\filekey}}% The footer
6921       for the pages of index.
6922   \glet\@evenfoot\@oddfoot}% anyway, it's intended to be oneside.
6923 \g@addto@macro\glossary@prologue{%
6924   \gdef\@oddfoot{\strut\ChangeHistory\hfill\thepage}}% The footer
6925       for the changes history.
6926   \glet\@evenfoot\@oddfoot}%
6927 \gdef\@oddfoot{%
6928   \fileinfo% The footer of the file pages will be its name and, if there is
6929     a file info, also the date and version.
6930   \xa\ifx\csname\ver@\currentfile\endcsname\relax
6931     \File\thefilediv:\{\narrativett\currentfile}%
6932   \else
6933     \GetFileInfo{\currentfile}%
6934     \File\thefilediv:\{\narrativett\filename}%
6935     \Date:\filedate\%
6936     \Version:\fileversion
6937   \fi
6938   \hfill\thepage}%
6939 \glet\@evenfoot\@oddfoot% see line 6931.
6940 \xa\def\csname\filedivname\name\endcsname{File}%
6941   we redefine the
6942     name of the proper division to 'File'.
6943 \ifx\filediv\section

```

```

6955 \let\division=\subsection
6956 \let\subdivision=\subsubsection
6957 \let\subsubdivision=\paragraph

```

If `\filediv` is higher than `\section` we don't change the three divisions (they are `\section`, `\subsection` and `\subsubsection` by default). `\section` seems to me the lowest reasonable sectioning command for the file. If `\filediv` is lower you should rather rethink the level of a file in your documentation not redefine the two divisions.

```

6965 \fi}%

```

The `\filediv` and `\filedivname` macros should always be set together. Therefore provide a macro that takes care of both at once. Its `#1` should be a sectioning name without the backslash.

```

\SetFileDiv 6970 \def\SetFileDiv#1{%
6971   \edef\filedivname{\#1}%
6972   \@xa\let\@xa\filediv\csname#1\endcsname}
\SelfInclude 6976 \def\SelfInclude{\DocInclude{\jobname}}

```

The `ltxdoc` class makes some preparations for inputting multiple files. We are not sure if the user wishes to use `ltxdoc`-like way of documenting (maybe they will prefer what I offer, `gmdocc.cls` e.g.), so we put those preparations into a declaration.

```

\if@ltxDocInclude 6989 \newif\if@ltxDocInclude
\ltxLookSetup 6991 \newcommand*\ltxLookSetup{%
6992   \SetFileDiv{part}%
6993   \ltxPageLayout
6994   \@ltxDocIncludetrue
6995 }
6997 \@onlypreamble\ltxLookSetup

```

The default is that we `\DocInclude` the files due to the original `gmdoc` input settings.

```

7001 \let\incl@DocInput=\DocInput
7003 \@emptyif\currentfile% for the pages outside the \DocInclude's scope. In
                           force for all includes.

```

If you want to `\Doc/SelfInclude` doc-likes:

```

\olddocIncludes 7023 \newcommand*\olddocIncludes{%
7024   \let\incl@DocInput=\OldDocInput}

```

And, if you have set the previous and want to set it back:

```

\gmdocIncludes 7027 \newcommand*\gmdocIncludes{%
7028   \let\incl@DocInput=\DocInput
7029   \AtBeginInput{\QueerEOL}}% to move back the \StraightEOL declaration
                           put at begin input by \olddocIncludes.

```

Redefinition of `\maketitle`

`\maketitle` A not-so-slight alteration of the `\maketitle` command in order it allow multiple titles in one document seems to me very clever. So let's copy again (`ltxdoc.dtx` the lines 643–656):

"The macro to generate titles is easily altered in order that it can be used more than once (an article with many titles). In the original, diverse macros were concealed after

use with `\relax`. We must cancel anything that may have been put into `\@thanks`, etc., otherwise all titles will carry forward any earlier such setting!"

But here in gmdoc we'll do it locally for (each) input not to change the main title settings if there are any.

```

7047 \AtBeginInput{%
7048   \providecommand*\maketitle{\par
7049     \begingroup\def\thefootnote{\fnsymbol{footnote}}%
7050     \setcounter{footnote}\z@%
7051     \def\@makefnmark{\rlap{\@textsuperscript{\normalfont%
7052       \atthefnmark}}}}%
7053   \long\def\@makefntext##1{\parindent\z@\noindent
7054     \hb@xt@1.8em{%
7055       \hss\@textsuperscript{\normalfont\atthefnmark}}##1}%
7056   \if@twocolumn\twocolumn[\@maketitle]%
7057   \else\newpage\global\z@\@maketitle\fi
7058   \thispagestyle{titlepage}\@thanks\endgroup
7059 }
```

"For special formatting requirements (such as in TUGboat), we use page style `titlepage` for this; this is later defined to be `plain`, unless already defined, as, for example, by `ltugboat.sty`."

```
7061 }
```

"If the driver file documents many files, we don't want parts of a title of one to propagate to the next, so we have to cancel these:"

```

7065   \setcounter{footnote}\z@
7066   \gdef\@date{\today}\g@empty\@thanks%
7067   \g@relax\@author\g@relax\@title%
7068 }
```

"When a number of articles are concatenated into a journal, for example, it is not usual for the title pages of such documents to be formatted differently. Therefore, a class such as `ltugboat` can define this macro in advance. However, if no such definition exists, we use page style `plain` for title pages."

```
7075 \@ifundefined{ps@titlepage}{\let\ps@titlepage=\ps@plain}{}
```

And let's provide `\@maketitle` just in case: an error occurred without it at `\TeX`ing with `mwbk.cls` because this class with the default options does not define `\@maketitle`. The below definitions are taken from `report.cls` and `mwrep.cls`.

```

7080 \providecommand*\@maketitle{%
7081   \newpage\null\vskip\z@\relax%
7082   \begin{center}%
7083     \titlesetup
7084     \let\footnote\thanks
7085     {\LARGE\@title\par}%
7086     \vskip\z@.5em%
7087     {\large\lineskip.5em%
7088      \begin{tabular}[t]{c}%
7089        \strut\@author
7090        \end{tabular}\par}%
7091     \vskip\z@.5em%
7092     {\large\@date}%
7093   \end{center}}
```

```

7094 \par\vskip1.5em\relax}%

```

We'd better restore the primary meanings of the macros making a title. (L^AT_EX 2_&
source, File F: ltsect.dtx Date: 1996/12/20 Version v1.0z, lines 3.5.7.9–12.14–17.)

```

\title 7098 \providecommand*\title[1]{\gdef\@title{#1}}
\author 7099 \providecommand*\author[1]{\gdef\@author{#1}}
\date 7100 \providecommand*\date[1]{\gdef\@date{#1}}
\thanks 7101 \providecommand*\thanks[1]{\footnotemark
7102 \protected@xdef\@thanks{\@thanks
7103 \protect\footnotetext[\the\c@footnote]{#1}}%
7104 }%
7105 \providecommand*\and{%
7106 \begin{tabular}{l}
7107 \hskip1em\@plus.17fil%
7108 \begin{tabular}{t}{c}}%
7109 \end{tabular} And finally, let's ini-
7110 \title{#1}%
7111 }% end of \AtBeginInput.

```

The ltxdoc class redefines the `\maketitle` command to allow multiple titles in one document. We'll do the same and something more: our `\Doc/SelfInclude` will turn the file's `\maketitle` into a part or chapter heading. But, if the `\ltxLookSetup` declaration is in force, `\Doc/SelfInclude` will make for an included file a part's title page and an article-like title.

Let's initialise the file division macros.

```

7125 \@relaxen\filediv
7126 \@relaxen\filedivname
7127 \@relaxen\thefilediv

```

If we don't include files the ltxdoc-like way, we wish to redefine `\maketitle` so that it typesets a division's heading.

Now, we redefine `\maketitle` and its relatives.

```

\InclMaketitle 7137 \def\InclMaketitle{%
\and 7143 {\def\and{,}%
7144 we make \and just a comma.
7145 \let\thanks=\gobble%
7146 for the toc version of the heading we discard
7147 \thanks.
7148 \protected@xdef\includetotoc{%
7149 \@title\@ifauthor{%
7150 \protect\space(\@author){}}%
7151 we add the author iff the 'files
7152 have different authors' and author exists (@variousauthors)
7153 }%
7154 \def\thanks##1{\footnotemark
7155 \protected@xdef\@thanks{\@thanks%
7156 to keep the previous \thanks
7157 if there were any.
7158 \protect\footnotetext[\the\c@footnote]{##1}}%
7159 for some
7160 mysterious reasons so defined \thanks do typeset the footnote
7161 mark and text but they don't hyperlink it properly. A hyperref bug?
7162 \empty\@emptyify\@thanks
7163 \protected@xdef\includetitle{%
7164 [\includetotoc]}%
7165 braces to allow [ and ] in the title to toc.
7166 \protect\@title

```

```

7163      {\protect \smallerr% this macro is provided by the gutils package
7164          after the relsize package.
7165      \@ifauthor
7166      {\protect \[\o.15em] \@nx \author
7167          \ifx \relax \date \else , \fi}% after use, \date is let to
7168          % \relax.
7169      {\ifx \relax \date \else \protect \[\o.15em] \fi}

```

The default is that all the included files have the same author(s). In this case we won't print the author(s) in the headings. Otherwise we wish to print them. The information which case are we in is brought by the `\if@variousauthors` switch defined in line 7200.

If we wish to print the author's name (`\if@variousauthors`), then we'll print the date after the author, separated with a comma. If we don't print the author, there still may be a date to be printed. In such a case we break the line, too, and print the date with no comma.

```

7181      \protect \date }% end of \incl@filedivtitle's brace (2nd
7182          or 3rd argument).
7183      }% end of \incl@filedivtitle's \protected@xdef.

```

We `\protect` all the title components to avoid expanding `\footnotemark` hidden in `\thanks` during `\protected@xdef` (and to let it be executed during the typesetting, of course).

```

7187      }% end of the comma-\and's group.
7188      \@xa \filediv \incl@filedivtitle
7189      \@thanks
7190      \g@relaxen \author \g@relaxen \title \g@relaxen \date
7191      \g@emptyify \thanks
7192 }% end of \InclMaketitle.

```

What I make the default, is an assumption that all the multi-documented files have the same author(s). And with the account of the other possibility I provide the below switch and declaration.

```
\if@variousauthors 7200 \newif\if@variousauthors
                    (its name comes from files have different authors).
```

```
\PrintFilesAuthors 7204 \newcommand*{\PrintFilesAuthors}{\@variousauthorstrue}
                    And the counterpart, if you change your mind:
```

```
\SkipFilesAuthors 7206 \newcommand*{\SkipFilesAuthors}{\@variousauthorsfalse}
\@ifauthor 7208 \def\@ifauthor{%
    % #1 what if true
    % #2 what if false
    \ifnum\numexpr\if@variousauthors1\else0\fi*
        \ifx\@author\relax0\else\ifx\@author\@empty0\else1\fi%
            \fi>0
        \@xa\@firstoftwo
    \else
        \@xa\@secondoftwo
    \fi
}
7219 }
```

The file's date and version information

Define \filedate and friends from info in the \ProvidesPackage etc. commands.

```
\GetFileInfo 7226 \def\GetFileInfo#1{%
  \filename 7227 \def\filename{\#1}%
  \filedate 7228 \def\gmu@tempb##1##2##3\relax##4\relax{%
    \def\filedate{\##1}%
  }%
  \fileversion 7230 \def\fileversion{\##2}%
  \fileinfo 7231 \def\fileinfo{\##3}%
}%
\edef\gmu@tempa{\csname ver@\#1\endcsname}%
7232 \cxa\gmu@tempb\gmu@tempa\relax?\relax\relax\relax}
```

Since we may documentally input files that we don't load, as doc e.g., let's define a declaration to be put (in the comment layer) before the line(s) containing \Provides.... The \FileInfo command takes the stuff till the closing] and subsequent line end, extracts from it the info and writes it to the .aux and rescans the stuff. ε -T_EX provides a special primitive for that action but we remain strictly T_EXnical and do it with writing to a file and inputting that file.

```
\FileInfo 7244 \newcommand*\FileInfo{%
  \bgroup
  \gmd@ctallsetup
  \bgroup% yes, we open two groups because we want to rescan tokens in 'usual'
          catcodes. We cannot put \gmd@ctallsetup into the inner macro because
          when that will be executed, the \inputlineno will be too large (the last
          not the first line).
  \let\do\@makeother
  \do\do\{\do\}\do\^\^\M\do\\%
  \gmd@fileinfo}

  \foone{%
    \catcode`!\z@
    \catcode`(\@ne
    \catcode`)\tw@
    \let\do\@makeother
    \do\% we make space 'other' to keep it for scanning the code where it may be
          leading.
    \do\{\do\}\do\^\^\M\do\\}%
  }

\gmd@fileinfo 7265 !\def\gmd@fileinfo#1\Provides#2{\#3}\#4[\#5]\#6^\^\M%
  (!egroup% we close the group of changed catcodes, the catcodes of the arguments
   are set. And we are still in the group for \gmd@ctallsetup.
  \gmd@writeFI(\#2)(\#3)(\#5)%
  \gmd@FIrescan(\#1\Provides#2{\#3}\#4[\#5]\#6)% this macro will close the group.

  )%
}

\gmd@writeFI 7278 \def\gmd@writeFI#1#2#3{%
  {\newlinechar=\endlinechar%
   \immediate\write\@auxout{%
     \global\@nx\@namedef{%
       ver@\#2.\ifP@\firstofmany#1\@nil\sty\else\cls%
         \fi}\{#3\}}}}
  \foone\obeylines{%
```

```

\gmd@FIrescan 7287 \def\gmd@FIrescan#1{%
7292 {\newlinechar=\endlinechar\scantokens{\#1}}\egroup^M}

And, for the case the input file doesn't contain \Provides..., a macro for explicit providing the file info. It's written in analogy to \ProvidesFile, source 2e, file L v1.1g, l. 102.

\ProvideFileInfo 7300 \def\ProvideFileInfo#1{%
7301 \begingroup
7302 \catcode`\_10\catcode\endlinechar_10%
7303 \makeother\/\makeother\&%
7304 \kernel@ifnextchar[\{\gmd@providefii{\#1}\}{\gmd@providefii{%
7305 \#1}[]}]%
7306 }

\gmd@providefii 7309 \def\gmd@providefii#1[#2]{%
7310 (we don't write the file info to .log)
7311 \xa\xdef\csname_ver@\#1\endcsname{#2}%
7312 \endgroup}

```

And a self-reference abbreviation (intended for providing file info for the driver):

```
\ProvideSelfInfo 7316 \def\ProvideSelfInfo{\ProvideFileInfo{\jobname.tex}}
```

A neat conventional statement used in doc's documentation e.g., to be put in \thanks to the title or in a footnote:

```
\filenote 7320 \newcommand*\filenote{This_file_has_version_number%
7321 \fileversion{}_dated_\filedate{.}}
```

And exactly as \thanks:

```
\thfileinfo 7322 \newcommand*\thfileinfo{\thanks\filenote}
```

Miscellanea

The main inputting macro, \DocInput has been provided. But there's another one in doc and it looks very reasonably: \IndexInput. Let's make analogous one here:

```

7333 \foone{\obeylines}%
7334 {%
\IndexInput 7335 \pdef\IndexInput#1{%
7336 \StoreMacro\code@delim%
7337 \CodeDelim*\^Z%
7338 \def\gmd@iihook{%
7339 this hook is \edefed!
7340 \nx^M%
7341 \code@delim\relax\nx\let\nx\EOFMark\relax}%
7342 \DocInput{\#1}\RestoreMacro\code@delim}%
7343 }%
7344 }
```

How does it work? We assume in the input file is no explicit <char>. This char is chosen as the code delimiter and will be put at the end of input. So, entire file contents will be scanned char by char as the code.

The below environment I designed to be able to skip some repeating texts while documenting several packages of mine into one document. At the default settings it's just a \StraightEOL group and in the \skipgmlonely declaration's scope it gobbles its contents.

```
\gmlonely 7360 \newenvironment{gmlonely}{\StraightEOL}{}%
```

```

\skipgmlonly 7362 \newcommand\skipgmlonly[1][]{%
7363   \def\gmu@tempa{%
7364     \def\gmd@skipgmltext{%
7365       \g@emptyify\gmd@skipgmltext
7367       #1%
7368     }% not to count the lines of the substituting text but only of the text omitted
7370   \gmu@tempa
7371   \xa\AtBeginInput\x{a}{\gmu@tempa}%
7372   \renewenvironment{gmlonly}{%
7373     \StraightEOL
7374     \@filesfalse% to forbid writing to .toc, .idx etc.
7375     \setbox0=\vbox\bgroup\egroup\gmd@skipgmltext}%

```

Sometimes in the commentary of this package, so maybe also others, I need to say some char is of category 12 ('other sign'). This I'll mark just as $_{12}$ got by \catother.

```

7382 \foone{\catcode`\_=8 }% we ensure the standard \catcode of _.
7383 {%
7384   \newcommand*\catother{$\_{}_{12}$}%

```

Similarly, if we need to say some char is of category 13 ('active'), we'll write $_{13}$, got by \catactive

```

\catactive 7387 \newcommand*\catactive{$\_{}_{13}$}%
and a letter,  $_{11}$ 

```

```

\catletter 7389 \newcommand*\catletter{$\_{}_{11}$}%
7390 }

```

For the copyright note first I used just `verse` but it requires marking the line ends with `\\"` and indents its contents while I prefer the copyright note to be flushed left. So

```

copyrnote 7395 \renewenvironment*{copyrnote}{%
7396   \StraightEOL\everypar{\hangindent3em\relax\hangafter1}%
7397   \par\addvspace\medskipamount\parindent\z@\obeylines}%
7398   \@codeskipputgfalse\stanz{%

```

I renew the quotation environment to make the fact of quoting visible.

```

\gmd@quotationname 7402 \StoreEnvironment{quotation}
quotation 7403 \def\gmd@quotationname{quotation}
7404 \renewenvironment{quotation}{%

```

The first non-me user complained that `abstract` comes out in quotation marks. That is because `abstract` uses quotation internally. So we first check whether the current environment is quotation or something else.

```

7411   \ifx\@currenvir\gmd@quotationname
7412     \afterfi{\par``\ignorespaces}%
7413     \else\afterfi{\storedcsname{quotation}}%
7414     \fi}
7415   {\ifx\@currenvir\gmd@quotationname
7416     \afterfi{\ifhmode\unskip\fi''\par}%
7417     \else\afterfi{\storedcsname{endquotation}}%
7418     \fi}

```

For some mysterious reasons `\noindent` doesn't work with the first (narrative) paragraph after the code so let's work it around:

```
\gmdnoindent 7423 \def\gmdnoindent{%
7424   \ifvmode\leavevmode\hskip-\parindent\ignorespaces
7425   \fi}%

```

\ignorespaces is added to eat a space inserted by \gmd@textEOL. Without it it also worked but it was a bug: since \parindent is a dimen not skip, TeX looks forward and expands macros to check whether there is a stretch or shrink part and therefore it gobbled the \gmd@textEOL's space.

When a verbatim text occurs in an in-line comment, it's advisable to precede it with % if it begins a not first line of such a comment not to mistake it for a part of code. Moreover, if such a short verb breaks in its middle, it should break with the percent at the beginning of the new line. For this purpose provide \inverb. It breaks with a % at the beginning of new line. Ist starred version puts % also at the end of the upper line.

```
\inverb 7439 \pdef\inverb{%
7440   \gm@ifstar{%
7441     \def\gmu@tempa{\verbhyphen}%
7442       the pre-break.
7443     \empty\gmu@tempb%
7444       the no-break.
7445     \gmd@inverb}%
7446   {\empty\gmu@tempa%
7447     the pre-break empty
7448     \def\gmu@tempb{\gmbboxedspace}%
7449       the no-break boxed space.
7450     \gmd@inverb} }
```

```
\gmbboxedspace 7449 \newcommand*\gmbboxedspace{\hbox{\normalfont{}}}
```

```
\gmd@nlperc 7451 \pdef\gmd@nlperc{%
7452   \ifhmode\unskip\fi
7453   \begingroup\hyphenpenalty\inverbpenalty\relax
7454   \discretionary{\hbox{\gmu@tempa}}% (pre-break). I always put a \hbox
7455     here to make this discretionary score the \hyphenpenalty not \exhy
7456     phenpenalty (The TeXbook p. 96) since the latter may be 10,000 in Polish
7457     typesetting.
7458   {\hbox{\narrationmark}}% (post-break)
7459   {\gmu@tempb}% (no-break).
7460   \endgroup
7461   \penalty1000\hskip0pt\relax}
```

```
\inverbpenalty 7469 \def\inverbpenalty{-1000}
```

```
\gmd@inverb 7471 \pdef\gmd@inverb{%
7472   \gmd@nlperc
7473   \ifmmode\hbox\else\leavevmode\null\fi
7474   \bgroup
7475   \ttverbatim
7476   \narrativett
```

```
\breakablevisspace 7477 \def\breakablevisspace{%
7478   \discretionary{\visiblespace}{\narrationmark}{%
7479     \visiblespace}}%
```

```
\breakbslash 7479 \def\breakbslash{%
7480   \discretionary{}{\narrationmark\type@bslash}{%
7481     \type@bslash}}%
```

```
\breakbrace 7481 \def\breakbrace{%
7482   \discretionary{%
7483     \xiilbrace\verbhyphen}{%
7484     \narrationmark}{%
7485     \xiilbrace}}%
```

```

7486  \gm@verb@eol
7489  \@sverb@chbsl% It's always with visible spaces.
7490 }

\nlperc 7492 \pdef\nlperc{\newline\narrationmark\ignorespaces}

\nlpercent 7494 \pdef\nlpercent{%
7502   \emptyify\gmu@tempa
7503   \def\gmu@tempb{\gboxedspace}%
7504   \gmd@nlperc
7506 }

\incs 7509 \pdef\incs{%
7518   \emptyify\gmu@tempa
7519   \def\gmu@tempb{\gboxedspace}%
7520   \gmd@nlperc\cs
7522 }

\inenv 7524 \def\inenv{\incs[]}%
\inenv an in-line \env

\incmd 7526 \def\incmd{%
7527 it has to be \def to let it expand to let \cmd convert its argument
7528   to a safe string.
7529   \nlpercent\cmd}

\inhash 7530 \def\inhash{\nlpercent\hash}

```

As you see, \inverb and \nlpercent insert a discretionary that breaks to % at the beginning of the lower line. Without the break it's a space (alas at its natural width i.e., not flexible) or, with the starred version, nothing. The starred version puts % also at the end of the upper line. Then \inverb starts sth. like \verb* but the breakables of it break to % in the lower line.

TO-DO: make the space flexible (most probably it requires using sth. else than \discretionary).

An optional hyphen for CSes in the in-line comment:

```

\cs 7548 \@xa\ampulexdef\csname\string\cs\endcsname
7549 [\ampulexhash1]_{\{\{\ampulexhash1\}\}}_{\begingroup}{%
7550   \ifdefined}
\+ 7550 {\begingroup\def\+\{\discre{\gmv@hyphen}{\narrationmark}\}%
7551   \ifdefined}
\ds 7555 \providecommand*\ds{DocStrip}

```

A shorthand for \CS:

```

\CS 7558 \pdef\CS{%
7559   \acro\CS}%
7560   \@ifnextcat\ a{}% we put a space if the next token is _11. It's the next best
7561   thing to checking whether the CS consisting of letters is followed by a space.

\CSs 7564 \pdef\CSs{\CS{}es\@ifnextcat\ a{}% for pluralis.
\CSes 7566 \pdef\CSes{\CS{}es\@ifnextcat\ a{}% for pluralis.

```

Finally, a couple of macros for documenting files playing with %'s catcode(s). Instead of % I used &. They may be at the end because they're used in the commented thread i.e. after package's \usepackage.

```

\CDAnd 7575 \newcommand*\CDAnd{\CodeDelim\&}
\CDPerc 7577 \newcommand*\CDPerc{\CodeDelim\%}

```

And for documenting in general:

A general sectioning command because I foresee a possibility of typesetting the same file once as independent document and another time as a part of bigger whole.

```
\division 7585 \let \division=\section  
\subdivision 7588 \let \subdivision=\subsection  
\subsubdivision 7591 \let \subsubdivision=\subsubsection
```

To kill a tiny little bug in doc.dtx (in line 3299 `\gmu@tempb` and `\gmu@tempc` are written plain not verbatim):

```
gmd@mc 7597 \newcounter{gmd@mc}
```

Note it is after the `macrocode` group

```
\gmd@mchook 7600 \def \gmd@mchook {\stepcounter{gmd@mc} %  
7601   \gmd@mcdiag  
7602   \ifcsname \gmd@mchook \the\c@gmd@mc \endcsname  
7603   \afterfi{\csname \gmd@mchook \the\c@gmd@mc \endcsname} %  
7604   \fi}  
\AfterMacrocode 7606 \long\def \AfterMacrocode#1#2 {\@namedef{gmd@mchook#1}{#2}}
```

What have I done? I declare a new counter and employ it to count the `macrocode[★]`s (and `oldmc[★]`s too, in fact) and attach a hook to (after) the end of every such environment. That lets us to put some stuff pretty far inside the compiled file (for the buggie in doc.dtx, to redefine `\gmu@tempb/c`).

One more detail to explain and define: the `\gmd@mcdiag` macro may be defined to type out a diagnostic message (the `macrocode[★]`'s number, code line number and input line number).

```
7616 \@emptyify \gmd@mcdiag  
\mcdiagOn 7618 \def \mcdiagOn {\def \gmd@mcdiag { %  
7619   \typeout {^J\bslash_end{\gmd@lastenvir}_No.\the\c@gmd@mc  
7620   \space\on@line, \cln.\the\c@codelenum.}}}  
\mcdiagOff 7622 \def \mcdiagOff {\@emptyify \gmd@mcdiag}
```

An environment to display the meaning of macro parameters: its items are automatically numbered as #1, #2 etc.

```
7626 \DeclareEnvironment{enumargs}{o} % the optional argument specifies num-  
    ber of #'s; it's of the o type to inform if it was not given by the user to handle  
    a possible active char touched by argument's catcher; can be 1 (the default), 2  
    or 4; any else produces one #.
```

```
7638 { %  
7639   \StraightEOL  
7640   \if@aftercode  
7641     \edef \gmu@tempa {\the\leftskip} %  
7642     \edef \gmu@tempb {\the\hangindent} %  
7643   \fi  
7644   \enumerate  
7645   \if@aftercode  
7646     \leftskip=\glueexpr \gmu@tempa+\gmu@tempb \relax  
7647   \fi  
7648   \edef \gmd@ea@hashes { %
```

```

7649  \# \ifcase \IfValueTF{\#1}{\#1}{1} \relax
7650  \or \or \# \or \or \# \# \# \fi }%
7652  \@namedef{label \@enumctr} {%
7653  \env{\if@aftercode \narrationmark \fi
7654  \relax% to stop \ignorespaces
7655  \gmd@ea@bwrap
7656  \gmd@ea@hashes
7657  \csname \the \@enumctr \endcsname
7658  \gmd@ea@ewrap} }% of \label<@enumctr>.
7659  \let \mand \item
7660  \provide \gmd@ea@wraps {%
7661  \emptyify \gmd@ea@ewrap
7662  \emptyify \gmd@ea@bwrap} %
7663  \gmd@ea@wraps
7664  \def \opt {%
7665  \def \gmd@ea@bwrap {[]}\def \gmd@ea@ewrap {} }%
7666  \item
7667  \gmd@ea@wraps} %
7669  \settowidth{\@tempdima}{\narrativett_x\gmd@ea@hashes7x}%
7670  \edef \gmd@ea@xxxwd {\the \@tempdima}%
7672  \DeclareCommand \dc ! {%
7673  \Q{*>} % (1) we check whether there's a sergeant right of the prefix or a star
7674  to suppress parentheses,
7675  \Q{P!1L\long_iI} % (2) an optional 'bare' prefix for a 'long' argument or
7676  for ignored
7677  b % (3) prefix(es) in curly braces ( This way we allow the prefix(es) to be
7678  braced or not at the author's option),
7679  S{bBcCmoOQsSTAGK\afterassignment} % (4) (optional) argument type
7680  specifier,
7681  b % (5) (optional) default value of the specified argument or (for K and G)
7682  mandatory.
7683  b % (6) default of K and G.
7685  }{%
7686  \IfAmong_x\among{\#\#1}%
7687  { % a * suppresses bracket/brace/parentheses decoration.
7688  \def \gmd@ea@bwrap {\hbox _to _\gmd@ea@xxxwd \bgroup \hss}%
7689  \def \gmd@ea@ewrap {\hss \egroup}%
7690  }%
7691  { % if there's no * in #1, be wrap the item label in braces/brackets/parentheses.
7692  \IfAmong_x\among{bB} { % I decide not to print m type arguments in
7693  braces because the braces are not mandatory for this type.
7694  \def \gmd@ea@bwrap {\{}%
7695  \def \gmd@ea@ewrap {\}}%
7696  }{%
7697  \IfAmong_x\among{cC} {%
7698  \def \gmd@ea@bwrap {\{}%
7699  \def \gmd@ea@ewrap {\}}%
7700  }{%
7701  \IfAmong_x\among{oO} {%
7702  \def \gmd@ea@bwrap {[}%
7703  \def \gmd@ea@ewrap {]}%
7704  }{%
7705  }

```

```

7706 } {}%
7707 \IfAmong \#4 \among {G} {%
7708   \def \gmd@ea@bwrap {\detokenize @xa{\@firstoftwo##5}}%
7709   \def \gmd@ea@ewrap {\detokenize @xa{%
7710     \@secondoftwo##5}}%
7711 } {}%
7712 \IfAmong \#4 \among {A} {%
7713   \def \gmd@ea@bwrap {<}%
7714   \def \gmd@ea@ewrap {>}%
7715 } {}%
7716 } % of if no * in #1.
7717 \IfAmong \#4 \among {mQsSTK\afterassignment} {%
7718   \def \gmd@ea@bwrap {\hbox to \gmd@ea@xxxwd \bgroup \hss}%
7719   \def \gmd@ea@ewrap {\hss \egroup}%
7720 } {}%

```

we add a normal space

```

7721   \addtomacro \gmd@ea@ewrap {{\normalfont \ }}%
7722   \IfValueT {##2} {%
7723     \addtomacro \gmd@ea@ewrap {>{\string##2\ }}}%
7724   \IfValueT {##3} {%
7725     \addtomacro \gmd@ea@ewrap {>\{\##3\ }}}%
7726   \IfValueT {##4} {%
7727     \ifx s##4%
7728       \addtomacro \gmd@ea@ewrap {%
7729         \llap {\metachar[]}\scanverb{*}\metachar}%
7730       \else \addtomacro \gmd@ea@ewrap {##4}%
7731       \fi}%
7732   \IfValueT {##5} {%
7733     \addtomacro \gmd@ea@ewrap {\{%
7734       %\ttverbatim breakable chars won't work because we are in the item's label's
7735       \% \hbox.
7736       \scanverb*##5}%
7737     \}}%
7738   \IfValueT {##6} {%
7739     \addtomacro \gmd@ea@ewrap {\{%
7740       %\ttverbatim breakable chars won't work because we are in the item's label's
7741       \% \hbox.
7742       \scanverb*##6}%
7743     \}}%
7744   \def \gmd@blubra {%
7745     \addtomacro \gmd@ea@bwrap {%
7746       \begingroup
7747       \relaxen \gmd@ea@hashes
7748       \namedef {the \enumctr} {\<ign.\>}%
7749     }%
7750     \prependmacro \gmd@ea@ewrap {%
7751       \endgroup}%
7752     \addtomacro \gmd@ea@ewrap {%
7753       \global \advance \csname c@\enumctr \endcsname \m@ne
7754     }%
7755     \emptyify \gmd@blubra
7756   }%

```

```

7757   \IfIntersect{\#2}{\#1}{\gmd@blubra}{}%
7758   \IfIntersect{\#3}{\#1}{\gmd@blubra}{}%
7759   \IfAmong{\#4}{among}{\afterassignment}{\gmd@blubra}{}%
7760 \item\relax}%
7761   \IfNoValueT{\#1}{\@ifnextac@gobble}{}% to gobble a possible ac-
    tive line end or active ^A or ^B that might have occurred because
    of \futurelet of the optional argument checker.
7762 }% of begin definition
7763 {\endenumerate}

```

The starred version is intended for lists of arguments some of which are optional: to align them in line.

```

enumargs*
\gmd@ea@wraps 7771 \newenvironment*{enumargs*}{%
7772   \def\gmd@ea@wraps{%
7773     \def\gmd@ea@bwrap{\ } \def\gmd@ea@ewrap{\ }}%
7774   \enumargs} \endenumargs}

```

doc-compatibility

My TeX Guru recommended me to write hyperlinking for doc. The suggestion came out when writing of gmdoc was at such a stage that I thought it to be much easier to write a couple of \lets to make gmdoc able to typeset sources written for doc than to write a new package that adds hyperlinking to doc. So...

The doc package makes % an ignored char. Here the % delimits the code and therefore has to be ‘other’. But only the first one after the code. The others we may re\catcode to be ignored and we do it indeed in line [2527](#).

At the very beginning of a doc-prepared file we meet a nice command \CharacterTable. My TeX Guru says it’s a bit old fashioned these days so let’s just make it notify the user:

```

\CharacterTable 7797 \def\CharacterTable{\begingroup
7798   \makeother{\makeother\%}
7799   \CharacterTable}
7800 \foone{%
7801   \catcode`\[=\catcode`\]=2%
7802   \makeother{\makeother\%}}
7803 [
7804 \CharacterTable 7805 \def\CharacterTable{\#1\#2}{[\endgroup
7806   \message[^J^J gmdoc.sty package:^J
7807   ===== The input file contains the \bslash
7808   CharacterTable.^J
7809   ===== If you really need to check the correctness of
7810   the chars,^J
7811   ===== please notify the author of gmdoc.sty at the
7812   email address^J
7813   ===== given in the legal notice in gmdoc.sty.^J]^J}%
7814 ]

```

Similarly as doc, gmdoc provides macrocode, macro and environment environments. Unlike in doc, \end{macrocode} does not require to be preceded with any particular number of spaces. Unlike in doc, it is not a kind of verbatim, however, which means the code and narration layers remains in force inside it which means that any text

after the first % in a line will be processed as narration (and its control sequences will be executed). For a discussion of a possible workaround see line [8184](#).

Let us now look over other original doc's control sequences and let's 'domesticate' them if they are not yet.

\DescribeMacro
\DescribeEnv

The \DescribeMacro and \DescribeEnv commands seem to correspond with my \TextUsage macro in its plain and starred version respectively except they don't typeset their arguments in the text i.e., they do two things of the three. So let's \def them to do these two things in this package, too:

```
\DescribeMacro 7832 \outer\def\DescribeMacro{%
7833   \@bsphack
7834   \begingroup\MakePrivateLetters
7835   \gmd@ifonetoken\Describe@Macro\Describe@Env}
```

Note that if the argument to \DescribeMacro is not a (possibly starred) control sequence, then as an environment's name shall it be processed *except* the \MakePrivIateOthers re\catcodeing shall not be done to it.

```
\DescribeEnv 7840 \outer\def\DescribeEnv{%
7841   \@bsphack
7842   \begingroup\MakePrivateOthers\Describe@Env}
```

Actually, I've used the \Describe... commands myself a few times, so let's \def a common command with a starred version:

```
\Describe 7847 \outer\def\Describe{%
  It doesn't typeset its argument in the point of occurrence.
  \leavevmode
  \@bsphack
  \begingroup\MakePrivateLetters
  \gmd@ifstar{\MakePrivateOthers\Describe@Env}{%
    \Describe@Macro}}
```

The below two definitions are adjusted ~s of \Text@UsgMacro and \Text@UsgEnvir.

```
\Describe@Macro 7857 \long\def\Describe@Macro#1{%
7858   \endgroup
7859   \strut\Text@Marginize*{#1}%
7860   \@usgentryze{#1}%
7861   we declare kind of formatting the entry
7862   \text@indexmacro{#1}%
7863   \@esphack}
```

```
\Describe@Env 7865 \def\Describe@Env#1{%
7866   \endgroup
7867   \strut\Text@Marginize*{#1}%
7868   \@usgentryze{#1}%
7869   we declare the 'usage' kind of formatting the entry and
7870   index the sequence #1.
7871   \text@indexenvir{#1}%
7872   \@esphack}
```

Note that here the environments' names are typeset in \narrativett font just like the macros', *unlike* in doc.

My understanding of 'minimality' includes avoiding too much freedom as causing chaos not beauty. That's the philosophical and æ sthetic reason why I don't provide \MacroFont. In my opinion there's a noble tradition of typesetting the TeX code in

\MacroFont

\tt font and this tradition sustained should be. If one wants to change the tradition, let them redefine \tt, in T_EX it's no problem. I suppose \MacroFont is not used explicitly, and that it's (re)defined at most, but just in case let's \let:

7886 \let \MacroFont \tt

\CodeIndent We have provided \CodeIndent in line 2344. And it corresponds with doc's \MacroIndent so

\MacroIndent 7894 \let \MacroIndent \CodeIndent

And similarly the other skips:

\MacrocodeTopsep 7896 \let \MacrocodeTopsep \CodeTopsep

\MacroTopsep Note that \MacroTopsep is defined in gmdoc and has the same rôle as in doc.

\SpecialEscapechar 7900 \let \SpecialEscapechar \CodeEscapeChar

\theCodelineNo is not used in gmdoc. Instead of it there is \LineNumFont declaration and a possibility to redefine \theCodelineNum as for all the counters. Here the \LineNumFont is used two different ways, to set the benchmark width for a line number among others, so it's not appropriate to put two things into one macro. Thus let's give the user a notice if they defined this macro:

Because of possible localness of the definitions it seems to be better to add a check at the end of each \DocInput or \IndexInput.

```
7914 \AtEndInput {\@ifundefined{\theCodelineNo} {} { \PackageInfo {%
  gmdoc} {The
    \string \theCodelineNo \space macro has no effect
    here, please use
  \string \LineNumFont \space for setting the font and/or
  \string \theCodelineNum \space to set the number
  format.}}}
```

I hope this lack will not cause big trouble.

For further notifications let's define a shorthand:

```
\noeffect@info 7922 \def \noeffect@info#1 {\@ifundefined{#1} {} { \PackageInfo {%
  gmdoc} {^^J%
    The \bslash#1 macro is not supported by this
    package^^J
  and therefore has no effect but this notification.^^J
  If you think it should have, please contact the
    maintainer^^J
  indicated in the package's legal note.^^J}}}
```

The four macros formatting the macro and environment names, namely \PrintDescribeMacro, \PrintMacroName, \PrintDescribeEnv and \PrintEnvName are not supported by gmdoc. They seem to me to be too internal to take care of them. Note that in the name of (aesthetic) minimality and (my) convenience I deprive you of easy knobs to set strange formats for verbatim bits: I think they are not advisable.

Let us just notify the user.

```
7939 \AtEndInput {%
7940   \noeffect@info{\PrintDescribeMacro}%
7941   \noeffect@info{\PrintMacroName}%
7942   \noeffect@info{\PrintDescribeEnv}%
}
```

```
7943 \noeffect@info {PrintEnvName} }
```

\CodelineNumbered

The `\CodelineNumbered` declaration of `doc` seems to be equivalent to our `noindex` option with the `linesnotnum` option set off so let's define it such a way.

\CodelineNumbered

```
7948 \def\CodelineNumbered{\AtBeginDocument{\gag@index}}
7949 \@onlypreamble\CodelineNumbered
```

Note that if the `linesnotnum` option is in force, this declaration shall not revert its effect.

I assume that if one wishes to use `doc`'s interface then they'll not use `gmdoc`'s options but just the default.

The `\CodelineIndex` and `\PageIndex` declarations correspond with the `gmdoc`'s default and the `pageindex` option respectively. Therefore let's `\let`

```
7961 \let\CodelineIndex\@pageindexfalse
7962 \@onlypreamble\CodelineIndex
```

```
7964 \let\PageIndex\@pageindextrue
7966 \@onlypreamble\PageIndex
```

The next two declarations I find useful and smart:

\DisableCrossrefs

```
7970 \def\DisableCrossrefs{\@bsphack\gag@index\@esphack}
```

\EnableCrossrefs

```
7972 \def\EnableCrossrefs{\@bsphack\ungag@index}
```

\DisableCrossrefs

```
7973 \def\DisableCrossrefs{\@bsphack\@esphack}\@esphack}
```

The latter definition is made due to the footnote 6 on p. 8 of the Frank Mittelbach's `doc`'s documentation and both of them are copies of lines 302–304 of it modulo `\[un]gag@index`.

The subsequent few lines I copy almost verbatim ;-) from the lines 611–620.

\AlsoImplementation

```
7981 \newcommand*\AlsoImplementation{\@bsphack
7982 \long\def\StopEventually##1{\gdef\Finale{##1}}% we define \Finale%
    % naley just to expand to the argument of \StopEventually not to to
    add anything to the end input hook because \Finale should only be ex-
    ecuted if entire document is typeset.
    %\init@checksum is obsolete in gmdoc at this point: the CheckSum counter is reset
    % just at the beginning of (each of probably numerous) input(s).
    \@esphack}
7993 \AlsoImplementation
```

“When the user places an `\OnlyDescription` declaration in the driver file the document should only be typeset up to `\StopEventually`. We therefore have to redefine this macro.”

\OnlyDescription

```
8002 \def\OnlyDescription{\@bsphack\long\def\StopEventually##1{%
```

“In this case the argument of `\StopEventually` should be set and afterwards `TEX` should stop reading from this file. Therefore we finish this macro with”

```
8006 ##1\endinput}\@esphack}
```

“If no `\StopEventually` command is given we silently ignore a `\Finale` issued.”

```
8011 \@relaxen\Finale
```

\meta

The `\meta` macro is so beautifully crafted in `doc` that I couldn't resist copying it into `gmutils`. It's also available in Knuthian (*The T_EXbook* format's) disguise `\<the argu-`

ment>.

The checksum mechanism is provided and developed for a slightly different purpose.

Most of doc's indexing commands have already been 'almost defined' in gmdoc:

```
\SpecialMainEnvIndex 8023 \let \SpecialMainIndex=\DefIndex  
8026 \def \SpecialMainEnvIndex {\csname \CodeDefIndex\endcsname}% we don't  
type \DefIndex explicitly here because it's \outer, remember?  
\SpecialIndex 8031 \let \SpecialIndex=\CodeCommonIndex  
\SpecialUsageIndex 8033 \let \SpecialUsageIndex=\TextUsgIndex  
\SpecialEnvIndex 8035 \def \SpecialEnvIndex {\csname \TextUsgIndex\endcsname}  
\SortIndex 8037 \def \SortIndex#1#2{\index{#1\actualchar#2}}
```

"All these macros are usually used by other macros; you will need them only in an emergency."

Therefore I made the assumption(s) that 'Main' indexing macros are used in my 'Code' context and the 'Usage' ones in my 'Text' context.

`\verbatimchar` Frank Mittelbach in doc provides the `\verbatimchar` macro to (re)define the `\verb[]*`'s delimiter for the index entries. The gmdoc package uses the same macro and its default definition is `{&}`. When you use doc you may have to redefine `\verb|batimchar` if you use (and index) the `\+ control sequence`. gmdoc does a check for the analogous situation (i.e., for processing `\&`) and if it occurs it takes `$` as the `\verb*`'s delimiter. So strange delimiters are chosen deliberately to allow any 'other' chars in the environments' names. If this would cause problems, please notify me and we'll think of adjustments.

```
\verbatimchar 8057 \def \verbatimchar {&}  
\IndexPrologue \IndexPrologue is defined in line 5858. And other doc index commands too.  
8073 \@ifundefined{main} {} {\let \DefEntry=\main}  
8075 \@ifundefined{usage} {} {\let \UsgEntry=\usage}
```

About how the DocStrip directives are supported by gmdoc, see section The DocStrip.... This support is not *that* sophisticated as in doc, among others, it doesn't count the modules' nesting. Therefore if we don't want an error while gmdocumenting doc-prepared files, better let's define doc's counter for the modules' depths.

```
StandardModuleDepth 8083 \newcounter{StandardModuleDepth}  
For now let's just mark the macro for further development DocstyleParms
```

```
\ 8088 \noeffect@info{DocstyleParms}
```

For possible further development or to notify the user once and forever:

```
\DontCheckModules 8093 \@emptyify \DontCheckModules \noeffect@info{DontCheckModules}  
\CheckModules 8094 \@emptyify \CheckModules \noeffect@info{CheckModules}
```

`\Module` The `\Module` macro is provided exactly as in doc.

```
\AltMacroFont 8098 \@emptyify \AltMacroFont \noeffect@info{AltMacroFont}
```

"And finally the most important bit: we change the `\catcode` of `%` so that it is ignored (which is how we are able to produce this document!). We provide two commands to do the actual switching."

```
8104 \def \MakePercentIgnore {\catcode`\%9\relax}  
8105 \def \MakePercentComment {\catcode`\%14\relax}
```

gmdocing doc.dtx

The author(s) of doc suggest(s):

“For examples of the use of most—if not all—of the features described above consult the doc.dtx source itself.”

Therefore I hope that after doc.dtx has been gmdoc-ed, one can say gmdoc is doc-compatible “at most—if not at all”.

TEXing the original doc with my humble¹⁴ package was a challenge and a milestone experience in my TEX life.

One of minor errors was caused by my understanding of a ‘shortverb’ char: due to gmverb, in the math mode an active ‘shortverb’ char expands to itself’s ‘other’ version thanks to \string (It’s done with | in mind). doc’s concept is different, there a ‘shortverb’ char should in the math mode work as shortverb. So let it be as they wish: gmverb provides \OldMakeShortVerb and the old-style input commands change the inner macros so that also \MakeShortVerb works as in doc (cf. line 8146).

We also redefine the macro environment to make it mark the first code line as the point of defining of its argument, because doc.dtx uses this environment also for implicit definitions.

```
\OldDocInput 8143 \def \OldDocInput {%
  8145   \AtBeginOnce{ \StraightEOL
  8146     \let \MakeShortVerb=\old@MakeShortVerb
  8148   \OldMacrocodes }%
  8149   \bgroup \makeother \_% it's to allow _ in the filenames. The next macro will
      close the group.
  8151   \Doc@Input }
```

We don’t switch the @codeskipput switch neither we check it because in ‘old’ world there’s nothing to switch this switch in the narration layer.

I had a hot and wild TEX all the night and what a bliss when the ‘Successfully formatted 67 page(s)’ message appeared.

My package needed fixing some bugs and adding some compatibility adjustments (listed in the previous section) and the original doc.dtx source file needed a few adjustments too because some crucial differences came out. I’d like to write a word about them now.

The first but not least is that the author(s) of doc give the CS marking commands non-macro arguments sometimes, e.g., \DescribeMacro{StandardModuleDepth}. Therefore we should launch the *starred* versions of corresponding gmdoc commands. This means the doc-like commands will not look for the CS’s occurrence in the code but will mark the first codeline met.

Another crucial difference is that in gmdoc the narrative and the code layers are separated with only the code delimiter and therefore may be much more mixed than in doc. among others, the macro environment is *not* a typical verbatim like: the texts commented out within macrocode are considered a normal commentary i.e., not verbatim. Therefore some macros ‘commented out’ to be shown verbatim as an example source must have been ‘additionally’ verbatimized for gmdoc with the shortverb chars e.g. You may also change the code delimiter for a while, e.g., the line

```
8184 \% \AVerySpecialMacro \% delete the first \% when. . .
```

was got with

¹⁴ What a *false* modesty! ;)

```
\CodeDelim\.
% \AVerySpecialMacro % delete the first %
when.\unskip|..|\CDPerc
```

One more difference is that my shortverb chars expand to their $_2$ versions in the math mode while in doc remain shortverb, so I added a declaration \OldMakeShortVerb etc.

Moreover, it's \TeX ing doc what inspired adding the \StraightEOL and \QueerEOL declarations.

\OCRInclude

I realised that I want to print all my \TeX source files verbatim just in case my computers and electronic memories break so that I can reconstruct them via OCR . For this purpose I provide \OCRInclude. It takes the same arguments as \DocInclude only typesets a file with no index nor line numbers.

```
\OCRInclude 8209 \DeclareCommand\OCRInclude{0{}m0{} }{%
8210   \StoreMacro\incl@DocInput
\incl@DocInput 8211 \def\incl@DocInput##1{%
8212   \begingroup
8213   \CodeSpacesBlank
8214   \begin{inputhook}
8215     \title{\currentfile}\maketitle
8216     \noverbatimspecials
8217     \relaxen@\xverbatim
8218     \relaxen\check@percent
8219     \RestoreMacro@\verbatim
8220     \verbatimleftskip\z@skip
8221     \verbatim
8222     \makeother\|% because \ttverbatim doesn't do that.
8223     \texcode@hook% we add some special stuff, e.g. in gmdoc.cls we
8224     \input{##1}%
8225   \endgroup}%
8226   \csname\string\DocInclude\endcsname{#1}{#2}{#3}%
8227   \RestoreMacro\incl@DocInput
8228 }
```

Polishing, development and bugs

- \MakePrivateLetters theoretically may interfere with \activeating some chars to allow line breaks. But making a space or an opening brace a letter seems so perverse that we may feel safe not to take account of such a possibility.
- When countalllines* option is enabled, the comment lines that don't produce any printed output result with a (blank) line too because there's put a hypertarget at the beginning of them. But for now let's assume this option is for draft versions so hasn't be perfect.
- Marcin Woliński suggests to add the marginpar clauses for the AMS classes as we did for the standard ones in the lines 2180–2185. Most probably I can do it on request when I only know the classes' names and their 'marginpar status'.
- When the countalllines* option is in force, some \list environments shall raise the 'missing \item' error if you don't put the first \item in the same line as \begin{*environment*} because the (comment-) line number is printed.

- I'm prone to make the control sequences hyperlinks to the(ir) 'definition' occurrences. It doesn't seem to be a big work compared with what has been done so far.
- Is `\RecordChanges` really necessary these days? Shouldn't be the `\makeglossary` command rather executed by default?¹⁵
 - Do you use `\listoftables` and/or `\listoffigures` in your documentations? If so, I should 'EOL-straighten' them like `\tableofcontents`, I suppose (cf. line 2630).
 - Some lines of non-printing stuff such as `\Define...` and `\changes` connecting the narration with the code resulted with unexpected large vertical space. Adding a fully blank line between the printed narration text and not printed stuff helped.
 - Specifying `codespacesgrey`, `\codespacesblank` results in typesetting all the spaces grey including the leading ones.
 - About the DocStrip [verbatim mode directive](#) see above.

[No] *<eof>*

Until version 0.99i a file that is `\DocInput` had to be ended with a comment line with an `\EOF` or `\NoEOF` CS that suppressed the end-of-file character to make input end properly. Since version 0.99i however the proper ending of input is achieved with `\everyeof` and therefore `\EOF` and `\NoEOF` become a bit obsolete.

If the user doesn't wish the documentation to be ended by '*<eof>*', they should redefine the `\EOFMark` CS or end the file with a comment ending with `\NoEOF` macro defined below¹⁶:

```

8306 \foone{\catcode`\\^M\active}%
\@NoEOF \def\@NoEOF#1^M{%
8308   \relax\@EOFMark\endinput}%
\@EOF \def\@EOF#1^M{\endinput}%
\NoEOF \def\NoEOF{\@QueerEOL\@NoEOF}%
\EOF \def\EOF{\@QueerEOL\@EOF}

```

As you probably see, `\[No]EOF` have the 'immediate' `\endinput` effect: the file ends even in the middle of a line, the stuff after `\(No)EOF` will be gobbled unlike with a bare `\endinput`.

```

8325 \endinput
8327 </ package>

```

¹⁵ It's understandable that ten years earlier writing things out to the files remarkably decelerated TeX, but nowadays it does not in most cases. That's why `\makeindex` is launched by default in gmdoc.

¹⁶ Thanks to Bernd Raichle at BachoTeX 2006 Pearl Session where he presented `\input`ing a file inside `\edef`.

b. The `gmdocc` Class For `gmdoc` Driver Files¹

Written by Natror (Grzegorz Murzynowski),
natror at o2 dot pl

© 2006–2010 by Natror (Grzegorz Murzynowski).

This program is subject to the L^AT_EX Project Public License.

See [http://www.ctan.org/tex--archive/help/Catalogue/licenses.lppl.html](http://www.ctan.org/tex-archive/help/Catalogue/licenses.lppl.html) for the details of that license.

LPPL status: "author-maintained".

```
49 \NeedsTeXFormat {LaTeX2e}
50 \ProvidesClass {gmdocc}
51 [2010/03/02 v0.84 a class for gmdoc driver
   files (GM)]
```

Intro

This file is a part of `gmdoc` bundle and provides a document class for the driver files documenting (L^A)T_EX packages &c. with my `gmdoc.sty` package. It's not necessary, of course: most probably you may use another document class you like.

By default this class loads `mwart` class with `a4paper` (default) option and `\modern` package with `T1` fontencoding. It loads also my `gmdoc` documenting package which loads some auxiliary packages of mine and the standard ones.

If the `mwart` class is not found, the standard `article` class is loaded instead. Similarly, if the `\modern` is not found, the standard Computer Modern font family is used in the default font encoding.

Usage

For the ideas and details of `gmdocing` of the (L^A)T_EX files see the `gmdoc.sty` file's documentation (chapter a). The rôle of the `gmdocc` document class is rather auxiliary and exemplary. Most probably, you may use your favourite document class with the settings you wish. This class I wrote to meet my needs of fine formatting, such as not numbered sections and sans serif demi bold headings.

However, with the users other than myself in mind, I added some conditional clauses that make this class works also if an `mwcls` class or the `\modern` package are unknown.

Of rather many options supported by `gmdoc.sty`, this class chooses my favourite, i.e., the default. An exception is made for the `noindex` option, which is provided by this class and passed to `gmdoc.sty`. This is intended for the case you don't want to make an index.

Simili modo, the `nochanges` option is provided to turn creating the change history off.

¹ This file has version number v0.84 dated 2010/03/02.

Both of the above options turn the *writing out to the files* off. They don't turn off `\PrintIndex` nor `\PrintChanges`. (Those two commands are no-ops by themselves if there's no `.ind` (n)or `.gls` file respectively.)

`outeroff`

One more option is `outeroff`. It's intended for compiling the documentation of macros defined with the `\outer` prefix. It `\relaxes` this prefix so the '`\outer`' macros' names can appear in the arguments of other macros, which is necessary to pretty mark and index them.

I decided not to make discarding `\outer` the default because it seems that L^AT_EX writers don't use it in general and `gmdoc.sty` *does* make some use of it.

`debug`

This class provides also the `debug` option. It turns the `\if@debug` Boolean switch True and loads the `trace` package that was a great help to me while debugging `gmdoc.sty`.

The default base document class loaded by `gmdocc.cls` is Marcin Woliński `mwart`. If you have not installed it on your computer, the standard article will be used.

Moreover, if you like MW's classes (as I do) and need `\chapter` (for multiple files' input e.g.), you may declare another `mwcls` with the option homonymic with the class's name: `mwrep` for `mwrep` and `mwbk` for `mwbk`. For the symmetry there's also `mwart` option (equivalent to the default setting).

`mwart`

The existence test is done for any MW class option as it is in the default case.

Since version 0.99g (November 2007) the bundle goes X_ET_EX and that means you can use the system fonts if you wish, just specify the `sysfonts` option and the three basic X_ET_EX-related packages (`fontspec`, `xunicode` and `xltxtra`) will be loaded and then you can specify fonts with the `fontspec` declarations. For use of them check the driver of this documentation where the T_EX Gyre Pagella font is specified as the default Roman.

There are also some options for mono and sans fonts, see the changes history for details.

The `minion` option sets Adobe Minion Pro as the main font, the `pagella` sets T_EX Gyre Pagella as the main font.

The `cronos` option sets Adobe Cronos Pro as the sans serif font, the `trebuchet` option sets MS Trebuchet as sans serif.

The `cursor` (working only with X_ET_EX & `fontspec`) option sets T_EX Gyre Cursor as the typewriter font. It emboldens it to the optical weight of Computer/Latin Modern Mono in the code (`embolden=2.5`) and leaves light (`embolden=1`) for verbatims in the narrative. Moreover, this option also prepares a condensed version (`extend=0.87`) for verbatims in `marginpars`.

Note that with no option for the monospaced font the default (with X_ET_EX) will be Latin Modern Mono and then Latin Modern Mono Light Condensed is set for verbatims in `marginpars` (if available).

This class sets `\verb@imspecials@»[;]` if the engine is X_ET_EX, see the `gmverb` documentation to learn about this declaration. Remember that `\verb@imspecials` whatever would they be, have no effect on the code layer.

`\EOFMark`

The `\EOFMark` in this class typesets like this (of course, you can redefine it as you wish):

□

The Code

`178 \RequirePackage {xkeyval}`

A shorthands for options processing (I know `xkeyval` to little to redefine the default prefix and family).

```
\gm@DOX 183 \newcommand*{\gm@DOX}{\DeclareOptionX{gmcc}{}}
\gm@EOX 184 \newcommand*{\gm@EOX}{\ExecuteOptionsX{gmcc}{}}
```

We define the `class` option. I prefer the `mwcls`, but you can choose anything else, then the standard article is loaded. Therefore we'd better provide a Boolean switch to keep the score of what was chosen. It's to avoid unused options if `article` is chosen.

```
\ifgmcc@mwcls 193 \newif\ifgmcc@mwcls
```

Note that the following option defines `\gmcc@class#1`.

`class` 196 \gm@DOX{class} { % the default will be Marcin Woliński class (`mwcls`) analogous to
article, see line 357.

```
\gmcc@CLASS 198 \def\gmcc@CLASS{\#1}%
199 \@for\gmcc@resa:=mwart,mwrep,mwbk\do{%
200 \ifx\gmcc@CLASS\gmcc@resa\gmcc@mwcistrue\fi}%
201 }
```

`mwart` 203 \gm@DOX{mwart} {\gmcc@class{mwart}}% The `mwart` class may also be declared explicitly.

`mwrep` 206 \gm@DOX{mwrep} {\gmcc@class{mwrep}}% If you need chapters, this option chooses an MW class that corresponds to `report`,

`mwbk` 210 \gm@DOX{mwbk} {\gmcc@class{mwbk}}% and this MW class corresponds to `book`.

`article` 213 \gm@DOX{article} {\gmcc@class{article}}% you can also *choose* `article`. A meta-
remark: When I tried to do the most natural thing, to `\ExecuteOptionsX` inside such declared option, an error occurred: 'undefined control sequence
% \XKV@resa-> @nil'.

`outeroff` 221 \gm@DOX{outeroff} {\let\outer\relax}% This option allows `\outer`-prefixed
macros to be `gmdoc`-processed with all the bells and whistles.

```
\if@debug 225 \newif\if@debug
```

`debug` 227 \gm@DOX{debug} {\@debugtrue}% This option causes `trace` to be loaded and the
Boolean switch of this option may be used to hide some things needed only
while debugging.

`noindex` 232 \gm@DOX{noindex} {%
233 \PassOptionsToPackage{noindex}{gmdoc}}% This option turns the writing
out to `.idx` file off.

```
\if@gmccnochanges 237 \newif\if@gmccnochanges
```

`nochanges` 239 \gm@DOX{nochanges} {\@gmccnochangestrue}% This option turns the writing
out to `.glo` file off.

Since version 0.99g the `gmdoc` bundle goes $X\!T\!\!E\!X$. That means that if $X\!T\!\!E\!X$ is detected, we may load the `fontspec` package and the other two of basic three $X\!T\!\!E\!X$ -related, and then we `\fontspec` the fonts. But the default remains the old way and the new way is given as the option below.

```
\ifgmcc@oldfonts 263 \newif\ifgmcc@oldfonts
```

`sysfonts` 265 \gm@DOX{sysfonts} {\gmcc@oldfontsfalse}

`mptt` 274 \gm@DOX{mptt}[17]{\relax}% now a no-op, left only for backwards compatibility. It was an option for setting the `marginpar` typewriter font.

```
\gmcc@tout 284 \def\gmcc@tout#1{\typeout{^\^J@@@@\gmdocc@class:#1^\^J}}
```

```

\gmcc@setfont 286 \def\gmcc@setfont#1{%
287   \gmcc@oldfontsfalse% note that if we are not in XETEX, this switch will be
288   turned true in line 415
289   \AtEndOfClass{%
290     \ifdefined\zf@init\afterfi{%
291       \gmcc@tout{Main font set to #1}%
292       \def\gmcc@dff{Numbers={OldStyle, Proportional}}
293       \@xa\setmainfont@\xa[\gmcc@dff, Mapping=tex-text]{%
294         #1}%
295       \@xa\defaultfontfeatures@\xa{\gmcc@dff, Scale=MatchLowercase}%
296       when put before \setmainfont,
297       \gmath
298       \def\LineNumFont{%
299         \normalfont\scriptsize\addfontfeature{%
300           Numbers=Monospaced}}%
301       }%
302     \else\afterfi{\gmcc@tout{I~can~set~main~font~to~#1~only~in~XeTeX/fontspec}}%
303     \fi
304   }%
305   }%
306   }%
307   }%
308   }%
309   \else\afterfi{\gmcc@tout{I~can~set~main~font~to~#1~only~in~XeTeX/fontspec}}%
310   \fi
311   }%
312   }%
313   }%
314   \gm@DOX{minion}{\gmcc@setfont{Minion Pro}}
315   \gm@DOX{pagella}{\gmcc@setfont{TeX Gyre Pagella}}
316   }%
317   }%
318   \gm@DOX{cronos}{%
319     \AtEndOfClass{\setsansfont[Mapping=tex-text]{Cronos Pro}}}
320   \gm@DOX{trebuchet}{%
321     \AtEndOfClass{\setsansfont[Mapping=tex-text]{Trebuchet MS}}}
322   }%
323   \gm@DOX{myriad}{%
324     \AtEndOfClass{\setsansfont[Mapping=text-text]{Myriad Web Pro}}}
325   }%
326   \gm@DOX{lsv}{%
327     \AtEndOfClass{\setsansfont[Mapping=tex-text]{Lucida Sans Unicode}}}
328   }%
329   }%
330   \gm@DOX{cursor}{%
331     \AtEndOfClass{%
332       \setmonofont[FakeBold=2.5, BoldFeatures={FakeBold=0},
333       FakeStretch=0.87, Ligatures=NoCommon
334       ]{TeX Gyre Cursor}%
335       \def\marginpartt{\tt\addfontfeature{FakeBold=2,
336         FakeStretch=0.609}%
337       \color{black}}% to provide proper color when marginpar occurs be-
338       tween lines that break a coloured text.
339       }%
340       \def\narrativett{\ttfamily\addfontfeature{FakeBold=1}}%
341       \let\UrlFont\narrativett
342       }% of \AtEndOfClass.
343       }% of the cursor option.
344       }%
345       }%
346       }%
347       }%
348       }%
349       }%
350       }%
351       }%
352       }%
353       }%
354       }%
355       }%
356       }%
357       }%
358       }%
359       }%
360       }%
361       }%
362       }%
363       }%
364       }%
365       }%
366       }%
367       }%
368       }%
369       }%
370       }%
371       }%
372       }%
373       }%
374       }%
375       }%
376       }%
377       }%
378       }%
379       }%
380       }%
381       }%
382       }%
383       }%
384       }%
385       }%
386       }%
387       }%
388       }%
389       }%
390       }%
391       }%
392       }%
393       }%
394       }%
395       }%
396       }%
397       }%
398       }%
399       }%
400       }%
401       }%
402       }%
403       }%
404       }%
405       }%
406       }%
407       }%
408       }%
409       }%
410       }%
411       }%
412       }%
413       }%
414       }%
415       }%
416       }%
417       }%
418       }%
419       }%
420       }%
421       }%
422       }%
423       }%
424       }%
425       }%
426       }%
427       }%
428       }%
429       }%
430       }%
431       }%
432       }%
433       }%
434       }%
435       }%
436       }%
437       }%
438       }%
439       }%
440       }%
441       }%
442       }%
443       }%
444       }%
445       }%
446       }%
447       }%
448       }%
449       }%
450       }%
451       }%
452       }%
453       }%
454       }%
455       }%
456       }%
457       }%
458       }%
459       }%
460       }%
461       }%
462       }%
463       }%
464       }%
465       }%
466       }%
467       }%
468       }%
469       }%
470       }%
471       }%
472       }%
473       }%
474       }%
475       }%
476       }%
477       }%
478       }%
479       }%
480       }%
481       }%
482       }%
483       }%
484       }%
485       }%
486       }%
487       }%
488       }%
489       }%
490       }%
491       }%
492       }%
493       }%
494       }%
495       }%
496       }%
497       }%
498       }%
499       }%
500       }%
501       }%
502       }%
503       }%
504       }%
505       }%
506       }%
507       }%
508       }%
509       }%
510       }%
511       }%
512       }%
513       }%
514       }%
515       }%
516       }%
517       }%
518       }%
519       }%
520       }%
521       }%
522       }%
523       }%
524       }%
525       }%
526       }%
527       }%
528       }%
529       }%
530       }%
531       }%
532       }%
533       }%
534       }%
535       }%
536       }%
537       }%
538       }%
539       }%
540       }%
541       }%
542       }%
543       }%
544       }%
545       }%
546       }%
547       }%
548       }%
549       }%
550       }%
551       }%
552       }%
553       }%
554       }%
555       }%
556       }%
557       }%
558       }%
559       }%
560       }%
561       }%
562       }%
563       }%
564       }%
565       }%
566       }%
567       }%
568       }%
569       }%
570       }%
571       }%
572       }%
573       }%
574       }%
575       }%
576       }%
577       }%
578       }%
579       }%
580       }%
581       }%
582       }%
583       }%
584       }%
585       }%
586       }%
587       }%
588       }%
589       }%
590       }%
591       }%
592       }%
593       }%
594       }%
595       }%
596       }%
597       }%
598       }%
599       }%
599       }%
600       }%
601       }%
602       }%
603       }%
604       }%
605       }%
606       }%
607       }%
608       }%
609       }%
610       }%
611       }%
612       }%
613       }%
614       }%
615       }%
616       }%
617       }%
618       }%
619       }%
620       }%
621       }%
622       }%
623       }%
624       }%
625       }%
626       }%
627       }%
628       }%
629       }%
630       }%
631       }%
632       }%
633       }%
634       }%
635       }%
636       }%
637       }%
638       }%
639       }%
640       }%
641       }%
642       }%
643       }%
644       }%
645       }%
646       }%
647       }%
648       }%
649       }%
650       }%
651       }%
652       }%
653       }%
654       }%
655       }%
656       }%
657       }%
658       }%
659       }%
660       }%
661       }%
662       }%
663       }%
664       }%
665       }%
666       }%
667       }%
668       }%
669       }%
670       }%
671       }%
672       }%
673       }%
674       }%
675       }%
676       }%
677       }%
678       }%
679       }%
680       }%
681       }%
682       }%
683       }%
684       }%
685       }%
686       }%
687       }%
688       }%
689       }%
690       }%
691       }%
692       }%
693       }%
694       }%
695       }%
696       }%
697       }%
698       }%
699       }%
699       }%
700       }%
701       }%
702       }%
703       }%
704       }%
705       }%
706       }%
707       }%
708       }%
709       }%
709       }%
710       }%
711       }%
712       }%
713       }%
714       }%
715       }%
716       }%
717       }%
718       }%
719       }%
719       }%
720       }%
721       }%
722       }%
723       }%
724       }%
725       }%
726       }%
727       }%
728       }%
729       }%
729       }%
730       }%
731       }%
732       }%
733       }%
734       }%
735       }%
736       }%
737       }%
738       }%
739       }%
739       }%
740       }%
741       }%
742       }%
743       }%
744       }%
745       }%
746       }%
747       }%
748       }%
749       }%
749       }%
750       }%
751       }%
752       }%
753       }%
754       }%
755       }%
756       }%
757       }%
758       }%
759       }%
759       }%
760       }%
761       }%
762       }%
763       }%
764       }%
765       }%
766       }%
767       }%
768       }%
769       }%
769       }%
770       }%
771       }%
772       }%
773       }%
774       }%
775       }%
776       }%
777       }%
778       }%
779       }%
779       }%
780       }%
781       }%
782       }%
783       }%
784       }%
785       }%
786       }%
787       }%
788       }%
789       }%
789       }%
790       }%
791       }%
792       }%
793       }%
794       }%
795       }%
796       }%
797       }%
798       }%
799       }%
799       }%
800       }%
801       }%
802       }%
803       }%
804       }%
805       }%
806       }%
807       }%
808       }%
809       }%
809       }%
810       }%
811       }%
812       }%
813       }%
814       }%
815       }%
816       }%
817       }%
818       }%
819       }%
819       }%
820       }%
821       }%
822       }%
823       }%
824       }%
825       }%
826       }%
827       }%
828       }%
829       }%
829       }%
830       }%
831       }%
832       }%
833       }%
834       }%
835       }%
836       }%
837       }%
838       }%
839       }%
839       }%
840       }%
841       }%
842       }%
843       }%
844       }%
845       }%
846       }%
847       }%
848       }%
849       }%
849       }%
850       }%
851       }%
852       }%
853       }%
854       }%
855       }%
856       }%
857       }%
858       }%
859       }%
859       }%
860       }%
861       }%
862       }%
863       }%
864       }%
865       }%
866       }%
867       }%
868       }%
869       }%
869       }%
870       }%
871       }%
872       }%
873       }%
874       }%
875       }%
876       }%
877       }%
878       }%
879       }%
879       }%
880       }%
881       }%
882       }%
883       }%
884       }%
885       }%
886       }%
887       }%
888       }%
889       }%
889       }%
890       }%
891       }%
892       }%
893       }%
894       }%
895       }%
896       }%
897       }%
898       }%
899       }%
899       }%
900       }%
901       }%
902       }%
903       }%
904       }%
905       }%
906       }%
907       }%
908       }%
909       }%
909       }%
910       }%
911       }%
912       }%
913       }%
914       }%
915       }%
916       }%
917       }%
918       }%
919       }%
919       }%
920       }%
921       }%
922       }%
923       }%
924       }%
925       }%
926       }%
927       }%
928       }%
929       }%
929       }%
930       }%
931       }%
932       }%
933       }%
934       }%
935       }%
936       }%
937       }%
938       }%
939       }%
939       }%
940       }%
941       }%
942       }%
943       }%
944       }%
945       }%
946       }%
947       }%
948       }%
949       }%
949       }%
950       }%
951       }%
952       }%
953       }%
954       }%
955       }%
956       }%
957       }%
958       }%
959       }%
959       }%
960       }%
961       }%
962       }%
963       }%
964       }%
965       }%
966       }%
967       }%
968       }%
969       }%
969       }%
970       }%
971       }%
972       }%
973       }%
974       }%
975       }%
976       }%
977       }%
978       }%
979       }%
979       }%
980       }%
981       }%
982       }%
983       }%
984       }%
985       }%
986       }%
987       }%
988       }%
989       }%
989       }%
990       }%
991       }%
992       }%
993       }%
994       }%
995       }%
996       }%
997       }%
998       }%
999       }%
999       }%
1000      }%
1001      }%
1002      }%
1003      }%
1004      }%
1005      }%
1006      }%
1007      }%
1008      }%
1009      }%
1009      }%
1010      }%
1011      }%
1012      }%
1013      }%
1014      }%
1015      }%
1016      }%
1017      }%
1018      }%
1019      }%
1019      }%
1020      }%
1021      }%
1022      }%
1023      }%
1024      }%
1025      }%
1026      }%
1027      }%
1028      }%
1029      }%
1029      }%
1030      }%
1031      }%
1032      }%
1033      }%
1034      }%
1035      }%
1036      }%
1037      }%
1038      }%
1039      }%
1039      }%
1040      }%
1041      }%
1042      }%
1043      }%
1044      }%
1045      }%
1046      }%
1047      }%
1048      }%
1049      }%
1049      }%
1050      }%
1051      }%
1052      }%
1053      }%
1054      }%
1055      }%
1056      }%
1057      }%
1058      }%
1059      }%
1059      }%
1060      }%
1061      }%
1062      }%
1063      }%
1064      }%
1065      }%
1066      }%
1067      }%
1068      }%
1069      }%
1069      }%
1070      }%
1071      }%
1072      }%
1073      }%
1074      }%
1075      }%
1076      }%
1077      }%
1078      }%
1079      }%
1079      }%
1080      }%
1081      }%
1082      }%
1083      }%
1084      }%
1085      }%
1086      }%
1087      }%
1088      }%
1089      }%
1089      }%
1090      }%
1091      }%
1092      }%
1093      }%
1094      }%
1095      }%
1096      }%
1097      }%
1098      }%
1099      }%
1099      }%
1100      }%
1101      }%
1102      }%
1103      }%
1104      }%
1105      }%
1106      }%
1107      }%
1108      }%
1109      }%
1109      }%
1110      }%
1111      }%
1112      }%
1113      }%
1114      }%
1115      }%
1116      }%
1117      }%
1118      }%
1119      }%
1119      }%
1120      }%
1121      }%
1122      }%
1123      }%
1124      }%
1125      }%
1126      }%
1127      }%
1128      }%
1129      }%
1129      }%
1130      }%
1131      }%
1132      }%
1133      }%
1134      }%
1135      }%
1136      }%
1137      }%
1138      }%
1139      }%
1139      }%
1140      }%
1141      }%
1142      }%
1143      }%
1144      }%
1145      }%
1146      }%
1147      }%
1148      }%
1149      }%
1149      }%
1150      }%
1151      }%
1152      }%
1153      }%
1154      }%
1155      }%
1156      }%
1157      }%
1158      }%
1158      }%
1159      }%
1160      }%
1161      }%
1162      }%
1163      }%
1164      }%
1165      }%
1166      }%
1167      }%
1168      }%
1169      }%
1169      }%
1170      }%
1171      }%
1172      }%
1173      }%
1174      }%
1175      }%
1176      }%
1177      }%
1178      }%
1178      }%
1179      }%
1180      }%
1181      }%
1182      }%
1183      }%
1184      }%
1185      }%
1186      }%
1187      }%
1188      }%
1188      }%
1189      }%
1190      }%
1191      }%
1192      }%
1193      }%
1194      }%
1195      }%
1196      }%
1197      }%
1198      }%
1198      }%
1199      }%
1200      }%
1201      }%
1202      }%
1203      }%
1204      }%
1205      }%
1206      }%
1207      }%
1208      }%
1209      }%
1209      }%
1210      }%
1211      }%
1212      }%
1213      }%
1214      }%
1215      }%
1216      }%
1217      }%
1218      }%
1219      }%
1219      }%
1220      }%
1221      }%
1222      }%
1223      }%
1224      }%
1225      }%
1226      }%
1227      }%
1228      }%
1229      }%
1229      }%
1230      }%
1231      }%
1232      }%
1233      }%
1234      }%
1235      }%
1236      }%
1237      }%
1238      }%
1239      }%
1239      }%
1240      }%
1241      }%
1242      }%
1243      }%
1244      }%
1245      }%
1246      }%
1247      }%
1248      }%
1249      }%
1249      }%
1250      }%
1251      }%
1252      }%
1253      }%
1254      }%
1255      }%
1256      }%
1257      }%
1258      }%
1259      }%
1259      }%
1260      }%
1261      }%
1262      }%
1263      }%
1264      }%
1265      }%
1266      }%
1267      }%
1268      }%
1269      }%
1269      }%
1270      }%
1271      }%
1272      }%
1273      }%
1274      }%
1275      }%
1276      }%
1277      }%
1278      }%
1278      }%
1279      }%
1280      }%
1281      }%
1282      }%
1283      }%
1284      }%
1285      }%
1286      }%
1287      }%
1287      }%
1288      }%
1289      }%
1290      }%
1291      }%
1292      }%
1293      }%
1294      }%
1295      }%
1296      }%
1297      }%
1298      }%
1298      }%
1299      }%
1300      }%
1301      }%
1302      }%
1303      }%
1304      }%
1305      }%
1306      }%
1307      }%
1308      }%
1309      }%
1309      }%
1310      }%
1311      }%
1312      }%
1313      }%
1314      }%
1315      }%
1316      }%
1317      }%
1318      }%
1319      }%
1319      }%
1320      }%
1321      }%
1322      }%
1323      }%
1324      }%
1325      }%
1326      }%
1327      }%
1328      }%
1329      }%
1329      }%
1330      }%
1331      }%
1332      }%
1333      }%
1334      }%
1335      }%
1336      }%
1337      }%
1338      }%
1339      }%
1339      }%
1340      }%
1341      }%
1342      }%
1343      }%
1344      }%
1345      }%
1346      }%
1347      }%
1348      }%
1349      }%
1349      }%
1350      }%
1351      }%
1352      }%
1353      }%
1354      }%
1355      }%
1356      }%
1357      }%
1358      }%
1359      }%
1359      }%
1360      }%
1361      }%
1362      }%
1363      }%
1364      }%
1365      }%
1366      }%
1367      }%
1368      }%
1369      }%
1369      }%
1370      }%
1371      }%
1372      }%
1373      }%
1374      }%
1375      }%
1376      }%
1377      }%
1378      }%
1378      }%
1379      }%
1380      }%
1381      }%
1382      }%
1383      }%
1384      }%
1385      }%
1386      }%
1387      }%
1387      }%
1388      }%
1389      }%
1390      }%
1391      }%
1392      }%
1393      }%
1394      }%
1395      }%
1396      }%
1397      }%
1398      }%
1398      }%
1399      }%
1400      }%
1401      }%
1402      }%
1403      }%
1404      }%
1405      }%
1406      }%
1407      }%
1408      }%
1409      }%
1409      }%
1410      }%
1411      }%
1412      }%
1413      }%
1414      }%
1415      }%
1416      }%
1417      }%
1418      }%
1419      }%
1419      }%
1420      }%
1421      }%
1422      }%
1423      }%
1424      }%
1425      }%
1426      }%
1427      }%
1428      }%
1429      }%
1429      }%
1430      }%
1431      }%
1432      }%
1433      }%
1434      }%
1435      }%
1436      }%
1437      }%
1438      }%
1438      }%
1439      }%
1440      }%
1441      }%
1442      }%
1443      }%
1444      }%
1445      }%
1446      }%
1447      }%
1448      }%
1449      }%
1449      }%
1450      }%
1451      }%
1452      }%
1453      }%
1454      }%
1455      }%
1456      }%
1457      }%
1458      }%
1459      }%
1459      }%
1460      }%
1461      }%
1462      }%
1463      }%
1464      }%
1465      }%
1466      }%
1467      }%
1468      }%
1469      }%
1469      }%
1470      }%
1471      }%
1472      }%
1473      }%
1474      }%
1475      }%
1476      }%
1477      }%
1478      }%
1478      }%
1479      }%
1480      }%
1481      }%
1482      }%
1483      }%
1484      }%
1485      }%
1486      }%
1487      }%
1488      }%
1488      }%
1489      }%
1490      }%
1491      }%
1492      }%
1493      }%
1494      }%
1495      }%
1496      }%
1497      }%
1498      }%
1498      }%
1499      }%
1500      }%
1501      }%
1502      }%
1503      }%
1504      }%
1505      }%
1506      }%
1507      }%
1508      }%
1509      }%
1509      }%
1510      }%
1511      }%
1512      }%
1513      }%
1514      }%
1515      }%
1516      }%
1517      }%
1518      }%
1519      }%
1519      }%
1520      }%
1521      }%
1522      }%
1523      }%
1524      }%
1525      }%
1526      }%
1527      }%
1528      }%
1529      }%
1529      }%
1530      }%
1531      }%
1532      }%
1533      }%
1534      }%
1535      }%
1536      }%
1537      }%
1538      }%
1538      }%
1539      }%
1540      }%
1541      }%
1542      }%
1543      }%
1544      }%
1545      }%
1546      }%
1547      }%
1548      }%
1549      }%
1549      }%
1550      }%
1551      }%
1552      }%
1553      }%
1554      }%
1555      }%
1556      }%
1557      }%
1558      }%
1559      }%
1559      }%
1560      }%
1561      }%
1562      }%
1563      }%
1564      }%
1565      }%
1566      }%
1567      }%
1568      }%
1569      }%
1569      }%
1570      }%
1571      }%
1572      }%
1573      }%
1574      }%
1575      }%
1576      }%
1577      }%
1578      }%
1578      }%
1579      }%
1580      }%
1581      }%
1582      }%
1583      }%
1584      }%
1585      }%
1586      }%
1587      }%
1588      }%
1588      }%
1589      }%
1590      }%
1591      }%
1592      }%
1593      }%
1594      }%
1595      }%
1596      }%
1597      }%
1598      }%
1598      }%
1599      }%
1600      }%
1601      }%
1602      }%
1603      }%
1604      }%
1605      }%
1606      }%
1607      }%
1608      }%
1609      }%
1609      }%
1610      }%
1611      }%
1612      }%
1613      }%
1614      }%
1615      }%
1616      }%
1617      }%
1618      }%
1619      }%
1619      }%
1620      }%
1621      }%
1622      }%
1623      }%
1624      }%
1625      }%
1626      }%
1627      }%
1628      }%
1629      }%
1629      }%
1630      }%
1631      }%
1632      }%
1633      }%
1634      }%
1635      }%
1636      }%
1637      }%
1638      }%
1638      }%
1639      }%
1640      }%
1641      }%
1642      }%
1643      }%
1644      }%
1645      }%
1646      }%
1647      }%
1648      }%
1648      }%
1649      }%
1650      }%
1651      }%
1652      }%
1653      }%
1654      }%
1655      }%
1656      }%
1657      }%
1658      }%
1659      }%
1659      }%
1660      }%
1661      }%
1662      }%
1663      }%
1664      }%
1665      }%
1666      }%
1667      }%
1668      }%
1669      }%
1669      }%
1670      }%
1671      }%
1672      }%
1673      }%
1674      }%
1675      }%
1676      }%
1677      }%
1678      }%
1678      }%
1679      }%
1680      }%
1681      }%
1682      }%
1683      }%
1684      }%
1685      }%
1686      }%
1687      }%
1688      }%
1688      }%
1689      }%
1690      }%
1691      }%
1692      }%
1693      }%
1694      }%
1695      }%
1696      }%
1697      }%
1698      }%
1698      }%
1699      }%
1700      }%
1701      }%
1702      }%
1703      }%
1704      }%
1705      }%
1706      }%
1707      }%
1708      }%
1709      }%
1709      }%
1710      }%
1711      }%
1712      }%
1713      }%
1714      }%
1715      }%
1716      }%
1717      }%
1718      }%
1719      }%
1719      }%
1720      }%
1721      }%
1722      }%
1723      }%
1724      }%
1725      }%
1726      }%
1727      }%
1728      }%
1729      }%
1729      }%
1730      }%
1731      }%
1732      }%
1733      }%
1734      }%
1735      }%
1736      }%
1737      }%
1738      }%
1738      }%
1739      }%
1740      }%
1741      }%
1742      }%
1743      }%
1744      }%
1745      }%
1746      }%
1747      }%
1748      }%
1748      }%
1749      }%
1750      }%
1751      }%
1752      }%
1753      }%
1754      }%
1755      }%
1756      }%
1757      }%
1758      }%
1759      }%
1759      }%
1760      }%
1761      }%
1762      }%
1763      }%
1764      }%
1765      }%
1766      }%
1767      }%
1768      }%
1769      }%
1769      }%
1770      }%
1771      }%
1772      }%
1773      }%
1774      }%
1775      }%
1776      }%
1777      }%
1778      }%
1778      }%
1779      }%
1780      }%
1781      }%
1782      }%
1783      }%
1784      }%
1785      }%
1786      }%
1787      }%
1788      }%
1788      }%
1789      }%
1790      }%
1791      }%
1792      }%
1793      }%
1794      }%
1795      }%
1796      }%
1797      }%
1798      }%
1798      }%
1799      }%
1800      }%
1801      }%
1802      }%
1803      }%
1804      }%
1805      }%
1806      }%
1807      }%
1808      }%
1809      }%
1809      }%
1810      }%
1811      }%
1812      }%
1813      }%
1814      }%
1815      }%
1816      }%
1817      }%
1818      }%
1819      }%
1819      }%
1820      }%
1821      }%
1822      }%
1823      }%
1824      }%
1825      }%
1826      }%
1827      }%
1828      }%
1828      }%
1829      }%
1830      }%
1831      }%
1832      }%
1833      }%
1834      }%
1835      }%
1836      }%
1837      }%
1838      }%
1838      }%
1839      }%
1840      }%
1841      }%
1842      }%
1843      }%
1844      }%
1845      }%
1846      }%
1847      }%
1848      }%
1848      }%
1849      }%
1850      }%
1851      }%
1852      }%
1853      }%
1854      }%
1855      }%
1856      }%
1857      }%
1858      }%
1859      }%
1859      }%
1860      }%
1861      }%
1862      }%
1863      }%
1864      }%
1865      }%
1866      }%
1867      }%
1868      }%
1869      }%
1869      }%
1870      }%
1871      }%
1872      }%
1873      }%
1874      }%
1875      }%
1876      }%
1877      }%
1878      }%
1878      }%
1879      }%
1880      }%
1881      }%
1882      }%
1883      }%
1884      }%
1885      }%
1886      }%
1887      }%
1888      }%
1888      }%
1889      }%
1890      }%
1891      }%
1892      }%
1893      }%
1894      }%
1895      }%
1896      }%
1897      }%
1898      }%
1898      }%
1899      }%
1899      }%
1900      }%
1901      }%
1902      }%
1903      }%
1904      }%
1905      }%
1906      }%
1907      }%
1908      }%
1909      }%
1909      }%
1910      }%
1911      }%
1912      }%
1913      }%
1914      }%
1915      }%
1916      }%
1917      }%
1918      }%
1919      }%
1919      }%
1920      }%
1921      }%
1922      }%
1923      }%
1924      }%
1925      }%
1926      }%
1927      }%
1928      }%
1928      }%
1929      }%
1929      }%
1930      }%
1931      }%
1932      }%
1933      }%
1934      }%
1935      }%
1936      }%
1937      }%
1938      }%
1938      }%
1939      }%
1940      }%
1941      }%
1942      }%
1943      }%
1944      }%
1945      }%
1946      }%
1947      }%
1948      }%
1948      }%
1949      }%
1949      }%
1950      }%
1951      }%
1952      }%
1953      }%
1954      }%
195
```

```

367 \DeclareOptionX*{\PassOptionsToPackage{\CurrentOption}{gmdoc}}
369 \ProcessOptionsX[gmcc]<>
386 \ifgmcc@mwcls
387   \IfFileExists{\gmcc@CLASS.cls}{}{\gmcc@mwclsfalse}%
387     As announced,
387     we do the ontological test to any mwcls.
389 \fi
390 \ifgmcc@mwcls
394   \LoadClass[fleqn,oneside,noindentfirst,11pt,%
394     withmarginpar,
395     sfheadings]{\gmcc@CLASS}%
398 \else
399   \LoadClass[fleqn,11pt]{article}%
399     Otherwise the standard article is
399     loaded.
401 \fi
408 \RequirePackage{gmutils}[2008/10/08]%
408   we load it early to provide \ifXeTeX,
408   but after loading the base class since this package redefines some environments.
412 \ifgmcc@mwcls\afterfi\ParanoidPostsec\fi
415 \@ifXeTeX{}{\gmcc@oldfontstrue}
418 \AtBeginDocument{\mathindent=\CodeIndent}

```

The `fleqn` option makes displayed formulæ be flushed left and `\mathindent` is their indentation. Therefore we ensure it is always equal `\CodeIndent` just like `\leftskip` in `verbatim`. Thanks to that and the `\edverbs` declaration below you may display single `verbatim` lines with `\[... \]`:

```

\[[\verb|\stuff|]\].
426 \ifgmcc@oldfonts
427   \IfFileExists{lmodern.sty}%
427     We also examine the ontological status of
427     this package
429   \RequirePackage{lmodern}%
429     and if it shows to be satisfactory (the package
429     shows to be), we load it and set the proper font encoding.
432   \RequirePackage[T1]{fontenc}%
433 }{}%

```

A couple of diacritics I met while `gmdocing` these files and The Source etc. Some why the accents didn't want to work at my \LaTeX settings so below I define them for \LaTeX as respective chars.

```

\grave 437 \def\grave{\`a}%
\acute 438 \def\acute{\^c}%
\acute 439 \def\acute{\^e}%
\idiaeres 440 \def\idiaeres{"\i}%
\nacute 441 \def\nacute{\^n}%
\circum 442 \def\circum{\^o}%
\oumlaut 443 \def\oumlaut{\^o}%
\uumlaut 444 \def\uumlaut{\^u}%
445 \else% this case happens only with \LaTeX.
446   \let\do\relaxen
447   \do\Finv\do\Game\do\beth\do\gimel\do\daleth%
447     these five caused the
447     'already defined' error.

```

```

449 \let \@zf@euenctrue \zf@euencfalse
450   \XeTeXthree%
\grave{a} \def \grave{\char"00E0}%
\acute{a} \def \acute{\char"0107}%
\grave{e} \def \grave{\char"00E9}%
\idiaeres \def \idiaeres {\char"00EF}%
\acute{n} \def \acute{\char"0144}%
\oumlaut \def \oumlaut {\char"00F6}%
\uumlaut \def \uumlaut {\char"00FC}%
\circum{a} \def \circum{\char"00F4}%
\AtBeginDocument{%
\ae \def \ae{\char"00E6}%
\l \def \l{\char"0142}%
\oe \def \oe{\char"0153}%
}%
\fi

```

Now we set the page layout.

```

\gmdoccMargins@params
468 \RequirePackage{geometry}
469 \def \gmdoccMargins@params {{top=77pt, height=687pt, =53 lines but
      the lines option seems not to work 2007/11/15 with TeX Live 2007 and
      XeTeX0.996-patch1
      left=4cm, right=2.2cm} }
470 \def \gmdoccMargins{%
471   \xa\newgeometry{\gmdoccMargins@params}
472 \xa\geometry{\gmdoccMargins@params}
473 \if@debug% For debugging we load also the trace package that was very helpful to
      me.
474   \RequirePackage{trace}%
475   \errorcontextlines=100% And we set an error info parameter.
476 \fi
\ifdtraceon \newcommand*\ifdtraceon{\if@debug\afterfi\traceon\fi}
\ifdtraceoff \newcommand*\ifdtraceoff{\if@debug\traceoff\fi}

```

We load the core package:

```

477 \RequirePackage{gmdoc}
478 \ifgmcc@oldfonts
479   \ifpackageloaded{lmodern}{%
480     The Latin Modern font family provides
481     a light condensed typewriter font that seems to be the most suitable for the
482     marginpar CS marking.
\marginpartt 483   \def \marginpartt {\normalfont\fontseries{lc}\ttfamily}%
484 }%
485 \else
\marginpartt 486   \def \marginpartt {\fontspec{LMTypewriter10}%
487     LightCondensed}%
488 \fi
489 \raggedbottom
490 \setcounter{secnumdepth}{0}% We wish only the parts and chapters to be
491 numbered.

```


c. The gutils Package¹

Written by Grzegorz Murzynowski,
natror at o2 dot pl

© 2005, 2006, 2007, 2008, 2009, 2010 by Grzegorz Murzynowski.

This program is subject to the L^AT_EX Project Public License.

See [http://www.ctan.org/tex--archive/help/Catalogue/licenses.lppl.html](http://www.ctan.org/tex-archive/help/Catalogue/licenses.lppl.html) for the details of that license.

LPPL status: "author-maintained".

Many thanks to my T_EX Guru Marcin Woliński for his T_EXnical support.

```
106 \NeedsTeXFormat {LaTeX2e}
107 \ProvidesPackage {gutils}
108     [2010/03/04 v0.991 some rather TeXnical macros, some
      of them tricky (GM)]
```

Intro

The gutils.sty package provides some macros that are analogous to the standard L^AT_EX ones but extend their functionality, such as `\ifnextcat`, `\addtomacro` or `\begin{...}`. The others are just conveniences I like to use in all my TeX works, such as `\afterfi`, `\pk` or `\cs`.

I wouldn't say they are only for the package writers but I assume some nonzero (L^A)T_EX-awareness of the user.

For details just read the code part.

The remarks about installation and compiling of the documentation are analogous to those in the chapter gmdoc.sty and therefore omitted.

Contents of the gutils.zip archive

The distribution of the gutils package consists of the following three files and a TDS-compliant archive.

```
gutils.sty
README
gutils.pdf
gutils.tds.zip
```

```
180 \ifx \XeTeXversion \relax
181   \let \XeTeXversion \undefined% If someone earlier used
      % \ifundefined{\XeTeXversion} to test whether the engine is XETEX,
      then \XeTeXversion is defined in the sense of e-TEX tests. In that case
      we \let it to something really undefined. Well, we might keep sticking
```

¹ This file has version number v0.991 dated 2010/03/04.

to `\@ifundefined`, but it's a macro and it eats its arguments, freezing their catcodes, which is not what we want in line 6155.

```
188 \fi  
190 \ifdefined\XeTeXversion  
191 \XeTeXinputencoding(utf-8)% we use Unicode dashes later in this file.  
192 \fi% and if we are not in XETEX, we skip them thanks to XETEX-test.
```

Options

```
\ifgmu@quiet  
213 \newif\ifgmu@quiet  
quiet 215 \DeclareOption{quiet}{\gmu@quiettrue}  
217 \ProcessOptions
```

A couple of abbreviations

```
\@xa 222 \let\@xa\expandafter  
\@nx 223 \let\@nx\noexpand
```

```
\@xau 225 \def\@xau#1{\unexpanded\@xa{#1}}
```

Note that there's only one `\expandafter`, after `\unexpanded`. It's because `\unex-` panded expands expandable tokens and gobbles `\relax`s while looking for an opening brace or `\bgroup`. Note also that it *requires* a text in braces.

Note also that (since v0.991) this is a 1-parameter macro so doesn't expand subsequent tokens until meets a *(balanced text)* but just takes first single token or *(text)*.

```
\pdef 240 \def\pdef{\protected\def}
```

And this one is defined, I know, but it's not `\long` with the standard definition and I want to be able to `\gobble` a `\par` sometimes.

```
\gobble 247 \long\def\gobble#1{}  
\@gobble 249 \let\@gobble\gobble  
\gobbletwo 250 \let\gobbletwo\@gobbletwo
```

```
\provide 254 \long\pdef\provide#1{  
255   \ifdefined#1%  
256     \ifx\relax#1\afterfifi{\def#1}%  
257     \else\afterfifi{\gmu@gobdef}%  
258   \fi  
259   \else\afterfi{\def#1}%  
260   \fi}
```

```
\gmu@gobdef 263 \long\def\gmu@gobdef#1#{  
264   \def\gmu@tempa{}% it's a junk macro assignment to absorb possible prefixes.  
266   \@gobble}
```

```
\pprovide 269 \def\pprovide{\protected\provide}
```

Note that both `\provide` and `\pprovide` may be prefixed with `\global`, `\outer`, `\long` and `\protected` because the prefixes stick to `\def` because all before it is expandable. If the condition(s) is false (`#1` is defined) then the prefixes are absorbed by a junk assignment.

Note moreover that unlike L^AT_EX's `\provide` command, our `\(p)\provide` allow any parameters string just like `\def` (because they just *expand* to `\def`).

```

\@nameedef 282 \long\def\@nameedef#1#2{%
283   \@xa\edef\csname#1\endcsname{#2} }

\ifs as a four-argument LATEX command robust to unbalanced \ifs and \fis

\gmu@if 287 \long\def\gmu@if#1#2{%
288   \csname_if#1\endcsname#2\@xa\@firstoftwo\else\@xa%
      \@secondoftwo\fi
289 }

```

\firstofone and the queer \catcodes

Remember that once a macro's argument has been read, its \catcodes are assigned forever and ever. That's what is \firstofone for. It allows you to change the \catcodes locally for a definition *outside* the changed \catcodes' group. Just see the below usage of this macro 'with T_EX's eyes', as my T_EX Guru taught me.

```

300 \long\def\firstofone#1{#1}
\scantwo 304 \long\pdef\scantwo#1#2{
305   \begingroup\endlinechar\m@ne
306   \@xa\endgroup\scantokens{#1#2}%
307 }

\@firstofthree 309 \long\def\@firstofthree#1#2#3{#1}
\@secondofthree 310 \long\def\@secondofthree#1#2#3{#2}
\@thirdofthree 311 \long\def\@thirdofthree#1#2#3{#3}
\@secondoffive 312 \long\def\@secondoffive#1#2#3#4#5{#2}

```

In some \if[cat?] test I needed to look only at the first token of a tokens' string (first letter of a word usually) and to drop the rest of it. So I define a macro that expands to the first token (or {<text>}) of its argument.

```

\@firstofmany 319 \long\def\@firstofmany#1#2\@@nil{#1}
\@allbutfirstof 322 \long\def\@allbutfirstof#1#2\@@nil{#2}

```

\afterfi and pals

It happens from time to time that you have some sequence of macros in an \if... and you would like to expand \fi before expanding them (e.g., when the macros should take some tokens next to \fi... as their arguments. If you know how many macros are there, you may type a couple of \expandafters and not to care how terrible it looks. But if you don't know how many tokens will there be, you seem to be in a real trouble. There's the Knuthian trick with \next. And here another, revealed to me by my T_EX Guru.

I think the situations when the Knuthian (the former) trick is not available are rather seldom, but they are imaginable at least: the \next trick involves an assignment so it won't work e.g. in \edef.

But \afterfi and pals are sensitive to \fis that may occur in macros' arguments so probably the safest and expandable way is the \expandafter\@(first|second)oftwo trick.

```

\longafterfi 353 \def\longafterfi{%

```

```
\afterfi 354 \long\def\afterfi##1##2\fi{\fi##1}
355 \long\def\afterfi
```

And two more of that family:

```
\afterfifi 357 \long\def\afterfifi##1##2\fi##3\fi{\fi\fi##1}
\afterffffi 359 \long\def\afterffffi##1##2\fi##3\fi{\fi##1}
```

Notice the refined elegance of those macros, that cover both ‘then’ and ‘else’ cases thanks to #2 that is discarded.

```
\afterffffifi 363 \long\def\afterffffifi##1##2\fi##3\fi##4\fi{\fi##1}
\afterffffifi 364 \long\def\afterffffifi##1##2\fi##3\fi##4\fi{\fi\fi##1}
\afterffffifi 365 \long\def\afterffffifi##1##2\fi##3\fi##4\fi{\fi\fi\fi##1}
```

\foone

The next command, \foone, is intended as two-argument for shortening of the \begingroup...% \firstofone {\endgroup...} hack.

```
\foone 372 \long\def\foone##1{\begingroup##1\relax\egfirstofone}
\egfirstofone 374 \long\def\egfirstofone##1{\endgroup##1}
\fooatletter 376 \def\fooatletter{\foone\makeatletter}
```

\@ifempty, \IfAmong, \IfIntersect \@ifinmeaning

After a short deliberation I make the \IfAmong... \among and \IfIntersect macros’ names apeless since they are intended for the macro and class writers, at the same level of abstraction as \DeclareCommand.

```
\@ifempty 386 \long\pdef\@ifempty#1{%
    % #1 the token(s) we test, it may be just {},
    % #2 the stuff executed if #1 is empty (is {}),
    % #3 the stuff executed if #1 consists of at least one token.
\gmu@reserved@ 392 \def\gmu@reserved@##1{%
393 \ifx\gmu@reserved@\@empty\@xa\@firstoftwo
394 \else\@xa\@secondoftwo
395 \fi}
\@ifnonempty 397 \long\def\@ifnonempty#1#2#3{\@ifempty{#1}{#3}{#2}}
\IfAmong 399 \long\pdef\IfAmong#1\among#2{%
    % #1 the token(s) whose presence we check,
    % #2 the list of tokens in which we search #1,
    % #3 the ‘if found’ stuff,
    % #4 the ‘if not found’ stuff.
\gmu@among@ 408 \long\def\gmu@among@##1##2\gmu@among@{%
409 \@ifempty{##2}\@secondoftwo\@firstoftwo}%
410 \gmu@among@##2\@1\gmu@among@}
\IfIntersect 412 \long\def\IfIntersect#1#2{%
    % this (not expandable) macro checks whether
    % the list of tokens #1 and #2 have nonempty intersection and if so executes
    % \@firstoftwo or else \@secondoftwo, so it’s a 4-argument com-
    % mand:
    % #1 first list to match,
```

```

        % #2 second list to match,
        % #3 if match (nonempty intersection),
        % #4 if not match (empty intersection).
422  \let\IfIntersect@next \@secondoftwo
423  \gmu@foreach#1\gmu@foreach{%
424      \xa\IfAmong\gmu@forarg\among{\#2}
425      {\let\IfIntersect@next \@firstoftwo}{}%
426  \IfIntersect@next}

```

We need a macro that iterates over every token on a list. L^AT_EX's `\@for` iterates over a list separated with commas so it's not the case. Our macro is much simpler.

```

\gmu@foreach
  \gmu@forer
    \long\def\gmu@foreach#1\gmu@foreach#2{%
432      \long\def\gmu@forer##1{%
433          \ifx\gmu@foreach##1\else
434              \long\def\gmu@forarg{\##1}%
435              ##2%
436              \xa\gmu@forer
437              \fi}%
438      \gmu@forer#1\gmu@foreach}

@ifinmeaning
  \long\pdef\@ifinmeaning#1\of#2{%
        % #1 the token(s) whose presence we check,
        % #2 the macro in whose meaning we search #1 (the first token of this
        % argument is expanded one level with \expandafter),
        % #3 the 'if found' stuff,
        % #4 the 'if not found' stuff.
442 \def\gmu@reserveda{\IfAmong#1\among}%
458   \def\gmu@reserveda{\@xa\gmu@reserveda\@xa{\#2}}
459 }
```

Global Boolean switches

The `\newgif` declaration's effect is used even in the L^AT_EX 2_E source by redefining some particular user defined ifs (UD-ifs henceforth) step by step. The goal is to make the UD-if's assignment global. I needed it at least twice during gmdoc writing so I make it a macro. It's an almost verbatim copy of L^AT_EX's `\newif` modulo the letter *g* and the `\global` prefix. (File d: ltdefns.dtx Date: 2004/02/20 Version v1.3g, lines 139–150)

```

\newgif
  \pdef\newgif#1{%
472   {\escapechar\m@ne
473     \global\let#1\iffalse
474     \g@if#1\iftrue
475     \g@if#1\iffalse
476   }%
477 }
```

'Almost' is also in the detail that in this case, which deals with `\global` assignments, we don't have to bother with storing and restoring the value of `\escapechar`: we can do all the work inside a group.

```

\@gif
  \def\@gif#1#2{%
483   \protected\@xa\gdef\csname\@xa\@gobbletwo\string#1%
484   g% the letter g for '\global'.
485   \@xa\@gobbletwo\string#2\endcsname
486   {\global\let#1#2}}
```

```

489 \pdef\newif#1{%
  We not only make \newif \protected but also make it to
  define \protected assignments so that premature expansion doesn't
  affect \if... \fi nesting.
496   \count@\escapechar\escapechar\m@ne
497   \let#1\iffalse
498   \@if#1\iftrue
499   \@if#1\iffalse
500   \escapechar\count@}

\@if 502 \def\@if#1#2{%
503   \protected\@xa\def\csname\@xa\@gobbletwo\string#1%
504   \@xa\@gobbletwo\string#2\endcsname
505   {\let#1#2} }

\hidden@iffalse 508 \pdef\hidden@iffalse{\iffalse}
\hidden@iftrue 509 \pdef\hidden@iftrue{\iftrue}

```

After \newgif\iffoo you may type {\foogtrue} and the \iffoo switch becomes globally equal \iftrue. Simili modo \foogfalse. Note the letter g added to underline globalness of the assignment.

If for any reason, no matter how queer ;-) may it be, you need *both* global and local switchers of your \if..., declare it both with \newif and \newgif.

Note that it's just a shorthand. \global\if<switch>(true|false) does work as expected.

There's a trouble with \refstepcounter: defining \@currentlabel is local. So let's \def a \global version of \refstepcounter.

Warning. I use it because of very special reasons in gmdoc and in general it is probably not a good idea to make \refstepcounter global since it is contrary to the original L^AT_EX approach.

```
\grefstepcounter 530 \pdef\grefstepcounter#1{%
531   {\let\protected@edef=\protected@xdef\refstepcounter{#1}}}
```

Naïve first try \globaldefs=\tw@ raised an error unknown_{command}\reserved@e. The matter was to globalize \protected@edef of \@currentlabel.

Thanks to using the true \refstepcounter inside, it observes the change made to \refstepcounter by hyperref.

2008/08/10 I spent all the night debugging \penalty 10000 that was added after a hypertarget in vertical mode. I didn't dare to touch hyperref's guts, so I worked it around with ensuring every \grefstepcounter to be in hmode:

```
\horefstepcounter 545 \pdef\horefstepcounter#1{%
546   \ifhmode\leavevmode\fi\grefstepcounter{#1}}
```

By the way I read some lines from *The T_EXbook* and was reminded that \unskip strips any last skip, whether horizontal or vertical. And I use \unskip mostly to replace a blank space with some fixed skip. Therefore define

```
\hunskip 553 \pdef\hunskip{\ifhmode\unskip\fi}
```

Note the two macros defined above are \protected. I think it's a good idea to make \protected all the macros that contain assignments. There is one more thing with \ifhmode: it can be different at the point of \edef and at the point of execution.

Another shorthand. It may decrease a number of \expandafters e.g.

```
\glet 563 \def\glet{\global\let}
```

L^AT_EX provides a very useful `\g@addto@macro` macro that adds its second argument to the current definition of its first argument (works iff the first argument is a no argument macro). But I needed it some times in a document, where @ is not a letter. So:

```
\gaddtomacro 571 \let \gaddtomacro=\g@addto@macro
```

The redefining of the first argument of the above macro(s) is `\global`. What if we want it local? Here we are:

```
\addto@macro 576 \long\def\addto@macro#1#2{%
580   \edef#1{\@xa\unexpanded\@xa{#1}\unexpanded{#2}}% \edef and \un%
      expanded to double the hashes.
582 }
```

And for use in the very document,

```
\addtomacro 586 \let\addtomacro=\addto@macro
```

2008/08/09 I need to prepend something not add at the end—so

```
\prependtomacro 589 \long\def\prependtomacro#1#2{%
591   \edef#1{\unexpanded{#2}\@xa\unexpanded\@xa{#1}}}
```

Note that `\prependtomacro` can be prefixed.

```
\addtotoks 595 \long\def\addtotoks#1#2{%
596   #1=\@xa{\the#1#2}}
\@emptify 599 \newcommand*\@emptify[1]{\let#1=\empty}
\emptify 600 \@ifdefinable\emptify{\let\emptify\@emptify}
```

Note the two following commands are in fact one-argument.

```
\g@emptify 604 \newcommand*\g@emptify{\global\@emptify}
\gemptify 605 \@ifdefinable\gemptify{\let\gemptify\g@emptify}
\@relaxen 608 \newcommand\@relaxen[1]{\let#1=\relax}
\relaxen 609 \@ifdefinable\relaxen{\let\relaxen\@relaxen}
```

Note the two following commands are in fact one-argument.

```
\g@relaxen 613 \newcommand*\g@relaxen{\global\@relaxen}
\grelaxen 614 \@ifdefinable\grelaxen{\let\grelaxen\g@relaxen}
```

\gm@ifundefined—a test that doesn't create any hash entry unlike \@ifundefined

I define it under another name not redefine `\@ifundefined` because I can imagine an odd case when something works thanks to `\@ifundefined`'s 'relaxation' effect.

```
\gm@ifundefined 623 \long\def\gm@ifundefined#1{%
624   \ifcsname#1\endcsname% defined...
625     \relax% but only as \relax—then
626     2nd argument.
627     \afterfifi\@firstoftwo%
628   \else% defined and not \relax—then 3rd argument.
629     \afterfifi\@secondoftwo%
630   \fi
631   \else% not defined—then 3rd.
632     \afterfi\@firstoftwo%
```

```

638     \fi}
\gm@testdefined 641 \long\def\gm@testdefined#1\iftrue{%
  This is a macro that expands to \iftrue
  or \iffalse depending on whether its argument is defined in the LATEX
  sense. Its syntax requires an \iftrue to force balancing \ifs with \fis.
646   \csname
647   \ifdefined#1%
648     \ifx#1\relax
649       \iffalse%
650     \else\iftrue%
651     \fi
652   \else\iffalse%
653   \fi\endcsname
654 }

\gm@testundefined 656 \long\def\gm@testundefined#1\iftrue{%
  we expand the last macro two
  levels. We repeat the parameter string to force the proper syntax.
659   \@xa\@xa\@xa\unless\gm@testdefined#1\iftrue}

```

Some ‘other’ and active stuff

Here I define a couple of macros expanding to special chars made ‘other’. It’s important the CS are expandable and therefore they can occur e.g. inside \csname... \endcsname unlike e.g. CS’es \chardefed.

```

670 \foone{\catcode`\_=8_}
\subs 671 {\let\subs=_}

674 \foone{\catcode`^=7_}
\sups 675 {\let\sups=^}

677 \foone{@makeother\_}
\xiunder 678 {\def\xiunder{_}{

680 \let\all@unders\xiunder
681 \foone{\catcode`\_=8_}
682 {\addtomacro\all@unders{_}}
683 \foone{\catcode`\_=11_}
684 {\addtomacro\all@unders{_}}
685 \foone{\catcode`\_=13_}
686 {\addtomacro\all@unders{_}}

```

Now \all@unders bears underscores of categories 8, 11, 12 and 13.

```

\all@stars 690 \def\all@stars{*}
691 \foone{\catcode`*=11_}
692 {\addtomacro\all@stars{*}}
693 \foone{\catcode`*=13_}
694 {\addtomacro\all@stars{*}}

```

And \all@stars bears stars of categories 11, 12 and 13.

```

698 \ifdefined\XeTeXversion
699   \chardef\_="005F
700 \fi
702 \foone{@makeother`\}%
703 {\def\backquote{\`}}

```

```

706 \foone{\catcode`\[=1\@makeother\{
707   \catcode`\]=2\@makeother\}}%
708 [%  

\xiilbrace  

\xiirbrace  

709   \def\xiilbrace[]%  

710   \def\xiirbrace[]%  

711 ]% of \firstofone

```

Note that L^AT_EX's `\@charlb` and `\@charrb` are of catcode 11 ('letter'), cf. The L^AT_EX 2_E Source file k, lines 129–130.

Now, let's define such a smart `_` (underscore) which will be usual `_8` in the math mode and `_12` ('other') outside math.

```

722 \foone{\catcode`\_=\\active}
723 {%
\xsmartunder  

724   \newcommand*\smartunder{%
725     \catcode`\_=\\active
726     \def_{\ifmmode\sub\else\_\\fi}}}% We define it as \_ not just as
    % \xiounder because some font encodings don't have _ at the \char
    % \_ position.
732 \foone{\catcode`\!=0
733   \@makeother\\}
734 {! newcommand*\xiibackslash{}}

\xibackslash  

\bslash  

738 \let\bslash=\xiibackslash

\xipercent  

742 \foone{\@makeother\%}
743 {\def\xipercent{}}

\xiand  

746 \foone{\@makeother\&}%
747 {\def\xiand{\&}}


\xiinspace  

749 \foone{\@makeother\_\_}%
750 {\def\xiinspace{\_}}



\xiishash  

752 \foone{\@makeother\#}%
753 {\def\xiishash{\#}}


```

We introduce `\visiblespace` from Will Robertson's `xltextra` if available. It's not sufficient `\@ifpackageloaded{xltextra}` since `\xxt@visiblespace` is defined only unless `no-verb` option is set. 2008/08/06 I recognised the difference between `\xiispace` which has to be plain 'other' char (used in `\xiistring`) and something visible to be printed in any font.

```

\xiispace@fallback  

762 \AtBeginDocument{%
763   \ifdefined\xxt@visiblespace
764     \let\visiblespace\xxt@visiblespace
765     \def\xxt@visiblespace@fallback{%
766       \fontspec[Latin Modern Mono]\textvisiblespace}%
767     \else
768       \let\visiblespace\xiinspace
769     \fi}
772 \foone\obeyspaces{\def\gmu@activespace{\_}}
\gmu@activespace
774 \foone\obeylines{\def\activeM{\^M}}
\activeM

```

```
\@ifnextcat, \@ifnextac, catcode-independent \gm@ifstar,
\@ifnextsingle, \@ifnextgroup
```

As you guess, we `\def \@ifnextcat` à la `\@ifnextchar`, see L^AT_EX 2_& source dated 2003/12/01, file d, lines 253–271. The difference is in the kind of test used: while `\@ifnextchar` does `\ifx`, `\@ifnextcat` does `\ifcat` which means it looks not at the meaning of a token(s) but at their `\catcode`(s). As you (should) remember from *The TeXbook*, the former test doesn't expand macros while the latter does. But in `\@ifnextcat` the peeked token is protected against expanding by `\noexpand`. Note that the first parameter is not protected and therefore it shall be expanded if it's expandable. Because an assignment is involved, you can't test whether the next token is an active char. But you can test if the next token is `{`₁ or `}`₂: `\@ifnextcat \bgroup..., \@ifnextcat \egroup...`

```
\@ifnextcat 794 \long \pdef \@ifnextcat#1#2#3 {%
 798   \def \reserved@d{#1}%
 799   \def \reserved@a{#2}%
 800   \def \reserved@b{#3}%
 801   \futurelet \let@token \ifncat}

\ifncat 804 \def \ifncat {%
 805   \ifx \let@token \sptoken
 806     \let \reserved@c \xifncat
 807   \else
 808     \ifcat \reserved@d \nx \let@token
 809       \let \reserved@c \reserved@a
 810     \else
 811       \let \reserved@c \reserved@b
 812     \fi
 813   \fi
 814   \reserved@c}

816 {\def \:{\let \sptoken= } \global \: % this makes \sptoken a space
      token.

819 \def \:{\@xifncat} \xa \gdef \:{\futurelet \let@token %
  \ifncat}}
```

Note the trick to get a macro with no parameter and requiring a space after it. We do it inside a group not to spoil the general meaning of `\:` (which we extend later).

The next command provides the real `\if` test for the next token. It should be called `\@ifnextchar` but that name is assigned for the future `\ifx` text, as we know. Therefore we call it `\@ifnextif`.

Having `\@ifnextcat` defined, let's apply it immediately. Similar thing does probably the `xspace` package.

```
\spifletter 833 \def \spifletter {\@ifnextcat \a{\space}{}}

\@ifnextif 836 \long \pdef \@ifnextif#1#2#3 {%
  (the future token in \noexpanded, unlike #1)

 843   \def \reserved@d{#1}%
 844   \def \reserved@a{#2}%
 845   \def \reserved@b{#3}%
 846   \futurelet \let@token \ifnif}
```

```

\@ifnif 849 \def \@ifnif {%
850   \ifx \@let@token \@sptoken
851     \let \reserved@c \@xifnif
852   \else
853     \if \reserved@d @nx \let@token
854       \let \reserved@c \reserved@a
855     \else % #1 of \@ifnextif is not \if-equivalent the future token. But this
856       may be because the future token is active so it would be \if-equivalent
857       if not passed through \futurelet. Let's manage this case.
858     \begingroup
859     \edef \gmu@tempa {%
860       \lccode` \@nx~` \reserved@d
861     } \gmu@tempa
862     \lowercase{ \endgroup
863       \ifx~} \let@token
864       \let \reserved@c \reserved@a
865     \else
866       \let \reserved@c \reserved@b
867     \fi
868   \fi
869 \fi
870 \reserved@c
871
872 {\def \: {\let \@sptoken= } : % this makes \@sptoken a space token.
873 \def \: {@xifnif} \xa \def \: {\futurelet \let@token \@ifnif}}

```

But how to peek at the next token to check whether it's an active char? First, we look with \@ifnextcat whether there stands a group opener. We do that to avoid taking a whole {...} as the argument of the next macro, that doesn't use \futurelet but takes the next token as an argument, tests it and puts back intact.

```

\@ifnextac 887 \long \pdef \@ifnextac#1#2{%
888   \@ifnextsingle
889   {\gmu@ifnac{#1}{#2}}%
890   {#2}}
\gmu@ifnac 892 \long \def \gmu@ifnac#1#2#3{%
893   \ifcat \@nx~\@nx#3%
894     \xa \firstoftwo
895   \else \xa \secondoftwo
896   \fi {#1#3}{#2#3}}

```

Yes, it won't work for an active char \let to {, but it will work for an active char \let to a char of catcode ≠ 1. (Is there anybody on Earth who'd make an active char working as \bgroup not just recatcode it to ?)

Having defined such tools, let's redefine \gmu@ifstar to make it work with whatever catcode * may be. make a version of \gmu@ifstar that would work with *₁₁.

```

\gmu@lowstar 910 \pdef \gmu@lowstar {%
911   \iffontchar \font "22C6" \char "22C6" \else \gmu@lowstarfake \fi }% we
912   could define it to be expandable (with ^^^^22c6) but the only goal of it
913   would be allowing this low star in \csname ... \endcsname. But it would
914   be misleading (not everyone can easily distinguish * from *) so IMHO it's
915   better to raise an error should such an active occur in a \csname. This

```

macro is intended to be \let to an active star only in verbatims. For usual text there's

Commands around making star low (via \activation) are defined in line 7048 because they use \DeclareCommand.

```

924 \def\gmu@tempa{*}
925 \foone{\catcode`\\*=\\active}
926 {\def\gmu@tempb{*}%
  it's defined in line ?? to make * defined (when it was
  undefined, \newcommand's \gm@ifstar test turned true, the next unde-
  fined token was gobbled and raised an error).
930   \let*=\\gmu@lowstar}

934 \edef\gmu@tempa{%
935   \long\pdef{@nx\gm@ifstar##1##2}{%
936     @nx@ifnextif\gmu@tempa%
937     {@nx@firstoftwo{##1}}%
938     {%
939       @nx@ifnextchar@xa@nx\gmu@tempb
940       {@nx@firstoftwo{##1}{##2}}%
941     }%
942   }%
943 }% of \gm@ifstar.
944 }\gmu@tempa

```

A test whether we can pick a single token. We have to check whether we are not next to { and whether we are not next to }.

```

@ifnestsingle 951 \long\pdef@ifnestsingle#1#2{%
  This macro checks whether the next token
  is able to be picked or is it a braced list of tokens or is it a group closer so there's
  no token to be picked.
955   @ifnextcat\bgroup{#2}{%
956     @ifnextcat\egroup{#2}{%
958       {#1}}}

@ifnextgroup 960 \long\pdef@ifnextgroup#1#2{%
  Note this macro turns true both before a group
  opener and before a group closer.
962   @ifnestsingle{#2}{#1}}

```

now let's apply this to sth. useful (used in gmverse and in some typesetting, e.g. for prof. JSB).

```

ignoreactiveM 967 \pdef\ignoreactiveM{%
968   @ifnextgroup{}{\gmu@checkM}}

970 \foone\obeylines{%
  we know it's a single token since we use this macro only
  in @ifnextgroup's 'else'.
@gmu@checkM 972 \long\pdef@gmu@checkM#1{%
973   \ifx
974     #1@xa\ignoreactiveM%
975   \else@xa#1\fi{}}

```

Now, define a test that checks whether the next token is a genuine space, \square_{10} that is. First define a CS let such a space. The assignment needs a little trick (*The T_EXbook* appendix D) since \let's syntax includes one optional space after =.

```

985 \let\gmu@reserveda\*%
986 \def\*{%
987   \let\*\gmu@reserveda
988   \let\gm@letspace=\square}%

```

```

989 \star_{\%}
100 \def\@ifnextspace#1#2{%
101   \let\gmu@reserveda\star_{\%}
102   \def\star_{\%}{%
103     \let\star_{\%}\gmu@reserveda
104     \ifx\@let@token\gm@letspace\afterfi{#1}%
105     \else\afterfi{#2}%
106     \fi}%
107   \futurelet\@let@token\star_{\%}

```

First use of this macro is for an active – that expands to --- if followed by a space. Another to make dot checking whether is followed by ~ without gobbling the space if it occurs instead.

Now a test if the next token is an active line end. I use it in gmdoc and later in this package for active long dashes.

```

1008 \foone\obeylines{%
1009   \long\pdef\@ifnextMac#1#2{%
1010     \@ifnextchar^{M{#1}{#2}}{}}

```

Storing and restoring the meanings of CSes

First a Boolean switch of globalness of assignments and its verifier.

```

\ifgmu@SMglobal 1017 \newif\ifgmu@SMglobal
\SMglobal 1019 \pdef\SMglobal{\gmu@SMglobaltrue}

```

The subsequent commands are defined in such a way that you can ‘prefix’ them with \SMglobal to get global (re)storing.

A command to store the current meaning of a CS in another macro to temporarily redefine the CS and be able to set its original meaning back (when grouping is not recommended):

```

\StoreMacro 1030 \pdef\StoreMacro{%
1031   \begingroup\makeatletter\gm@ifstar\egStore@MacroSt%
   \egStore@Macro}

```

The unstarred version takes a CS and the starred version a text, which is intended for special control sequences. For storing environments there is a special command in line 1177.

```

\egStore@Macro 1036 \long\def\egStore@Macro#1{\endgroup\Store@Macro{#1}}
\egStore@MacroSt 1037 \long\def\egStore@MacroSt#1{\endgroup\Store@MacroSt{#1}}
\Store@Macro 1039 \long\def\Store@Macro#1{%
1040   \escapechar_2
1041   \ifgmu@SMglobal\afterfi\global\fi
1042   \@xa\let\csname_\gmu/store\string#1\endcsname#1%
1043   \global\gmu@SMglobalfalse}
\Store@MacroSt 1046 \long\def\Store@MacroSt#1{%
1047   \edef\gmu@smtempa{%
1048     \ifgmu@SMglobal\global\fi
1049     \@nx\let\@xa\@nx\csname/gmu/store\bslash#1\endcsname%
       we add backslash because to ensure compatibility

```

between `\(Re) StoreMacro` and `\(Re) StoreMacro*`, that is. to allow writing e.g. `\StoreMacro\kitten` and then `\RestoreMacro*\{kitten}` to restore the meaning of `\kitten`.

```

1055   \@xa \@nx \csname#1\endcsname}
1056   \gmu@smtempa
1057   \global\gmu@SMglobalfalse}%

```

we wish the globality to be just once.

We make the `\StoreMacro` command a three-step to allow usage of the most inner macro also in the next command.

The starred version, `\StoreMacro*` works with csnames (without the backslash). It's first used to store the meanings of robust commands, when you may need to store not only `\foo`, but also `\csname\foo\endcsname`.

The next command iterates over a list of CSes and stores each of them. The CS'es may be separated with commas but they don't have to.

```

\StoreMacros 1073 \long \pdef \StoreMacros {\begingroup \makeatletter \Store@Macros}
\Store@Macros 1074 \long \def \Store@Macros#1 {\endgroup
1075   \gmu@setsetSMglobal
1076   \let \gml@StoreCS \Store@Macro
1077   \gml@storemacros#1.}

\gmu@setsetSMglobal 1080 \def \gmu@setsetSMglobal {%
1081   \if \gmu@SMglobal
1082     \let \gmu@setSMglobal \gmu@SMglobaltrue
1083   \else
1084     \let \gmu@setSMglobal \gmu@SMglobalfalse
1085   \fi}

```

And the inner iterating macro:

```

\gml@storemacros 1088 \long \def \gml@storemacros#1 {%
\gmu@reserveda 1089   \def \gmu@reserveda {\@nx#1}% My TeX Guru's trick to deal with \fi and
                      such, i.e., to hide #1 from TeX when it is processing a test's branch without
                      expanding.
1092   \if \gmu@reserveda.% a dot finishes storing.
1093     \global\gmu@SMglobalfalse
1094   \else
1095     \if \gmu@reserveda,% The list this macro is put before may contain com-
                      mas and that's O.K., we just continue the work.
1096       \afterfifi \gml@storemacros
1097   \else% what is else this shall be stored.
1098     \gml@StoreCS{\#1}% we use a particular CS to map \let it both to the
                      storing macro as above and to the restoring one as below.
1099     \afterfifi{\gmu@setSMglobal \gml@storemacros}%
1100   \fi
1104 \fi}

```

And for the restoring

```

\RestoreMacro 1110 \pdef \RestoreMacro {%
1111   \begingroup \makeatletter \gm@ifstar \egRestore@MacroSt%
1112     \egRestore@Macro}
\egRestore@Macro 1113 \long \def \egRestore@Macro#1 {\endgroup \Restore@Macro{\#1}}
\egRestore@MacroSt 1114 \long \def \egRestore@MacroSt#1 {\endgroup \Restore@MacroSt{\#1}}
\Restore@Macro 1116 \long \def \Restore@Macro#1 {%

```

```

1117  \escapechar92
1118  \gmu@ifstored#1{%
1119      \ifgmu@SMglobal\afterfi\global\fi
1120      \@xa\let\@xa#1\csname\gmu/store\string#1\endcsname
1121      \global\gmu@SMglobalfalse}%
1122  {\unless\ifgmu@quiet
1123      \PackageWarning{gmutils}{\@nx#1 is not stored, I do
1124          nothing with
1125          it}%
1126  }%
1127 }

\gmu@ifstored 1129 \long\def\gmu@ifstored#1#2#3{%
1130     \gm@ifundefined{/gmu/store}%
1131     \if\relax\@nx#1\else\bslash\fi
1132     \string#1\{\#3\}\{\#2\}%
1133 }

\gmu@storeifnotyet 1135 \long\pdef\gmu@storeifnotyet#1{%
1136     \if\relax\@nx#1% we check if it's a CS
1137     \gmu@ifstored{\#1}{}{\StoreMacro#1}%
1138     \fi}

\Restore@MacroSt 1141 \long\def\Restore@MacroSt#1{%
1142     \gm@ifundefined{/gmu/store\bslash#1}%
1143     {\unless\ifgmu@quiet
1144         \PackageWarning{gmutils}{\bslash#1 is not stored. I~do
1145             nothing}%
1146     }%
1147     {\edef\gmu@smtempa{%
1148         \ifgmu@SMglobal\global\fi
1149         \@nx\let\@xa\@nx\csname#1\endcsname
1150         \@xa\@nx\csname/gmu/store\bslash#1\endcsname}%
1151         cf. the com-
1152         mentary in line 1049.
1153         \gmu@smtempa}%
1154     \global\gmu@SMglobalfalse}
1155 }

\RestoreMacros 1155 \long\pdef\RestoreMacros{\begingroup\makeatletter%
1156     \Restore@Macros}

\Restore@Macros 1157 \long\def\Restore@Macros#1{\endgroup
1158     \gmu@setsetSMglobal
1159     \let\gml@StoreCS\Restore@Macro% we direct the core CS towards restor-
1160         ing and call the same iterating macro as in line 1077.
1161     \gml@storemacros#1.}


```

As you see, the `\RestoreMacros` command uses the same iterating macro inside, it only changes the meaning of the core macro.

And to restore *and* use immediately:

```

\StoredMacro 1168 \def\StoredMacro{\begingroup\makeatletter\Stored@Macro}
\Stored@Macro 1169 \long\def\Stored@Macro#1{\endgroup\Restore@Macro#1#1}

```

To be able to call a stored CS without restoring it.

```

\storedcsname 1172 \def\storedcsname#1{%

```

```
1173 \csname\gmu@store\bslash#1\endcsname}
```

2008/08/03 we need to store also an environment.

```
\StoreEnvironment 1177 \pdef\StoreEnvironment#1{%
 1179   \StoreMacro*{\#1}\StoreMacro*{end#1}}
```

```
\RestoreEnvironment 1181 \pdef\RestoreEnvironment#1{%
 1183   \RestoreMacro*{\#1}\RestoreMacro*{end#1}}
```

It happened (see the definition of \@docinclude in gmdoc.sty) that I needed to \relax a bunch of macros and restore them after some time. Because the macros were rather numerous and I wanted the code more readable, I wanted to \do them. After a proper defining of \do of course. So here is this proper definition of \do, provided as a macro (a declaration).

```
\StoringAndRelaxingDo 1198 \def\StoringAndRelaxingDo{%
 1199   \gmu@SMdo@setscope
 1200   \long\def\do##1{%
 1201     \gmu@SMdo@scope
 1202     @xa\let\csname\gmu/store\string##1\endcsname##1%
 1203     \gmu@SMdo@scope\let##1\relax}}
```

```
\gmu@SMdo@setscope 1205 \def\gmu@SMdo@setscope{%
 1206   \ifgmu@SMglobal\let\gmu@SMdo@scope\global
 1207   \else\let\gmu@SMdo@scope\relax
 1208   \fi
 1209   \global\gmu@SMglobalfalse}
```

And here is the counter-definition for restore.

```
\RestoringDo 1218 \long\def\RestoringDo{%
 1219   \gmu@SMdo@setscope
 1220   \long\def\do##1{%
 1221     \gmu@SMdo@scope
 1222     @xa\let@xa##1\csname\gmu/store\string##1\endcsname}}
```

Note that both \StoringAndRelaxingDo and \RestoringDo are sensitive to the \SMglobal ‘prefix’.

And to store a cs as explicitly named cs, i.e. to \let one csname another (\n@melet not \namelet because the latter is defined in Till Tantau’s beamer class another way) (both arguments should be text):

```
\n@melet 1230 \long\def\n@melet#1#2{%
 1231   \edef\gmu@nl@reserveda{%
 1232     \let@xa@\nx\csname#1\endcsname
 1233     @xa@\nx\csname#2\endcsname}%
 1234   \gmu@nl@reserveda}
```

The \global prefix doesn’t work with \n@melet so we define the alternative.

```
\gn@melet 1238 \long\def\gn@melet#1#2{%
 1239   \edef\gmu@nl@reserveda{%
 1240     \global\let@xa@\nx\csname#1\endcsname
 1241     @xa@\nx\csname#2\endcsname}%
 1242   \gmu@nl@reserveda}
```

```
\envirlet 1244 \long\pdef\envirlet#1#2{%
 1245   \n@melet{\#1}{\#2}}
```

```

1246     \n@melet {end#1} {end#2}%
1247 }

```

\DeclareCommand and \DeclareEnvironment

The code of this section is based on the `xparse` package version 0.17 dated 1999/09/10, the version available in TeX Live 2007-13, in Ubuntu packages at least. Originally considered a stub ‘im Erwartung’ (Schönberg) for the L^AT_EX3 bundle, it evolved to quite a nice tool I think.

I rewrote the ancient `xparse`’s code to use the usual catcodes (only with @ a letter) and not to use the `ldcsetup` package (which caused an error of undefined CS `\KV@def`). Then I added the `\protected` prefix to the first macro. And I added my concept of S/T arg. spec. And of b and B spec. And the Q argument type. And made `\DeclareEnvironment` makes the arguments passed also to the end macro.

After a short deliberation I rename the command to `\DeclareCommand` which is much shorter than original `\DeclareDocumentCommand` and more adequate at least in my case: I don’t only use this powerful declaration for ‘document’ commands.

In the args specification you can use:

- m for mandatory argument (braced or not),
- O {*default*} for optional argument (in square brackets) with default value *default* (which is passed as #*n* if the argument is missing),
- o equivalent of O {\NoValue}: optional argument (in square brackets) lack of which makes the `\NoValue` macro passed as #*n*. You can check it with `\If[No]Value(T|F|TF){\#<n>}`,
- s for optional star (if star is present, it becomes respective #*n*, or else `\NoValue` is at the #*n* (unlike in `xparse`!).
- S {*list*} {*default*} for a Single token, checks whether on the respective position there’s an unbraced token one of *list* and returns it on #*n* if present or {*default*} if absent. The {*default*} is a braced optional. If absent, `\NoValue` is the default.
- T as ‘Token’, just an alias for S.

Like in `xparse`, s is a shorthand for S {*}, but *unlike* in `xparse`, in the parsed arguments you get * or `\NoValue` as #*n*. You can now declare

```

\DeclareCommand{\mimbla}{S{+-}m}{%
  \leavevmode
  \ifx#1+\raise\fi
  \ifx#1-\lower\fi
  \ifx#1\NoValue\@tempdima=\fi
  7pt \hbox{#2}%
}

```

and after `\mimbla+{raised}` `\mimbla-{lowered}` `\mimbla{untouched}` get:

```

raised      untouched
lowered

```

This example is rather silly but shows that you spend only one # for two symbols while in `xparse` you would spend two. What if you wanted 10 options for the optional symbol? Here it’s no problem. And with any number *k* of tokens on the list the next # is always *n* + 1 not *n* + *k* + 1.

Q {*list*} as ‘seQuence’, a sequence of symbols from *list*; for the fundamental usage see line 2244. More clearly, an argument specified with Q {*tokens*} is a word over the alphabet *tokens* where `\` is ignored and shall not be passed in the respective #*n*.

- \DCcoordinate
- c for an optional argument in parentheses. Historically it comes from a ‘coordinate’ argument and may serve as such: if you declare \DCcoordinate, then its catcher will be redefined to check for presence of a comma and pass the argument as { % {<before comma>} {<after comma>} } if comma present. \NoValue is passed if absent. By default \DCnocoordinate is executed that defines the c type argument as just an optional in parentheses.
- \DCnocoordinate
- C {<default>} as c, with default value.
 - b for for an *optional* argument in curly braces. That’s strange for anyone acquainted with L^AT_EX and contrary to its basic convention, but practised by Till Tantau in the beamer class. If missing, \NoValue is passed as #<n> and therefore all the tests If[No]Value(T|F|TF) apply.
 - B {<default>} for a braced *optional* argument with the default value <default>.
 - K {<#-string>} [{<replacement>}] for a *mandatory* Knuthian delimited or undelimited macro parameters. This concept comes from Stephen Hicks’ suggestion at BachoT^EX 2009 of implementing arguments delimited with # { . So, in the mandatory first argument to K you give an arbitrary parameters string as described in Chapter 20 of *The T_EXbook*. If you don’t provide the replacement, only #1 of those parameters will be passed to the core macro of your command as #<n>. In both arguments you use single # char.
 - G {<pair of tokens>} [default value] a General catcher of an optional. For instance, to declare an optional in angles (used e.g. in Till Tantau’s beamer), declare G {<>} . Again, if second argument is absent, \NoValue is assumed.
 - A (for ‘angles’) a shorthand for G {<>} (accepts the second braced as the default value).
 - \afterassignment {<register>} a pseudo-argument that in fact *interrupts* parsing arguments to let an assignment to <register>. The default <register> is \@tempcnta. For example,
- ```
\DeclareCommand\llf{\afterassignment
 {{\lastlinefit\@tempcnta\par}}
```
- defines \llf to be a command that expects a value for a numerical assignment, assigns it to (\@tempcnta and then to) the \lastlinefit special register and executes \par with that setting inside a group.
- Actually, <register> may be empty {} and then each time our command is used the left side of the assignment has to be given explicitly and may even be different each time.
- Doesn’t add anything to the arguments’ list.  
*This pseudo-argument type should be considered experimental.*
- Note that you could limit acceptable values of a mandatory or an optional-in-brackets argument to a list inside definition of the command, using \IfAmong... \among... defined in line 399.
- The S/T, s and Q arguments are always ‘long’, they allow \par as their value that is. They have to be unbraced to be parsed so there’s no danger of “runaway argument”.
- By default, all the c, C, o, O, m, b and B are ‘short’, they don’t allow \par in them that is. Note however that \par is allowed in the default values. If you wish to allow \par for all the arguments, you can say \DeclareCommand\mycommand! ... — the optional ! makes all the arguments ‘long’. Instead of ! you can use L or l for ‘long’ or just \long itself.
- In the arguments specification string you can write >[{}]<prefix>[{} ] to make subsequent argument ‘long’ or ignored:
- Any of Pp!1L\long\par to make a particular argument ‘long’, allowing \par in it that is, and/or any of iI to make the argument ignored (just gobbled).
- (Note that also the c and C arguments may be made ‘long’. That’s because I use them not as coordinates but as just another kind of optional argument.)

Pp!1L\long\par  
ii

The concept of ignored arguments came to my head when I was declaring a command with three braced optionals and put optional stars only to distinguish the braced optionals.

For example, after

```
\DeclareCommand\GLBTQKi {%
 G{&&} {\#1 default}
 >LB{\#2 default}
 >iT{*ht}
 Q{0123456789}{o}
 K{\#1\par\#2\par}{{\text{\#1}}\over\text{\#2}}}}
```

and you get \GLBTQKi 4-argument with:

#1 G{&&} optional short in a pair of & with \NoValue as the default,  
 {#2} >LB optional long in curly braces,  
*(ign.)* >{i}iT{\*ht} star or \ht control sequence,  
 #3 Q{0123456789}{o} optional sequence of decimal digits with default o,  
 #4 K{\#1\par\#2\par}{{\text{\#1}}\over\text{\#2}}} mandatory sequence of  
 two arguments both delimited with \par, that will be passed the inner macro as  
 {\text{\#1}}\over\text{\#2}}.

\IfLong

The arguments may be tested inside the command with \IfLong {#*n*} {*what if long*} {*what if short*} . The test looks for \par at any level of nesting ({\par}—\par in braces will not hide) since it \detokenizes its argument.

\global \outer

If you wish to define your command \globally, you can specify \DeclareCommand% \mycommand\global. If you wish to forbid usage of your command in arguments of macros, add the \outer prefix. As with original T<sub>E</sub>X's \def and like, the prefixes are allowed in any order and in any number only here they come between the command's name and the arguments specification. You can also add \long, as we mentioned above, and, for the symmetry, also \protected, although the latter is *always* added since the command is not expandable.

iIwW\sphack

Handling of white spaces with optionals seems to me too complicated compared to the estimated weight of the problem and I haven't faced it so far so I don't provide anything. But!—but there are some commands that should be invisible in the typeset text, such as indexing commands and font declarations. For those there is a working L<sup>A</sup>T<sub>E</sub>X mechanism of \@bsphack—\@esphack and to use it I provide yet another 'prefix': if you type W or w (for 'white') or I or i (for 'invisible') or, if you prefer a prefix-like, \sphack<sup>2</sup> between the CS to be defined and arguments spec, \@bsphack and \@esphack will be added in proper places.

\IfDCMessages  
 \DCMessagesfalse  
 \DCMessagestrue  
 .qQ

The original inner macros of the ancient xparse had names like \@dc@o etc. According to my T<sub>E</sub>X Guru's advice I changed them to \ArgumentCatcher@*letter(s)* to make the error messages less confusing. Well, I don't know if they are but \ArgumentCatcher@PO looks better than \@dc@PO doesn't it?

Talking of messages, there's a Boolean switch \IfDCMessages . Its default setting is \DCMessagesfalse but if you set \DCMessagestrue, every \command created with my \DeclareCommand will issue a message "Parsing arguments for \command" at the beginning of its execution.

If you are positive that no such message will ever be useful, you can suppress the very placing of it in the command's definition with an optional argument to \Decla! recommand with all other 'prefixes', . or q or Q for 'quiet'.

---

<sup>2</sup> I don't define the \sphack control sequence and don't assume it's defined. I use it only as a marker and my use of it doesn't create an entry in the hash table.

To sum up, `\DeclareCommand` takes the following arguments:

- #1 >{P}m the command to be defined (can be even `\par` if you really wish),
- #2 Q{\long\global\outer\protected!1L.qIwW\sphack} optional e-TeX's prefix(es) and/or symbols for making all the arguments long (!, 1 or L) and/or to suppress placing of the diagnostic message in the definition (., q, Q) and/or for placing \@sphack—\@esphack (i, I, w, W, \sphack),
- #3 >{P}m the arguments specification (can contain `\par` as you see),
- #4 >{P}m the definition body. You refer to the arguments with #<n> and can test their presence and absence with `\If[No]Value(T|F|TF){#<n>}` and if the argument was specified with the >P prefix (allows `\par` in itself), you can test it with `\IfLong{#<n>}{{if long}}{if short}}`. You are also provided the `\IfAmong...%` `\among` and `\IfIntersect` tests defined earlier in this package to process the arguments, especially of the S/T and Q type.

There is also the `\DeclareEnvironment` command to define environments with sophisticated optionals. It takes the arguments analogous to those of `\DeclareCommand`.

The `iIwW\sphack` specifier however acts different: it doesn't add `\@bsphack` nor `\@esphack` but only `\@ignoretrue` to the end macro so the spaces following `\end{myenvir}` will be ignored. (I tried the space hack but it's problematic ("bad space factor" error) if an environment begins in the vertical mode and ends in horizontal.

So the arguments to `\DeclareEnvironment` are:

- #1 >{P}m the environment's name; you may wonder why it allows `\par`; it's to allow environments like `\string\par`—I met such an environment once.
- #2 Q{\long\global\protected!1L.qIwW\sphack} to prefix the command (note that `\outer` prefix will actually not work since the command is called with `\csname`), make all the arguments 'long' (`\long`, !, 1 or L), not to place the message issuer in the command (., q, Q) or to make the environment ignores spaces following its end (`iIwW\sphack`).
- #3 >{P}m the arguments specification (both for the begin and for the end macros)
- #4 >{P}m the begin definition,
- #5 >{P}m the end definition; it can use the same parameters (#<n>'s) as the begin definition. Note however that there is only one specification of the arguments and both begin and end have to have 9 parameters in total at most.

```
1582 \unless\ifdefined\@temptokenb
\@temptokenb 1583 \newtoks\@temptokenb
1584 \fi
\if@dc@alllong@ 1586 \newif\if@dc@alllong@% for an option of all arguments long.
\if@dc@quiet@ 1587 \newif\if@dc@quiet@% for suppressing the message of using of declared command.
\IfDCMessages 1590 \newif\IfDCMessages% a global switch to suppress the message about parsing.
\@dc@long@yes 1594 \def\@dc@long@yes {P}
\@dc@arguments 1596 \newtoks\@dc@arguments
\@dc@catchernum 1598 \newcount\@dc@catchernum
\@dc@argnum 1599 \newcount\@dc@argnum
\if@dc@ignore 1601 \newif\if@dc@ignore
1605 \long\pdef\DeclareCommand#1#2#3{%
 % #1 command to be defined,
 % #2 arguments specification,
```

```

% #3 definition body.
\@dc@catchernum 1616 \@dc@catchernum\z@
1617 \@dc@argnum\z@
1618 \toks@{}% in this register we will store the created sequence of argument catch-
ers.
1622 \@temptokena\toks@% in this register we will store the sequence of #1#2...
1623 \@temptokenb\toks@% in this register we will store the sequences of m's.
1627 \emptyify@\@dc@long@
1628 \@dc@ignorefalse
1629 \@dc#2X% X is the sentinel of parsing.
1630 \protected\edef#1{%
 \nx@\@dc@{\the\toks@}%
 \xa@\nx\csname\string#1\endcsname
 \nx@#1%
}%
1635 \edef\gmu@reserveda{%
 \long\def\@xa\@nx\csname\string#1\endcsname
 \the\@temptokena{%
 \unexpanded{#3}}}}%
1639 \gmu@reserveda}% of the 'draft' \DeclareCommand.

\@dc@ 1643 \long\def\@dc@{%
 #1% the argument catchers in a pair of braces (their arguments may contain \par
 so the macro is long).
 #2% \thecommand
 #3% \thecommand
 {%
 \ifx\protect\@typeset@protect
 \xa@\firstofone
 \else
 \protect#3\@xa\@gobble
 \fi
 {\@dc@arguments{#2}#1\the\@dc@arguments}}
 }%
1654 \def\@dc#1{%
 \ifx#1X% The main parsing macro.
 1655 \ifx#1\@temptokenb% if we meet the sentinel we stop. Note there may be some m's in
 % \@temptokenb and we might not care about them since they denote
 mandatory arguments and as such they will be grabbed by
 % \\\<declared command>. However, if any of them contains \par (while it
 shouldn't), then it wouldn't be noticed since \\<declared command>
 is always long. Therefore:
 1656 \@dc@add@ms
 1657 \else
 1658 \ifx#1>%
 1659 \let\next@dc\grab@prefix
 1660 \else% not X neither >
 1661 \if@dc@alllong@\let\@dc@long@\@dc@long@yes\fi
 1662 \ifx#1\afterassignment\@dc@ignoretrue\fi% if we parse the
 assigned pseudo-argument, we don't want to add anything to the ar-
 guments' toks register so we apply general mechanism of ignoring an
 argument.
 1663 \ifx#1\gobblespace\@dc@ignoretrue\fi% again, as above for ig-
 noring spaces.
 1664 \fi
 1665 }
 1666 }
```

```

1678 \ifnum\ifx#1mo\else1\fi\ifx\@dc@long@\empty_o\else
1679 2\fi
1680 \if@dc@ignore_1\elseo\fi=_o% we check whether #1 is
1681 % m and not prefixed.
1682 \addto@hook\@temptokenb_m% as you see, \@temptokenb serves
1683 as a storage of m's.
1684 \else% (not m) or prefixed
1685 \@dc@add@ms
1686 \if@dc@ignore\addtotoks\toks@{\@dc@ignorearg}\fi
1687 \@xa\addtotoks\@xa\toks@\@xa{%
1688 \csname\ArgumentCatcher@\@dc@long@\detokenize{\#1}\endcsname}
1689 argument catcher's name is \ArgumentCatcher@[P]> <arg.%
1690 % type>.
1691 \@temptokenb{}%
1692 \fi% of \if short m or not
1693 \advance\@dc@catchernum\@ne
1694 \unless\if@dc@ignore
1695 \advance\@dc@argnum\@ne
1696 \@temptokena\@xa{%
1697 \the\@xa\@temptokena\@xa##\the\@dc@argnum}%
1698 \else
1699 \@dc@ignorefalse
1700 \fi
1701 \IfAmong#1\among{ABCO}%
1702 {\let\next@dc\grab@default}%
1703 {\IfAmong#1\among{GQST}%
1704 {\let\next@dc\grab@defaults}%
1705 G, Q, S and T have second argu-
1706 ment, braced optional, which is the default value (\NoValue by
1707 default).
1708 {\gmu@if_x_{K#1}%
1709 {\let\next@dc\@dc@define@K}%
1710 {\gmu@if_x_{\afterassignment#1}%
1711 {\let\next@dc\dc\grab@afterass}%
1712 {\let\next@dc\@dc}%
1713 }%
1714 }%
1715 \unless\ifx\next@dc\@dc@define@K
1716 \emptyify\@dc@long@
1717 \fi
1718 \fi% of \if#1>.
1719 \@xa\next@dc
1720 \fi
1721 }%
1722 \def\@dc@add@ms{%
1723 \@xa\@ifempty\@xa{\the\@temptokenb}%
1724 {}%
1725 {\toks@\@xa{%
1726 \the\@xa\toks@
1727 \csname\ArgumentCatcher@\the\@temptokenb\endcsname}}}
1728 \long\def\grab@default#1{%
1729 the default value or the list of tokens for S may
1730 contain \par (why not?), that's why we make this macro \long.

```

```

1731 \toks@{\xa{\the\toks@{\#1}}%
1732 \let\next@dc\@dc
1733 \@dc
1734 }

\grab@defaults 1736 \long\def\grab@defaults#1{%
1737 \toks@{\xa{\the\toks@{\#1}}%
1738 \@ifnextchar\bgroup
1739 {\grab@default}%
1740 {\grab@default{\NoValue}}%
1741 }

\dc@grab@afterass 1743 \def\dc@grab@afterass{%
1744 \@ifnextchar\bgroup
1745 {\grab@default}%
1746 {\grab@default{\@tempcpta}}%
1747 }

\@dc@addargum 1750 \long\def\@dc@addargum{%
1751 \addtotoks\@dc@arguments}

1753 \StoreMacro\@dc@addargum

\@dc@ignorearg 1755 \pdef\@dc@ignorearg{%
\@dc@addargum 1756 \long\def\@dc@addargum##1{%
1757 \RestoreMacro\@dc@addargum}}

```

Parsing of Knuthian parameters string is easier to implement with full-feature \DeclareCommand so we postpone it till line [2363](#)

```

\grab@prefix 1763 \def\grab@prefix#1{%
1764 \IfIntersect{\#1}{\long!1L\par\lPp}%
\@dc@long@ 1765 {\def\@dc@long@{P}}{}%
1766 \IfIntersect{\#1}{Ii}%
1767 {\@dc@ignoretrue}{}%
1768 \@dc
1769 }

\ArgumentCatcher@o 1771 \long\def\ArgumentCatcher@o#1\@dc@arguments{%
with no default
1772 \@ifnextchar[%
\@dc@parse@next 1773 {\def\@dc@parse@next{\#1}\ArgumentCatcher@o@}%
the last macro has
to be short so we can't give it all the arguments' catchers since some de-
fault may contain \par. Therefore we hide all the argument catchers in
% \@dc@parse@next.
1778 {\@dc@addargum\NoValue#1\@dc@arguments}}}

\ArgumentCatcher@o@ 1780 \def\ArgumentCatcher@o@[#1]{%
1781 \@dc@addargum{\#1}\@dc@parse@next\@dc@arguments}

\ArgumentCatcher@Po 1783 \long\def\ArgumentCatcher@Po#1\@dc@arguments{%
parsing of an optional
with no default.
1784 \@ifnextchar[%
1785 {\ArgumentCatcher@Po@{\#1}}%
1786 {\@dc@addargum\NoValue#1\@dc@arguments}}}

\ArgumentCatcher@Po@ 1788 \long\def\ArgumentCatcher@Po@[#2]{%
1789 \@dc@addargum{\#2}\#1\@dc@arguments}

```

```

\ArgumentCatcher@O 1791 \long\def\ArgumentCatcher@O#1#2\@dc@arguments {%
 parsing of optional
 with default value.
 % #1 the default value,
 % #2 the tail of args parser.
1797 \@ifnextchar[%
\@dc@parse@next 1798 {\def\@dc@parse@next {\#2}\ArgumentCatcher@O}%
 note that even when
 an optional is short, its default may contain \par.
1800 {\@dc@addargum{\{#1\}}\#2\@dc@arguments} }

\ArgumentCatcher@PO 1803 \long\def\ArgumentCatcher@PO#1#2\@dc@arguments {%
 parsing of optional
 with default value.
 % #1 the default value,
 % #2 the tail of args parser.
1809 \@ifnextchar[%
1810 {\ArgumentCatcher@Po\#2}%
 note that even when an optional is short, its de-
 fault may contain \par.
1812 {\@dc@addargum{\{#1\}}\#2\@dc@arguments} }

\ArgumentCatcher@b 1815 \long\def\ArgumentCatcher@b#1\@dc@arguments {%
 parsing of an optional
 braced short argument.
1816 \@ifnextcat\bgroun
\@dc@parse@next 1817 {\def\@dc@parse@next {\#1}\ArgumentCatcher@m}%
 the last macro has
 to be short so we can't give it all the arguments' parsers since some de-
 fault may contain \par. Therefore we hide all the argument catchers in
 \% \@dc@parse@next.
1821 {\@dc@addargum\NoValue\#1\@dc@arguments} }

\ArgumentCatcher@Pb 1824 \long\def\ArgumentCatcher@Pb#1\@dc@arguments {%
 parsing of a braced
 and long optional with \NoValue default.
1826 \@ifnextcat\bgroun
1827 {\ArgumentCatcher@Pm\#1\@dc@arguments}%
1828 {\@dc@addargum\NoValue\#1\@dc@arguments}%
1829 }

\ArgumentCatcher@B 1831 \long\def\ArgumentCatcher@B#1#2\@dc@arguments {%
 parsing of a braced
 optional with default value.
 % #1 the default value,
 % #2 the tail of args catchers.
1837 \@ifnextcat\bgroun
\@dc@parse@next 1838 {\def\@dc@parse@next {\#2}\ArgumentCatcher@m}%
 note that even when
 an optional is short, its default may contain \par.
1840 {\@dc@addargum{\{#1\}}\#2\@dc@arguments} }

\ArgumentCatcher@PB 1842 \long\def\ArgumentCatcher@PB#1#2\@dc@arguments {%
 parsing of a braced
 and long optional with default value.
 #1 the default value,
 #2 the tail of args catchers.
1849 \@ifnextcat\bgroun
1850 {\ArgumentCatcher@Pm\#2\@dc@arguments}%
1851 {\@dc@addargum{\{#1\}}\#2\@dc@arguments} }

\ArgumentCatcher@S@ 1854 \long\def\ArgumentCatcher@S@% parsing of a Single continued:
1855 #1% the list to search #4 on,
1856 #2% the default value,
1857 #3% the tail of args parsers,

```

```

1858 #4% the token we search in #1 (it's an unbraced single token as we checked in line
1859 1873).
1860 {%
1861 \IfAmong#4 \among{#1}%
1862 { \@dc@addargum{{#4}}#3 \@dc@arguments }%
1863 { \@dc@addargum{{#2}}#3 \@dc@arguments#4 }%
1864 }

\ArgumentCatcher@S 1866 \long\def\ArgumentCatcher@S{%
 parsing of a Single token. It's always long
 since we look for a single unbraced token so there is no danger of "runaway
 argument".
1869 #1% the list we search optional token on,
1870 #2% the default value,
1871 #3% the tail of args parser.
1872 \@dc@arguments{%
1873 \@ifnextsingle
1875 {\ArgumentCatcher@S{#1}{#2}{#3}}%
1876 { \@dc@addargum{{#2}}#3 \@dc@arguments }%
 if we parse an opening or
 closing brace, then we are sure it's not any expected Single.
1879 }

1882 \let\ArgumentCatcher@PS\ArgumentCatcher@S{%
 as above: we always act
 'long' with single tokens.
1883 \let\ArgumentCatcher@T\ArgumentCatcher@S{%
 we allow T (for Token) as
 an alias of S.

1887 \edef\ArgumentCatcher@s{%
1888 \@nx\ArgumentCatcher@S{\@xa\@nx\all@stars}{\@nx\NoValue}}%
 the
 s arg spec is a shorthand for S{*}. Except the star may be of any category
 from {11, 12, 13}.
1891 \let\ArgumentCatcher@Ps\ArgumentCatcher@s

\ArgumentCatcher@Q 1893 \long\def\ArgumentCatcher@Q{%
 parsing of a seQuence of tokens from the list.
 It's always long since we look for unbraced tokens so there is no danger of
 "runaway argument".
1896 #1% the list we search optional tokens on,
1897 #2% the default value,
1898 #3% the tail of args parser.
1899 \@dc@arguments{%
1900 {%
 \def\@dc@parse@next{#3}%
 \emptyify\@dc@seQuence
 \ArgumentCatcher@Q{#1}{#2}%
}

\ArgumentCatcher@PQ 1906 \let\ArgumentCatcher@PQ\ArgumentCatcher@Q

\ArgumentCatcher@Q@ 1908 \long\def\ArgumentCatcher@Q@#1#2{%
 list, default
1910 \@ifnextsingle
1911 {\ArgumentCatcher@Q@@{#1}{#2}}%
1912 { \@dc@seQuence@finish{#2} }%
1913 }

\ArgumentCatcher@Q@@ 1915 \long\def\ArgumentCatcher@Q@@#1#2#3{%
 list, default, token to be checked
 we only launch this macro when not before opening curly brace. #1 is the list
 of acceptable values, #2 the default value, #3 is the token to be sought on the
 list.

```

```

1920 \IfAmong „#3\among{\#1}%
1921 {\addtomacro{@dc@seQuence#3%
1922 \ArgumentCatcher@Q@{\#1}{\#2}%
1923 {@dc@seQuence@finish{\#2}#3}%
1924 }

{@dc@seQuence@finish
1926 \long\def{@dc@seQuence@finish{\#1}%
1928 \ifx{@dc@seQuence\empty\def{@dc@seQuence{\#1}\fi
1929 {@xa{@dc@addargum{@xa{\@xa{@dc@seQuence}}%
1931 {@dc@parse@next{@dc@arguments„}

\ArgumentCatcher@c
1934 \long\def\ArgumentCatcher@c{\#1{@dc@arguments{%
1935 \ifnextchar{%
1936 {\def{@dc@parse@next{\#1}\ArgumentCatcher@c@}%
1937 {%
1938 {@dc@addargum{\{NoValue\}}{\#1{@dc@arguments}%
1941 }

\DCcoordinate
1943 \def\DCcoordinate{%
1944 \def\ArgumentCatcher@c{\#1}{%
1945 \IfAmong, \among{\#1}%
1946 {@xa{@dc@addargum{@dc@commasep{\#1{@dc@commasep}%
1947 {@dc@addargum{\{\#1\}}{@dc@parse@next{@dc@arguments}%

\@dc@commasep
1949 \def{@dc@commasep{\#1,\#2{@dc@commasep{\{\{\#1\{\#2\}}\}}}

\ArgumentCatcher@Pc@
1951 \long\def\ArgumentCatcher@Pc@{\#1}{%
1952 \IfAmong, \among{\#1}%
1953 {@xa{@dc@addargum{@dc@commasep{\#1{@dc@commasep}%
1954 {@dc@addargum{\{\#1\}}{@dc@parse@next{@dc@arguments}%
1955 }

\DCnocoordinate
1957 \def\DCnocoordinate{%
1958 \def\ArgumentCatcher@c{\#1}{%
1959 {@dc@addargum{\{\#1\}}{@dc@parse@next{@dc@arguments}%

\ArgumentCatcher@Pc@
1961 \long\def\ArgumentCatcher@Pc@{\#1}{%
1962 {@dc@addargum{\{\#1\}}{@dc@parse@next{@dc@arguments}%
1963 }

1965 \DCnocoordinate

\ArgumentCatcher@c
1967 \long\def\ArgumentCatcher@c{\#1\#2{@dc@arguments{%
1968 \ifnextchar{%
1969 {\def{@dc@parse@next{\#2}\ArgumentCatcher@c@}%
1970 {@dc@addargum{\{\#1\}}{\#2{@dc@arguments}}}

\ArgumentCatcher@Pc
1973 \long\def\ArgumentCatcher@Pc{\#1{@dc@arguments{%
1974 \ifnextchar{%
1975 {\def{@dc@parse@next{\#1}\ArgumentCatcher@Pc@}%
1976 {@dc@addargum{\{NoValue\}}{\#1{@dc@arguments}}%
1977 }

\ArgumentCatcher@PC
1979 \long\def\ArgumentCatcher@PC{\#1\#2{@dc@arguments{%
1980 \ifnextchar{%
1981 {\def{@dc@parse@next{\#2}\ArgumentCatcher@Pc@}%
1982 {@dc@addargum{\{\#1\}}{\#2{@dc@arguments}}}

\ArgumentCatcher@Pm
1985 \long\def\ArgumentCatcher@Pm{\#1{@dc@arguments\#2{%

```

```

1986 \@dc@addargum { {#2} }#1 \@dc@arguments}
\gmu@hashes 1988 \def \gmu@hashes#1#2 {%
 this is a fully expandable loop analogous to that of The
 ε-TeX Manual p. 9.
 1990 \ifnum#1<#2%
 1991 #####\number#1
 1992 \expandafter\gmu@hashes
 1993 \expandafter{\number\numexpr#1+1\expandafter}%
 1994 \expandafter{\number#2\expandafter}%
 1995 \fi}%
 of \gmu@hashes.

\gmu@hashesbraced 1997 \def \gmu@hashesbraced#1#2 {%
 1998 \ifnum#1<#2%
 1999 #####\number#1}%
 2000 \expandafter\gmu@hashesbraced
 2001 \expandafter{\number\numexpr#1+1\expandafter}%
 2002 \expandafter{\number#2\expandafter}%
 2003 \fi}%
 of \gmu@hashesbraced.

2005 \tempcpta=1
2006 \whilenum \tempcpta<9 \do {%
 2007 \edef \gmu@ms {\romannumerals \numexpr \tempcpta*1000 }%
 2011 \edef \gmu@tempa {%
 2012 \long \def \xa @nx \csname
 ArgumentCatcher@ \gmu@ms
 2013 \endcsname
 2014 ##1 @nx \@dc@arguments { \def \nx \@dc@parse@next {##1}%
 2015 \xa @nx \csname
 ArgumentCatcher@ \gmu@ms @%
 2016 \endcsname }%
 2017 \def \xa @nx \csname
 ArgumentCatcher@ \gmu@ms @%
 2018 \endcsname
 2020 \def \xa @nx \csname
 ArgumentCatcher@ \gmu@ms @%
 2022 \endcsname
 2023 \gmu@hashes1 { \numexpr \tempcpta+1 }%
 2024 { \nx \@dc@addargum {%
 2025 \gmu@hashesbraced1 { \numexpr \tempcpta+1 } }%
 2026 \nx \@dc@parse@next \nx \@dc@arguments
 2027 }% of \ArgumentCatcher@ <m's>@.
 2028 }% of \edef.
 2029 \gmu@tempa
 2030 \advance \tempcpta1 \relax
 2031 }% of the loop.

```

The loop above defines 8 pairs of macros as we see thanks to the code below:

```

2035 \if 1 1
2036 \tempcpta \ne
2037 \whilenum \tempcpta<9 \do {%
 2038 \edef \gmu@ms {\romannumerals \numexpr \tempcpta*1000 }%
 2039 \typeout { \bslash ArgumentCatcher@ \gmu@ms :%
 2040 ^J \xa \meaning \csname ArgumentCatcher@ \gmu@ms %
 \endcsname }%
 2041 \typeout { \bslash ArgumentCatcher@ \gmu@ms @ :%
 2042 ^J \xa \meaning \csname ArgumentCatcher@ \gmu@ms @ %
 \endcsname }%

```

```

2043 \advance\@tempcnta\@ne
2044 }
2045 \fi

```

The code above produces on the terminal:

```

\ArgumentCatcher@m:
macro:#1\@dc@arguments ->\def \@dc@parse@next
 {#1} \ArgumentCatcher@m@
\ArgumentCatcher@m@:
macro:#1->\addto@hook \@dc@arguments {{#1}}\@dc@parse@next
 \@dc@arguments
\ArgumentCatcher@mm:
macro:#1\@dc@arguments ->\def \@dc@parse@next
 {#1} \ArgumentCatcher@mm@
\ArgumentCatcher@mm@:
macro:#1#2->\addto@hook \@dc@arguments
 {{#1}{#2}}\@dc@parse@next \@dc@arguments
\ArgumentCatcher@mmm:
macro:#1\@dc@arguments ->\def \@dc@parse@next
 {#1} \ArgumentCatcher@mmm@
\ArgumentCatcher@mmm@:
macro:#1#2#3->\addto@hook \@dc@arguments
 {{#1}{#2}{#3}}\@dc@parse@next \@dc@arguments
\ArgumentCatcher@mmmm:
macro:#1\@dc@arguments ->\def \@dc@parse@next
 {#1} \ArgumentCatcher@mmmm@
\ArgumentCatcher@mmmm@:
macro:#1#2#3#4->\addto@hook \@dc@arguments
 {{#1}{#2}{#3}{#4}}\@dc@parse@next \@dc@arguments
\ArgumentCatcher@mmmm:
macro:#1\@dc@arguments ->\def \@dc@parse@next
 {#1} \ArgumentCatcher@mmmm@
\ArgumentCatcher@mmmm@:
macro:#1#2#3#4#5->\addto@hook \@dc@arguments
 {{#1}{#2}{#3}{#4}{#5}}\@dc@parse@next \to
ks@
\ArgumentCatcher@mmmmmm:
macro:#1\@dc@arguments ->\def \@dc@parse@next
 {#1} \ArgumentCatcher@mmmmmm@
\ArgumentCatcher@mmmmmm@:
macro:#1#2#3#4#5#6->\addto@hook \@dc@arguments
 {{#1}{#2}{#3}{#4}{#5}{#6}}\@dc@parse@ne
xt \@dc@arguments
\ArgumentCatcher@mmmmmmmm:
macro:#1\@dc@arguments ->\def \@dc@parse@next
 {#1} \ArgumentCatcher@mmmmmmmm@
\ArgumentCatcher@mmmmmmmm@:
macro:#1#2#3#4#5#6#7->\addto@hook \@dc@arguments
 {{#1}{#2}{#3}{#4}{#5}{#6}{#7}}\@dc@pa
rse@next \@dc@arguments
\ArgumentCatcher@mmmmmmmmmm:
macro:#1\@dc@arguments ->\def \@dc@parse@next
 {#1} \ArgumentCatcher@mmmmmmmmmm@

```

```

\ArgumentCatcher@mmmmmmmm@:
macro:#1#2#3#4#5#6#7#8->\addto@hook \@dc@arguments
 {#1}{#2}{#3}{#4}{#5}{#6}{#7}{#8})\@
dc@parse@next \@dc@arguments

umentCatcher@mmmmmmmm 2087 \def \ArgumentCatcher@mmmmmmmm{\the \@dc@arguments
2088 #1#2#3#4#5#6#7#8#9% yes, it has to be delimited (well, actually: started) by
 % \the \@dc@arguments not only \@dc@arguments. And it doesn't need
 to be long and launch inner short macro because there's nothing more to catch.
2092 \@dc@addargum{%
2093 {#1}{#2}{#3}{#4}{#5}{#6}{#7}{#8}{#9})\the \@dc@arguments}
2096 \long \def \ArgumentCatcher@Pm#1 \@dc@arguments#2% catcher of a long
 mandatory.
2097 \@dc@addargum{#2}#1 \@dc@arguments}

\ArgumentCatcher@K 2100 \long \def \ArgumentCatcher@K% a Knuthian delimited or undelimited argu-
 ments parsed and passed to the command's body in a digest (Knuthian re-
 placement)
2103 #1% the particular Knuthian macro
2104 #2% the tail of args parser.
2105 \@dc@arguments% tail delimiter
\@dc@parse@next 2106 {\def \@dc@parse@next {#2}#1}

2108 \let \ArgumentCatcher@PK \ArgumentCatcher@K

\gmu@twostring 2110 \long \def \gmu@twostring#1#2{\string#1\string#2}

\ArgumentCatcher@G 2112 \long \def \ArgumentCatcher@G% general catcher
2113 #1% left and right delimiter
2114 #2% default value
2115 #3% the tail of args parser.
2116 \@dc@arguments% tail delimiter
2117 {%
2118 \edef \@dc@Gname {ArgumentCatcher@G\gmu@twostring#1}%
2119 \unless \ifcsname \@dc@Gname \endcsname
2120 \xa \def \csname \@dc@Gname \xa \endcsname
2121 \@dc@Gparams#1 {\@dc@addargum{##2}##1 \@dc@arguments}%
2122 \fi
2123 \xa \ifnextchar \@firstoftwo#1%
2124 {\def \dc@parse@next {#3}% we hide possible \pars.
2125 \csname \@dc@Gname \endcsname \dc@parse@next }%
2126 {\@dc@addargum{#2}#3 \@dc@arguments}%
2127 }

\dc@parse@next 2130 \long \def \ArgumentCatcher@PG% general catcher
2131 #1% left and right delimiter
2132 #2% default value
2133 #3% the tail of args parser.
2134 \@dc@arguments% tail delimiter
2135 {%
2136 \edef \@dc@Gname {ArgumentCatcher@PG\gmu@twostring#1}%
2137 \unless \ifcsname \@dc@Gname \endcsname
2138 \long \xa \def \csname \@dc@Gname \xa \endcsname
2139 \@dc@Gparams#1 {\@dc@addargum{##2}##1 \@dc@arguments}%
2140 \fi

```

```

2141 \@xa\@ifnextchar\@firstoftwo#1%
2142 {\csname\@dc@Gname\endcsname{\#3}}%
2143 {\@dc@addargum{\#2}\#3\@dc@arguments}%
2144 }

\@dc@Gparams 2147 \long\def\@dc@Gparams#1#2{##1##2#2}%
% #1 delimited with first argument and #2 delimited with second. In fact it's
intended to pass #1 further (it will always be braced) and to catch an argument
put in a pair of tokens given \@dc@Gparams as the arguments.

\ArgumentCatcher@A 2153 \long\def\ArgumentCatcher@A{\ArgumentCatcher@G{<>}}
\ArgumentCatcher@PA 2154 \long\def\ArgumentCatcher@PA{\ArgumentCatcher@PG{<>}}
\ArgumentCatcher@a 2156 \long\def\ArgumentCatcher@a{\ArgumentCatcher@G{<>}{\NoValue}}
\ArgumentCatcher@Pa 2157 \long\def\ArgumentCatcher@Pa{\ArgumentCatcher@PG{<>}{%
\NoValue} }

\NoValue 2160 \def\NoValue{-NoValue-}

\IfNoValueTF 2162 \long\def\IfNoValueTF#1{%
 \@xa\ifx\@firstofmany#1\@empty\@nil\NoValue\@afterfi%
 \@firstoftwo
 \else\afterfi\@secondoftwo
 \fi}

\IfNoValueT 2172 \long\def\IfNoValueT#1#2{\IfNoValueTF{#1}{#2}\@empty}
\IfNoValueF 2174 \long\def\IfNoValueF#1#2{\IfNoValueTF{#1}\@empty{#2}}
\IfValueTF 2176 \long\def\IfValueTF#1#2#3{\IfNoValueTF{#1}{#3}{#2}}
\PutIfValue 2178 \long\def\PutIfValue#1{\IfValueT{#1}{#1}}
\IfValueT 2181 \let\IfValueT\IfNoValueF
\IfValueF 2184 \let\IfValueF\IfNoValueT
\IfLong 2186 \long\pdef\IfLong#1{%
 % #1 the argument to check for \par,
 % #2 what if \par found,
 % #3 what if \par not found.
 \edef\gmu@reserveda{\detokenize{#1}}%
 \edef\gmu@reserveda{%
 \IfAmong\string\par\@nx\among{\gmu@reserveda}}%
 \gmu@reserveda}

\afterassignment pseudo-argument
2201 \long\@namedef{ArgumentCatcher@}\detokenize{%
 \afterassignment}%
2202 #1% register used in the assignment
2203 #2% the tail of args parser.
2204 \@dc@arguments{%
\@dc@parse@next 2205 \def\@dc@parse@next{%
 \@dc@addargum\NoValue\@% to make ignoring mechanism work
 #2\@dc@arguments}%
 \afterassignment\@dc@parse@next
 #1}% optional space and = is left at user's discretion. In fact, #1 may be empty
 and then each time our command is used the left side of the assignment has
 to be given explicitly and may even be different each time.

```

```

2214 \n@melet {ArgumentCatcher@P\detokenize{\afterassignment}}}
2215 {ArgumentCatcher@\detokenize{\afterassignment}}
 \gobblespace pseudo-argument: useful between optional and Knuthian-type arguments and after all parsing. Equivalent >iT {\somedummymacro}. At first I wanted mark it \ignorespaces but the gobbling doesn't expand macros: it doesn't use \ig| norespaces but \ifnextchar's space trick.

2223 \long\namedef{ArgumentCatcher@\detokenize{\gobblespace}}%
2224 {\ArgumentCatcher@S{\dc@gobblespace@dum} {\NoValue} }

2226 \n@melet {ArgumentCatcher@P\detokenize{\gobblespace}}}
2227 {ArgumentCatcher@\detokenize{\gobblespace} }

\dc@gobblespace@dum 2229 \def\dc@gobblespace@dum {\dc@gobblespace@dum}
2231 \let\gobblespace\dc@gobblespace@dum
 end of catchers' definitions

\DeclareCommand 2237 \DeclareCommand\DeclareCommand_{\% This is the final version by now. Note
 we introduce the 'prefixes' at the #2 position only at this point so we have yet
 to prefix every argument specifier separately. Note also we only here 'teach'
 \DeclareCommand to pass the name of its main argument in the \dc@bsname
 macro.
2242 {>Pm_{\% (1) command to be defined (may be \par if you really
 wish),
2244 Q{\global\protected\outer\long!1L.qQiIwW\sphack\envhack}\% (2)
 an optional seQuence of \eTeX's prefixes and/or ! or 1 or L for 'all long'
 and/or . or q or Q for 'quiet' (message about calling the command is
 suppressed) and/or i or I as 'invisible' or W or w as 'white' or \sphack
 to make the command 'invisible' in the leading with the \@bsphack...
 \@esphack3,
2257 >Pm_{\% (3) arguments specification (may contain \par),
2258 >Pm_{\% (4) the definition body; may contain nearly anything including \par and
 tests for presence/absence of arguments
 \If[No]Value(T|F|TF) and for their longness \IfLong; you refer to the
 arguments with #<n>.
2262 }{\%
2270 \edef\dc@bsname{\string#1}\% we'll use it in K-type arg. catcher and to
 allow #1 == \par (\PackageWarning is short and \typeout too!).
2273 \gm@testdefined#1\iftrue
2274 \IfDCMessages
2275 \PackageWarning{gmutils}{(gmparse)}{Redefining command
 \dc@bsname\space
 with \string\DeclareCommand}\%
2276 \fi
2277 \else
2278 \IfDCMessages
2279 \typeout{^^J\string\DeclareCommand\dc@bsname\space
 (new) ^^J}\%
2280 \fi
2281 \fi
2282 \fi

```

<sup>3</sup> To deal with an 'invisible' environment, there is one more acceptable value of this argument: \en| vhack, which suppresses placing \@bsphack—\@esphack and places \@ignoretrue instead in the end macro.

First we parse the declaration options to know whether all the arguments are to be long and whether we should suppress the message "Parsing arguments for...".

```

2287 \@dc@alllong@false
2288 \@dc@quiet@false
2290 \IfIntersect {\#2} {\long!1L} {%
2292 \@dc@alllong@true} {}%
2293 \IfIntersect {\#2} {.\qQ} {\@dc@quiet@true} {}%
2294 \IfIntersect {\#2} {iIwW\sphack} %
2295 {%
2296 \def \@dc@bsphack@ {\@nx \@bsphack} %
2297 \def \@dc@esphack@ {\@nx \@esphack} %
2298 }%
2299 {\emptyify \@dc@bsphack@ \emptyify \@dc@esphack@} % if 'invisibility' is not
2300 specified.
2301 \IfAmong \envhack \among {\#2} %
2302 {%
2303 but if we define an environment:
2304 \emptyify \@dc@bsphack@ \emptyify \@dc@esphack@ %
2305 {%
2306 and if we don't define an environment:
2307 \emptyify \@dc@env@argstoend} %
2308 \relaxen \@dc@global@ \relaxen \@dc@outer@
2309 \IfAmong \outer \among {\#2} {\let \@dc@outer@ \outer} {}%
2310 \IfAmong \global \among {\#2} {\let \@dc@global@ \global} {}%

```

We initialise the scratch count and toks registers before parsing of the arguments specifiers.

```

2310 \@dc@catchernum\z@
2311 \@dc@argnum\z@
2312 \toks@ {}%
2313 \@temptokena\toks@
2314 \@temptokenb\toks@

```

We parse the arguments specifiers:

```
2316 \@dc#3X% X is the sentinel of parsing.
```

Now we define the basic macro:

```

2318 \@dc@outer@ \@dc@global@ % possibly the prefixes,
2319 \protected\edef#1 {%
2320 \@dc@bsphack@ % perhaps \@bsphack
2321 \unless\if@dc@quiet@
2322 \unexpanded{%
2323 \IfDCMessages
2324 \typeout{Parsing arguments for \dc@bsname.}%
2325 \@nx\fi
2326 \fi
2327 \@nx\@dc@{\the\toks@}%
2328 in the braces appear the argument catchers.
2329 \@xa\@nx\csname\dc@bsname\endcsname
2330 \ifx\@dc@outer@\outer
2331 \@xa\@nx\csname\@xa\gobble\string#1\endcsname%
2332 note the
2333 blank space after #1! If the command is to be \outer, we can't put
2334 it itself, so we put another, whose name is the same plus a space.
2335 Anyway, since #1 becomes \protected, this case will never be ex-
2336 ecuted.
2337 \else\@nx#1%
2338 \fi

```

```

2337 }%
2338 \edef\gmu@reserveda{%
2339 \@dc@global@% perhaps \global
2340 \long\def\@xa\@nx\csname\string#1\endcsname% for
2341 a command \command, define \\\command
2342 \the\@temptokena% it's #1#2#3...
2343 {%
2344 \@dc@env@argstoend% if we define an environment, this macro will
2345 provide passing the arguments to the end macro (see line 2409).
2346 \unexpanded{#4}%
2347 \@dc@esphack@}%
2348 }% of \edef.
2349 \gmu@reserveda
2350 }% of \DeclareCommand.

```

Now we have a convenient tool to implement parsing of a Knuthian argument(s).

Knuthian argument's default replacement

```
\@dc@K@defaultrepl 2359 \def\@dc@K@defaultrepl{##1}
```

2009/11/22 inner pair of braces removed (as leading to additional group causing errors).

```
\@dc@define@K 2363 \DeclareCommand\@dc@define@K\long{mb}{%
2364 first we add the particular CS
2365 to the toks register:
2366 \edef\gmu@tempa{%
2367 \unexpanded{\toks@\@xa}%
2368 {\@nx\the\toks@{%
2369 \@xa\@nx\csname\dc@bsname@K\the\@dc@catchernum%
2370 \endcsname}}%
2371 }\gmu@tempa
```

and define this macro:

```

2371 \edef\gmu@tempa{%
2372 \def\@xa\@nx\csname\dc@bsname@K\the\@dc@catchernum%
2373 \endcsname
2374 \unexpanded{#1}{%
2375 \@nx\@dc@addargum{%
2376 \IfValueTF{#2}{\unexpanded{#2}}{\@xau{%
2377 \@dc@K@defaultrepl}}}}%
2378 }% we add the Knuthian replacement to the toks register as #n.
2379 \@nx\@dc@parse@next\@dc@arguments% and continue parsing.
2380 }% of \edef. Now we execute this temporary macro, which means we \def\<command>@K<n>
2381 \@dc@global@% set properly before \@dc#3X.
2382 \if\P\@dc@long@\relax\long\fi
2383 \gmu@tempa

```

Now we clean up and continue construction of arguments parser.

```

2385 \emptyify\@dc@long@
2386 \@dc
2387 }
```

```
\DeclareEnvironment
```

```

2390 \DeclareCommand\DeclareEnvironment\long
2391 {m% Pm the environment's name; you may wonder why it allows \par. Once I met
2392 an environment with a name containing
2393 % \string\par.
```

```

2394 Q{\outer\global\protected\long!1L.qQiIwW\sphack\envhack}%
2395 a sequence of \TeX's prefixes and/or specifiers to make all the arguments
2396 'long' (\long, !, 1 or L) and/or not to place the message issuer in the
2397 command (., q, Q) and/or to make the environment ignore spaces fol-
2398 lowing its end (i, I, w, W or \sphack),
2399 m\uuu% the arguments specification (both for the begin and the end definitions),
2400 m\u% the begin definition,
2401 m\u% the end definition; it can contain any #<n>—the arguments of the environ-
2402 ment are passed to it, too.
2403 }{%
2404 \def\@dc@env@argstoend{%
2405 \edef
2406 {\@xa\@nx\csname\bslash#1(arguments)\endcsname
2407 {\unexpanded{%
2408 \@xa\@xa\@xa\unexpanded\@xa\@xa\@xa{\@xa\@gobble%
2409 \the\@dc@arguments}%
2410 }% of outer \unexpanded,
2411 }% of \edef's body.
2412 }
2413 Note that \@dc@env@argstoend will be put in an \edef and the arguments bearer
2414 it sets will be defined as custom for this particular environment (its name contains the
2415 name of the environment) and inside the environment's group and will bear a list of
2416 tokens {arg.1} {arg.2} ... not the CS \@dc@arguments so it's robust to nested envi-
2417 ronments and even to \DeclareEnvironment's occurrences inside the environment.
2418 }% of \@dc@env@argstoend.

2419 \def\gmu@reserveda{\envhack}% we add the information we define an en-
2420 vironment.
2421 \IfValueT{#2}{%
2422 {\addtomacro\gmu@reserveda{#2}}% we pass all the 'prefixes' to \Decl |
2423 % areCommand
2424
2425 {\@xa\DeclareCommand\csname#1\@xa\endcsname
2426 \gmu@reserveda{#3}{#4}}%
2427
2428 Now the begin definition is done. Let's get down to the end definition.
2429 \@dc@global@% note this CS was for sure redefined by the last \Declare |
2430 % Command (and by nothing else) and that's exactly what we want.
2431 \nameedef{end#1}{% It's \edef inside.
2432 {\@nx\@xa\@xa\@nx\csname\bslash\end#1\endcsname
2433 {\@xa\@nx\csname\bslash#1(arguments)\endcsname}%
2434
2435 We \edef the \end<env. name> macro to be
2436 \expandafter\end<env. name>\<env. name>(arguments)
2437 (it's \expandafter and two control sequences, the latter with (arguments) in its
2438 name).
2439 \IfIntersect{#2}{iIwW\sphack}{%
2440 {\@dc@global@\@xa\addtomacro
2441 \csname\end#1\endcsname{\@ignoretrue}}{}%
2442 \edef\gmu@reserveda{%
2443 \@dc@global@\long\def\% the inner end macro is always \long and
2444 perhaps defined \globally.
2445 \@xa\@nx\csname\bslash\end#1\endcsname
2446 \the\@temptokena% it's #1#2#3...—the inner end macro takes the same
2447 number of parameters as the begin macro.
2448 }

```

```

2458 { % the definition body
2459 \unexpanded{\#5} } %
2460 } % of \edef.
2461 \gmu@reserveda
2462 }

```

### Ampulex Compressa-like modifications of macros

Ampulex Compressa is a wasp that performs brain surgery on its victim cockroach to lead it to its lair and keep alive for its larva. Well, all we do here with the internal L<sup>A</sup>T<sub>E</sub>X macros resembles Ampulex's actions but here is a tool for a replacement of part of macro's definition.

The `\ampulexdef` command takes its #2 which has to be a macro and replaces part of its definition delimited with #5 and #6 with the replacement #7. The redefinition may be prefixed with #1. #2 may have parameters and for such a macro you have to set the parameters string and arguments string (the stuff to be taken by the one-step expansion of the macro) as the optional [#3] and [#4]. . If `\ampulexdef` doesn't find the start and end tokens in the meaning of the macro, it does nothing to it. You have to write ##### instead of # or you can use `\ampulexhash` as well. For an example use see line [3366](#).

```

\ampulexdef 2510 \DeclareCommand\ampulexdef\long
2511 {Q{\outer\long\global\protected} } % (1) (optional) prefix(es); allowed is
any sequence of them in any order, just like for the original TEX's \def.
2514 S {\def\edef\gdef\xdef\pdef} } % (2) (optional) kind of definition; if not
specified, \def will be used.
2516 m } % (3) macro to be redefined,
2517 O {} } % (4) \def's parameters string; empty by default,
2518 O {} } % (5) definition body's parameters to be taken in a one-step expansion of
the redefined macro; empty by default,
2520 m } % (6) start token(s),
2521 m } % (7) end token(s)
2522 m } % (8) the replacement
2523 } { % For the example of usage see 3366.
2530 \def\gmu@tempa {\#6} %
2531 \def\gmu@tempb {\#7} %
2532 \def\gmu@tempc {\#8} % we wrap the start, end and replacement tokens in
macros to avoid unbalanced \ifs.
2534 \edef\gmu@tempd {%
2535 \long\def\@nx\gmu@tempd
2536 #####1\detoken@xa\gmu@tempa
2537 #####2\detoken@xa\gmu@tempb
2538 #####3\@nx\gmu@tempd{%
2539 \unexpanded{%
2541 \@ifempty{\#3}{\hidden@iffalse}{\hidden@iftrue}}}} } %
2543 \gmu@tempd% it defines \gmu@tempc to produce an open \if depending on
whether the start and end token(s) are found in the meaning of #3.
2547 \edef\gmu@tempe {%
2548 \@nx\gmu@tempd\all@other#3% note that #3 may have parameters so we
have to look at its meaning not at its one-level expansion, which could
produce an 'extra \}' error.
2551 \detoken@xa\gmu@tempa

```

```

2552 \detoken@xa \gmu@tempb \@nx \gmu@tempd
2553 }%
2555 \gmu@tempe% we apply the checker and it produces an open \if.
2557 \edef\gmu@tempd{%
2558 \long\def\@nx\gmu@tempd
2559 #####1\@xa\unexpanded\@xa{\gmu@tempa}%
2560 #####2\@xa\unexpanded\@xa{\gmu@tempb}%
2561 #####3\@nx\ampulexdef{%
2562 we define a temporary macro with the parameters delimited with the 'start' and 'end' parameters of \ampulexdef.
2563 \@nx\unexpanded{#####1}%
2564 \@nx\@xa\@nx\unexpanded
2565 \@nx\@xa{\@nx\gmu@tempc}%
2566 we replace the part of the redefined macro's
2567 meaning with the replacement text.
2568 \@nx\unexpanded{#####3}}}%
2569 \gmu@tempd
2570 \edef\gmu@tempf{#5}%
2571 \edef\gmu@tempe{%
2572 \IfValueT{#1}{#1}\IfValueTF{#2}{#2}{\def}%
2573 \@nx#3#4{%
2574 \@xa\@xa\@xa\gmu@tempd\@xa#3\gmu@tempf\ampulexdef}%
2575 \gmu@tempe
2576 \fi}
2577 \ampulexhash 2581 \def\ampulexhash{####}%
2582 for your convenience (not to count the hashes).

```

For the heavy debugs I was doing while preparing gmdoc, as a last resort I used `\showlists`. But this command alone was usually too little: usually it needed setting `\showboxdepth` and `\showboxbreadth` to some positive values. So,

```
\gmshowlists 2588 \def\gmshowlists{\showboxdepth=1000\showboxbreadth=1000%
2589 \showlists}
```

```
\nameshow 2593 \newcommand\nameshow[1]{\@xa\show\csname#1\endcsname}
\nameshowthe 2594 \newcommand\nameshowthe[1]{\@xa\showthe\csname#1\endcsname}
```

Note that to get proper `\showthe\my@dimen14` in the ‘other’ @’s scope you write `\nameshowthe{\my@dimen}14`.

Standard `\string` command returns a string of ‘other’ chars except for the space, for which it returns `\space`. In gmdoc I needed the spaces in macros’ and environments’ names to be always `\space`, so I define

```
\xiistring 2604 \long\def\xiistring#1{%
2605 \if\@nx#1\xiispace
2606 \xiispace
2607 \else
2608 \afterfi{\string#1}%
2609 to make the same error as bare \string would
2610 cause in case of empty #1.
2611 \fi}
```

The next macro is applied to a `\detokenized` nonempty string to convert the spaces into ‘other’.

```
@xiispaces 2614 \def\@xiispaces#1\#2\@nil{%
2615 \#1%
2616 \ifx\@xiispaces#2\@xiispaces
2617 \else
```

```

2618 \xiinspace
2619 \afterfi{@xiispaces#2@@nil}%
2620 \fi}

```

## Environments redefined

### Almost an environment or redefinition of \begin

We'll extend the functionality of \begin: the non-starred instances shall act as usual and we'll add the starred version. The difference of the latter will be that it won't check whether the 'environment' has been defined so any name will be allowed.

This is intended to structure the source with named groups that don't have to be especially defined and probably don't take any particular action except the scoping.

(If the \begin\*'s argument is a (defined) environment's name, \begin\* will act just like \begin.)

Original L<sup>A</sup>T<sub>E</sub>X's \begin:

```

\def\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}

\@begnamedgroup 2652 \long\def\@begnamedgroup#1{%
 2653 \edef\@prevgrouplevel{\the\currentgrouplevel}% added 2009/03/24
 to handle special pseudo-environments that don't increase \current|
 grouplevel(such as document). Note it's \edefed outside the environ-
 ment's group.
 2657 \@ignorefalse% not to ignore blanks after group
 2658 \begingroup\@endpefalse
 2659 \edef\@prevenvir{\@currenvir}% Note we \edef it inside the group (for
 obvious reason), unlike the 'previous' grouplevel.
 2661 \edef\@currenvir{#1}% We could do recatcoding through \string or
 % \detokenize but all the name 'other' and 10 could affect a thousand
 packages so we don't do that and we'll recatcode in a testing macro, see line
 2748.
 2666 \edef\@currenvline{\on@line}%
 2667 \csname#1\endcsname}% if the argument is a command's name (an environ-
 ment's e.g.), this command will now be executed. (If the corresponding
 control sequence hasn't been known to TEX, this line will act as \relax.)

```

Let us make it the starred version of \begin.

```

\begin* 2676 \def\begin{\g@ifstar{\@begnamedgroup}{%
 \begin
 2677 \@begnamedgroup@ifcs}}
\@begnamedgroup@ifcs 2680 \def\@begnamedgroup@ifcs#1{%
 2681 \ifcsname#1\endcsname\afterfi{@begnamedgroup{#1}}%
 2682 \else\afterfi{@latex@error{Environment #1 undefined}}%
 \@eha}%
 2683 \fi}%

```

### \@ifenvir and improvement of \end

It's very clever and useful that `\end` checks whether its argument is `\ifx`-equivalent `\@currenvir`. However, in standard L<sup>A</sup>T<sub>E</sub>X it works not quite as I would expect: Since the idea of environment is to open a group and launch the CS named in the `\begin`'s argument. That last thing is done with `\csname... \endcsname` so the catcodes of chars are irrelevant (until they are `\active`, `_`, `,` etc.). Thus should be also in the `\end`'s test and therefore we ensure the compared texts are both expanded and made all 'other'.

First a (not expandable) macro that checks whether current environment is as given in #1. Why is this macro `\long`?—you may ask. It's `\long` to allow environments such as `\string\par`.

```

\@ifenvir 2707 \long\def\@ifenvir#1{%
 % #1 enquired environment name which will be confronted with \@cur-
 renvir
 % #2 what if true (if the names are equivalent4)
 % #3 what if false
2719 \@ifedetokens {\@currenvir} {#1} }

\@ifedetokens 2722 \long\pdef\@ifedetokens#1#2{%
 % #1 first list of tokens to be expanded and detokenized
 % #2 second list
 % #3 if agree
 % #4 else
2736 \edef\gmu@tempa{#1} % to get #1 fully expanded.
2737 \edef\gmu@reserveda{\@xa\detokenize\@xa{\gmu@tempa}}% with our
 brave new \begin, \@currenvir is fully expanded, remember?
2740 \edef\gmu@tempa{#2} % to get #2 fully expanded.
2741 \edef\gmu@reservedb{\@xa\detokenize\@xa{\gmu@tempa}}%
2742 \ifx\gmu@reserveda\gmu@reservedb\@xa\@firstoftwo
2743 \else\@xa\@secondoftwo
2744 \fi}

\@ifjobname 2746 \def\@ifjobname#1{\@ifedetokens {\jobname} {#1} }

\@ifprevenvir 2748 \long\def\@ifprevenvir#1{%
 % #1 enquired environment name which will be confronted with \@pre-
 venvir
 % #2 what if true (if the names are equivalent5)
 % #3 what if false
2760 \@ifedetokens {\@prevenvir} {#1} }

```

Note that `\@ifjobname` and `\@ifenvir` are expandable and in an `\edef` they expand to

`\@ifedetokens {\<current jobname>} {\<arg.1>}`

and

`\@ifedetokens {\<current envir>} {\<arg.1>}`

resp. which may be useful for some TeXvert.

```
\@checkend 2771 \def\@checkend#1{\@ifenvir{#1}{}{\@badend{#1}}}
```

---

<sup>4</sup> The names are checked whether they produce the same `\csname`. They don't have to have the same catcodes.

<sup>5</sup> The names are checked whether they produce the same `\csname`. They don't have to have the same catcodes.

Thanks to it you may write `\begin{macrocode*}` with  $*_{12}$  and end it with `\end{macrocode*}` with  $*_{11}$  (that was the problem that led me to this solution). The error messages looked really funny:

```
! LaTeX Error: \begin{macrocode} on input line 1844 ended
by \end{macrocode*}.
```

You might also write also `\end{macrocode}\star` where `\star` is defined as 'other' star or letter star.

### Storing and restoring the catcodes of specials

```
\gmu@storespecials 2787 \DeclareCommand\gmu@storespecials{o}{% we provide a possibility of adding
 stuff. For usage see line ??.
\gmu@restorespecials 2789 \def\do##1{\catcode`@nx##1=\the\catcode`##1\relax}%
2790 \edef\gmu@restorespecials{%
2791 \dospecials\do\^\^M}\IfValueT{#1}{#1}}
\gmu@septify 2793 \pdef\gmu@septify{%
 restoring the standard catcodes of specials. The name is
 the opposite of 'sanitize' :-). It restores also the original catcode of ^^M.
2796 \def\do{\relax\catcode`}%
2797 \do__10\do__0\do__1\do__2\do__3\do__4%
2798 \do__6\do__7\do__8\do__14\do__13\do\^\^M5\relax
%% \let\do\@makeother
%% \do\do\do\do\do\do\do\do\do\relax
2801 }
```

### From `relsize`

As file `relsize.sty`, v3.1 dated July 4, 2003 states, L<sup>A</sup>T<sub>E</sub>X 2<sub>E</sub> version of these macros was written by Donald Arseneau [asnd@triumf.ca](mailto:asnd@triumf.ca) and Matt Swift [swift@bu.edu](mailto:swift@bu.edu) after the L<sup>A</sup>T<sub>E</sub>X 2.09 `smaller.sty` style file written by Bernie Cosell [coseill@WILMA.BBN.COM](mailto:coseill@WILMA.BBN.COM).

I take only the basic, non-math mode commands with the assumption that there are the predefined font sizes.

```
\relsize You declare the font size with \relsize{<n>} where <n> gives the number of steps
 ("mag-step" = factor of 1.2) to change the size by. E.g., n = 3 changes from \normal-
\smaller size to \LARGE size. Negative n selects smaller fonts. \smaller == \relsize{-1};
\larger == \relsize{1}. \smallerr(my addition) == \relsize{-2};
\largerr guess yourself.
```

(Since \DeclareRobustCommand doesn't issue an error if its argument has been defined and it only informs about redefining, loading `relsize` remains allowed.)

```
\relsize 2831 \pdef\relsize#1{%
2832 \ifmmode\@nomath\relsize\else
2833 \begingroup
2834 \tempcnta\% assign number representing current font size
2835 \ifx\@currsize\normalsize\else\% funny order is to have
 most ...
2836 \ifx\@currsize\small\else\% ...likely sizes checked
 first
2837 \ifx\@currsize\footnotesize\else\%
2838 \ifx\@currsize\large\else\%
2839 \ifx\@currsize\Large\else\%
```

```

2840 \ifx\@currsize\Large\else
2841 \ifx\@currsize\scriptsize\else
2842 \ifx\@currsize\tiny\else
2843 \ifx\@currsize\huge\else
2844 \ifx\@currsize\Huge\else
2845 4\rs@unknown@warning% unknown state: \normal!
2846 size as starting point
2847 \fi\fi\fi\fi\fi\fi\fi\fi\fi

```

Change the number by the given increment:

```
2848 \advance\@tempcnta#1\relax
```

watch out for size underflow:

```

2850 \ifnum\@tempcnta<\z@\rs@size@warning{small}{\string%
2851 \tiny}\@tempcnta\z@\fi
2852 \@xa\endgroup
2853 \ifcase\@tempcnta% set new size based on altered number
2854 \tiny\or\scriptsize\or\footnotesize\or%
2855 \small\or\normalsize\or
2856 \large\or\Large\or\LARGE\or\huge\or\Huge
2857 \else
2858 \rs@size@warning{large}{\string\Huge}\Huge
2859 \fi\fi}% end of \relsize.

\rs@size@warning 2858 \providecommand*\rs@size@warning[2]{\PackageWarning{gmutils}%
2859 (relsize) }{%
2860 \ifx\requested\is too#1.\MessageBreak Using #2 instead} }

\rs@unknown@warning 2862 \providecommand*\rs@unknown@warning{\PackageWarning{gmutils}%
2863 (relsize) }{Current font size
2864 \ifx\unknown\Why!?\MessageBreak Assuming \string%
2865 \normalsize}

```

And a handful of shorthands:

```

\larger 2867 \DeclareRobustCommand*\larger[1][\@ne]{\relsize{+\#1}}
\smaller 2868 \DeclareRobustCommand*\smaller[1][\@ne]{\relsize{-\#1}}
\textlarger 2869 \DeclareRobustCommand*\textlarger[2][\@ne]{\relsize{+\#1}\#2}
\textsmaller 2870 \DeclareRobustCommand*\textsmaller[2][\@ne]{\relsize{-\#1}\#2}
\largerr 2871 \pdef\largerr{\relsize{+2}}
\smallerr 2872 \pdef\smallerr{\relsize{-2}}

```

## Meta-symbols

I fancy also another Knuthian trick for typesetting *meta-symbols* in *The T<sub>E</sub>Xbook*. So I repeat it here. The inner `\meta` macro is copied verbatim from doc's v2.1b documentation dated 2004/02/09 because it's so beautifully crafted I couldn't resist. I only don't make it `\long`.

"The new implementation fixes this problem by defining `\meta` in a radically different way: we prevent hyphenation by defining a `\language` which has no patterns associated with it and use this to typeset the words within the angle brackets."

```
\meta 2894 \pdef\meta#1{%
```

"Since the old implementation of \meta could be used in math we better ensure that this is possible with the new one as well. So we use \ensuremath around \langle and \rangle. However this is not enough: if \meta@font@select below expands to \itshape it will fail if used in math mode. For this reason we hide the whole thing inside an \nfss@text box in that case."

```

2902 { \meta@fontsetting \ensuremath \langle }%
2903 \ifmmode \gdef \nfss@text \fi
2904 {%
2905 this has to be a begin-group because \nfss@text becomes \hbox in math
2906 mode.
2907 \gmu@activespaceblank
2908 \meta@font@select

```

Need to keep track of what we changed just in case the user changes font inside the argument so we store the font explicitly.

```

2915 #1 /%
2916 }%
2917 { \meta@fontsetting \ensuremath \rangle }%
2918 }% of \meta.
2919

\gmu@activespaceblank 2920 \pdef \gmu@activespaceblank {%
2921 \gmu@activespace { \space \ignorespaces }%
2922 note the subtle per-
2923 vosity of this definition: if we meet more than one subsequent active
2924 spaces, then the first of them will typeset \space and its \ignorespaces
2925 will gobble \space of the second and will stop at \ignorespaces and
2926 this \ignorespaces will gobble the next \space and so on.
2927 }

\meta@fontsetting 2928 \def \meta@fontsetting { \color { red!50!black } }
\meta@font@select 2929 \def \meta@font@select { \meta@fontsetting \it }


```

But I define \meta@font@select as the brutal and explicit \it instead of the original \itshape to make it usable e.g. in the gmdoc's \cs macro's argument.

The below \meta's drag<sup>6</sup> is a version of *The TeXbook*'s one.

```

\<...> 2945 \def \<#1> { \meta { #1 } }
\metachar 2947 \pdef \metachar#1 { \begingroup \metacharfont \#1 \endgroup }
\metacharfont 2948 \def \metacharfont { \meta@fontsetting \rm }

```

### Macros for printing macros and filenames

First let's define three auxiliary macros analogous to \dywiz from polski.sty: a short-hands for \discretionary that'll stick to the word not spoiling its hyphenability and that'll won't allow a line break just before nor just after themselves. The \discretionary primitive has three arguments: #1 'before break', #2 'after break', #3 'without break', remember?

```

\discre 2961 \pdef \discre#1#2#3 { \leavevmode \kernosp %
2962 \discretionary { #1 } { #2 } { #3 } \penalty10000 \hskiposp \relax }
\discret 2964 \pdef \discret#1 { \discre { #1 } { #1 } { #1 } }

```

---

<sup>6</sup> Think of the drags that transform a very nice but rather standard 'auntie' ('Tante' in Deutsch) into a most adorable Queen ;-).

A tiny little macro that acts like `\-` outside the math mode and has its original meaning inside math.

```

2968 \def\:-{%
2969 \ifmmode\afterfi{\mskip\medmuskip}%
2970 \else\afterfi{\discre{\null}{}}% \null to get \hyphenpenalty
 not \exhyphenpenalty.%
2972 \fi}
2976 \let\gmu@discretionaryhyphen\-\% for the cases when we don't redefine \
 but use \
\bihyphen 2980 \DeclareCommand\bihyphen{O{*}}{%
 a redefinition of \- that makes it optional-
 argument to allow further hyphenation
\nu@discretionaryhyphen 2982 \DeclareCommand\gmu@discretionaryhyphen{T{\#1}ca}{%
 \IfValueT{\#2}{%
 \@ifempty{\#2}{%
 \def\gmu@bihyphen@char{\#2}}%
 }%
 \IfValueT{\#3}{%
 \@ifempty{\#3}{%
 \def\gmu@bihyphen@corr{\#3}}%
 }%
 \IfValueTF{\#1}\discre\discretionary
 {%
 before break
 \IfValueTF{\#2}{%
 \@ifempty{\#2}{\gmu@bihyphen@char{\#2}}%
 }%
 \ifnum\hyphenchar>\z@%
 \char\hyphenchar\font
 \fi}%
 end of before break
 {%
 after break
 \IfValueT{\#3}{%
 \@ifempty{\#3}{\gmu@bihyphen@corr{\#3}}%
 }%
 }%
 {%
 without break
 }%
 almost as in The TEXbook: unlike The TEXbook, we allow hyphenchars ≥ 255
 as we are XETEX.%
}%
 of \DeclareCommand\-
3008 \gmu@storeifnotyet\-\% original \- is a TEX's primitive, therefore we should
 store it.
3010 \let\-\gmu@discretionaryhyphen
3011 \let\@dischyp\gmu@discretionaryhyphen\% to override framed.sty
3012 }% of \bihyphen
3015 \relaxen\gmu@bihyphen@corr
\vs 3017 \pdef\vs{\discre{\visiblespace}{}}{\visiblespace}}

```

Then we define a macro that makes the spaces visible even if used in an argument (i.e., in a situation where `\catcode`ing has no effect).

```

\printspaces 3023 \def\printspaces#1{{\let~=\vs\let~=\\vs\gm@pswords#1\%
 \@@nil}}
\gm@pswords 3025 \def\gm@pswords#1#2\@@nil{%

```

```

3026 \ifx\relax#1\relax\else#1\fi
3027 \ifx\relax#2\relax\else\vs\penalty\hyphenpenalty%
 \gm@pswords#2@@nil\fi}%
 note that in the recursive call
 of \gm@pswords the argument string is not extended with a sentinel
 space: it has been already by \printspaces.

\sfname 3032 \pdef\sfname#1{\textsf{\printspaces{#1}}}

gmu@discretionaryslash 3034 \def\gmu@discretionaryslash{\discre{}{}\hbox{}{}% the
 second pseudo-argument nonempty to get \hyphenpenalty
 not \exhyphenpenalty.

\file 3039 \pdef\file#1{\gmu@printslashes#1\gmu@printslashes}

\gmu@printslashes 3041 \def\gmu@printslashes#1/#2\gmu@printslashes{%
3042 \sfname{#1}%
3043 \ifx\gmu@printslashes#2\gmu@printslashes
3044 \else
3045 \textsf{\gmu@discretionaryslash}%
3046 \afterfi{\gmu@printslashes#2\gmu@printslashes}\fi}

it allows the spaces in the filenames (and prints them as \square).

The macro defined below I use to format the packages' names.

\pk 3053 \pdef\pk#1{\textsf{#1}}

Some (if not all) of the below macros are copied from doc and/or ltxdoc.

A macro for printing control sequences in arguments of a macro. Robust to avoid
writing an explicit \ into a file. It calls \ttfamily not \tt to be usable in headings
which are boldface sometimes.

\cs 3067 \DeclareCommand\cs{0{\type@bslash\penalty\@M\hskip\z@skip}}{%
 % [#1] 0{\bslash} the control sequence's prefix, by default it's \
 % allowing hyphenation of subsequent word,
 % #2 m the control sequence or anything to be typeset in typewriter font.

 3080 \begingroup
 3081 \ifdef\verbatim@specials\verbatim@specials\fi
 3082 \edef\-\{ \discretionary{%
 3083 \ifdef\gmv@hyphen\gmv@hyphen
 3084 \else\unexpanded{\normalfont-}\}%
 3085 \fi}\}{}\}%
 3086 \def\{{\type@brace\yeshy}\def\}{\char`\\}%
 3087 \narrativett
 3088 \edef\narrativett@storedhyphenchar{\the\hyphenchar\font}%
 3089 \hyphenchar\font=\ifdef\gmv@hyphenchar\gmv@hyphenchar
 3090 \else" A6\fi
 3091 \cs@inner{#1} }

\cs@inner 3093 \pdef\cs@inner#1#2{%
 3094 #1#2%
 3095 \hyphenchar\font=\narrativett@storedhyphenchar\relax
 3096 \endgroup}

\narrativett 3098 \def\narrativett{\ttfamily}%
such name because I introduce it to distinguish the narrative verbatims from the code in gmdoc.

\env 3101 \long\pdef\env{\cs[]}

And for the special sequences like $\wedge\wedge A$:
```

```

3105 \foone{\@makeother^}
\hat{h}at 3106 {\pdef\hat{h}{\cs[^\^]}}
3108 \AtBeginDocument{%
\hash 3109 \@ifpackageloaded{gmdoc}{\def\hash{\cs[\#]}}{}}

And one for encouraging line breaks e.g., before long verbatim words.

\possfil 3112 \def\possfil{\hfil\penalty1000\hfilneg}
\possvfil 3114 \def\possvfil{\vfil
3115 \penalty\numexpr
3116 \gmu@minnum{\clubpenalty+\widowpenalty}{9999}%
3126 \relax\% eaten by \gmu@maxnum
3127 \relax\% eaten by \numexpr
3128 \vfilneg}

\gmu@extremnum 3130 \long\def\gmu@extremnum#1#2#3{%
3131 % #1 inequality sign
3132 % #2 left side of comparison
3133 % #3 right side of comparison
3136 \xa\ifx\@firstoftwo\@nil\relax\% this complicated test is to al-
3137 low arguments beginning with some \if<...> as in \possvfil e.g.
3138 \xa\@firstoftwo
3139 \else\@xa\@secondoftwo
3140 \fi
3141 {#2}%
3142 {%
3143 \ifnum\numexpr#2\relax#1\numexpr#3\relax
3144 \xa\@firstoftwo
3145 \else\@xa\@secondoftwo
3146 \fi
3147 { \gmu@extremnum#1{#2}}%
3148 { \gmu@extremnum#1{#3}}%
3149 }%
3150 }%
3151 }

\gmu@maxnum 3153 \def\gmu@maxnum{\gmu@extremnum>}
\gmu@minnum 3154 \def\gmu@minnum{\gmu@extremnum<}

```

### Typesetting arguments and commands

\arg We define a conditional and iterating command \arg that in math mode does what it used to do was in math and outside math it typesets mandatory, optional and picture (parenthesized) and angled arguments and optional stars. You can write

```
\arg[gefilte]*<fisch>(mit) {baigele}
```

to get

```
[<gefilte>][*]{<fisz>}(<mit>) {<bajgele>}
```

or even

```
\verb+\MoltoAdagio\arg*{Dankgesang}<an>[die_Gottheit]+
```

(where \ is the escape char in verbatims) to get

```
\MoltoAdagio[*]{<Dankgesang>}<>[<die Gottheit>]
```

(in der lydischen Tonart).

For more complicated arguments configurations consider using gmdoc's environment `enumargs`.

The five macros below are taken from the `ltxdoc.dtx`.

`\cmd{\foo}` Prints `\foo` verbatim. It may be used inside moving arguments. `\cs{\foo}` also prints `\foo`, for those who prefer that syntax. (This second form may even be used when `\foo` is `\outer`).

`\cmd` 3182 `\long\def\cmd#1{\@xa\cs\@xa{\@xa\cmd@to@cs\string#1}}%` it has to  
be un `\protected!` It has so many `\expandafters` to allow `\cmd\par` and  
still keep the `\cs` command 'short'.

`\cmd@to@cs` 3186 `\def\cmd@to@cs#1#2{\char\number`#2\relax}`

It can be short since it never gets actual control sequence as an argument only a string  
of 'other' tokens (and maybe spaces).

`\marg{text}` prints `{<text>}`, 'mandatory argument'.

`\marg` 3192 `\pdef\marg#1{{\narrativett\type@lbrace}\meta{#1}{%`  
`\narrativett\char`\\}}`

`\oarg{text}` prints `[<text>]`, 'optional argument'. Also `\oarg{text}` does  
that.

`\oarg` 3198 `\pdef\oarg{\ifnextchar[\oargsq\oarg}`  
3200 `\pdef\oarg#1{{\narrativett[]}\meta{#1}{\narrativett}}`  
3201 `\pdef\oargsq[#1]{\oarg{#1}}`

`\parg{te,xt}` prints `(<te,xt>)`, 'picture mode argument'.

`\parg` 3205 `\pdef\parg{\ifnextchar(\pargp\parg}`  
3207 `\def\parg#1{{\narrativett[]}\meta{#1}{\narrativett}}`  
3208 `\def\pargp(#1){\parg{#1}}`

3210 `\pdef\aaarg{\ifnextchar<\aaarga\aaarg}`

3211 `\def\aaarg#1{{\narrativett<}\meta{#1}{\narrativett>}}`

3212 `\def\aaarga<#1>{\aaarg{#1}}`

3214 `\def\@verbaarga#1#2>{\aaarg{#2}\arg@dc}`

3216 `\foone{\catcode`>\active}{%`

3217 `\def\@verbaargact#1#2>{\aaarg{#2}\arg@dc}%`

3218 `}`

3220 `\foone{\makeother\{\makeother\}}%`

3221 `\catcode`[=\ne\catcode`]=\tw@}`

3222 `[%`

3223 `\def\@verbmargm#1#2}{% for an argument in curly braces in a verbatim,  
where the braces are not groupers and not necessary 'other'. We'll know  
by \ifnextif that the future token is an opening brace. Note this macro  
has 2nd parameter delimited with 'other' closing brace (so may not act  
correctly when braces are nested (then hide them with special verbatim  
groupers)).`

3229 `\marg[#2]%`

3230 `\arg@dc`

3231 `]%`

3232 `]% of \foone`

`\arg@dc` 3235 `\DeclareCommand\arg@dc!{%`

```

3238 S {\arg} % (1)
3239 O % (2)
3240 C % (3)
3241 B % (4)
3242 A % (5)
3243 S {\arg} % (6) just gobbled (for backwards compatibility)
3244 } % This command iterates while it has arguments and typesets them in brackets,
 parentheses or curly braces. Note it gobbles subsequent \args and just iter-
 ates.
3247 \def\next{o}%
3248 \IfValueT{#1}%
3249 { \metachar[\scanverb{*}\metachar] \def\next{#1} }%
3250 \IfValueT{#2} { \oarg{#2} \def\next{#1} }%
3251 \IfValueT{#3} { \parg{#3} \def\next{#1} }%
3252 \IfValueT{#4} { \marg{#4} \def\next{#1} }%
3253 \IfValueT{#5} { \aarg{#5} \def\next{#1} }%
3254 @ifnextchar\egroup{\endgroup}{%
3255 \if1\next\@xa\arg@dc
3256 \else% it's crucial that we look for verbatim braces after we checked there
 were no #4, otherwise there would be an error.
3257 \def\next{%
3258 \@ifnextif\xiilbrace{\@verbmargm}{%
3259 % not active or other lbrace
3260 \@ifnextif<% then we look for angles
3261 \ifnum\catcode`>=\active
3262 \@xa\@verbaargact
3263 \else\@xa\@verbaarga
3264 \fi}{%
3265 % and if not angles neither verbatim braces, then
3266 \endgroup% if we have no more arguments to typeset, we
 close the group opened in line 3286
3267 }%
3268 }%
3269 }%
3270 }%
3271 }%
3272 }%
3273 \@xa\next
3274 \fi
3275 }% of not \egroup
3276 }% of \arg@dc

```

Now define the front-end macro of the \arg command:

```

3280 \foone{\obeylines}{%
3281 \AtBeginDocument{%
3282 \let\math@arg\arg%
3283 \pdef\arg{\ifmmode\math@arg%
3284 \else\afterfi{%
3285 \begingroup%
3286 \ifdefined\@ifQueerEOL\@ifQueerEOL{%
3287 \def^{\unskip\space}% in the 'queer' EOLs scope we keep
 line end active in case we have \arg {<arg.>} ending a line: the
 next char peeper touches line end or, if the line end was , gob-
 bles the space it turns into so the comment layer would 'leak'
 to the code layer.
3288 }{}%
3289 \fi}%
3290 }%
3291 }%
3292 }%
3293 }%
3294 }%
3295 }%

```

```

3296 \arg@dc}%
3297 \fi}%
3298 }% of \AtBeginDocument,
3299 }% of \foone.

```

Now you can write

```

\arg{mand.\arg}{opt.\arg}{pict.\arg}
to get {\mand. arg} [{opt. arg}]{pict. arg}. (Yes, with only one \arg!)
And $ \arg(1+i) = \pi/4$ for $\arg(1 + i) = \pi / 4$.

```

```

\cat 3311 \DeclareCommand\cat{\O{"0123456789ABCDEF}}{%
3312 ${}_1\{\the\numexpr#1\}\m@th\$@\ifnextcat\space{}}

```

### Not only preamble!

Let's remove some commands from the list to erase at begin document! Primarily that list was intended to save memory not to forbid anything. Nowadays, when memory is cheap, the list of only-preamble commands should be rethought IMHO.

```

\not@onlypreamble 3328 \newcommand\not@onlypreamble[1]{{%
3329 \def\do##1{\ifx##1##1\else\@nx\do\@nx##1\fi}%
3330 \xdef\@preamblecmds{\@preamblecmds}%
3332 \not@onlypreamble\@preamblecmds
3333 \not@onlypreamble\@ifpackageloaded
3334 \not@onlypreamble\@ifclassloaded
3335 \not@onlypreamble\@ifl@aded
3336 \not@onlypreamble\@pkgextension

```

And let's make the message of only preamble command's forbidden use informative a bit:

```

\gm@notprerr 3341 \def\gm@notprerr{\can be used only in preamble (\online)}
3343 \AtBeginDocument{%
3344 \def\do##1{\@nx\do\@nx##1}%
3345 \edef\@preamblecmds{%
3346 \def\@nx\do##1{%
3347 \def##1{\@nx\PackageError{gmutils/LaTeX}{%
3348 {\@nx\string##1\@nx\gm@notprerr}\@nx\@eha}}%
3349 \@preamblecmds}}

```

A subtle error raises: the L<sup>A</sup>T<sub>E</sub>X standard \not@onlypreamble and what \document does with \@preamblecmds makes any two of 'only preamble' CS's \ifx-identical inside document. And my change makes any two CS's \ifx-different. The first it causes a problem with is standard L<sup>A</sup>T<sub>E</sub>X's \nocite that checks \ifx\@onlypreamble\document. So hoping this is a rare problem, we circumvent it. 2008/08/29 a bug is reported by Edd Barrett that with natbib an 'extra ' error occurs so we wrap the fix in a conditional.

```

\gmu@nocite@ampulex 3366 \def\gmu@nocite@ampulex{%
 we wrap the stuff in a macro to hide an open \if.
 And not to make the begin-input hook too large. the first optional argument
 is the parameters string and the second the argument for one-level expansion
 of \nocite. Both hash strings are doubled to pass the first \def.
3372 \ampulexdef\nocite[####1][{{####1}}]%
 note the double brace around
 % #3.
3374 \ifx

```

```

3375 { \onlypreamble \document }%
3376 \iftrue}
3379 \AtBeginDocument {\gmu@nocite@ampulex}%

```

### Third person pronouns

Is a reader of my documentations ‘she’ or ‘he’ and does it make a difference?

Previous versions of this documentation were consequently alternating ‘he’ and ‘she’ and provided specific macros for that purpose. Now I’m not that queer and gender so I take what is normally used, ‘they’ that is.

(The issue of human sexes and genders (certainly much more numerous than 2) is complex and delicate and a TeX macro package is probably not the best place to discuss it.)

```

\heshe 3428 \def \heshe {they}
\hisher 3429 \def \hisher {their}
\himher 3430 \def \himher {them}
\hishers 3431 \def \hishers {theirs}

\HeShe 3433 \def \HeShe {They}
\HisHer 3434 \def \HisHer {Their}
\HimHer 3435 \def \HimHer {Them}
\HisHers 3436 \def \HisHers {Theirs}

```

### Improvements to mwcls sectioning commands

That is, ‘Experi-mente’<sup>7</sup> mit MW sectioning & `\refstepcounter` to improve mwcls’s cooperation with hyperref. They shouldn’t make any harm if another class (non-mwcls) is loaded.

We `\refstep` sectioning counters even if the sectionings are not numbered, because otherwise

1. pdfTeX cried of multiply defined `\labels`,
2. e.g. in a table of contents the hyperlink `<rozdzia\l\_\_Kwiaty\_polskie>` linked not to the chapter’s heading but to the last-before-it change of `\ref`.

```

3454 \AtBeginDocument {%
 because we don't know when exactly hyperref is loaded and
 maybe after this package.
}
```

```

NoNumSecs 3456 \@ifpackageloaded{hyperref}{\newcounter{NoNumSecs}}%
3457 \setcounter{NoNumSecs}{617}%
 to make \refing to an unnumbered
 section visible (and funny?).

```

```

gm@hyperrefstepcounter 3459 \def \gm@hyperrefstepcounter {\refstepcounter{NoNumSecs}}%
\gm@targetheading 3460 \pdef \gm@targetheading#1{%
 \hypertarget{\#1}{\#1}}%
 end of then

```

```

gm@hyperrefstepcounter 3461 { \def \gm@hyperrefstepcounter {}%
\gm@targetheading 3462 \def \gm@targetheading#1{\#1}}%
 end of else

```

```

3463 }%
3464 }%
 of \AtBeginDocument

```

Auxiliary macros for the kernel sectioning macro:

```

3467 \def \gm@dontnumbersectionsoutofmainmatter{%
3468 \if@mainmatter \else \HeadingNumberedfalse \fi}
3469 \def \gm@clearpagesduetoopenright{%

```

---

<sup>7</sup> A. Berg, Wozzeck.

```
3470 \if@openright\cleardoublepage\else\clearpage\fi}
```

To avoid \defing of \mw@sectionxx if it's undefined, we redefine \def to gobble the definition and restore the original meaning of itself.

Why shouldn't we change the ontological status of \mw@sectionxx (not define if undefined)? Because some macros (in gmdocc e.g.) check it to learn whether they are in an mwcls or not.

But let's make a shorthand for this test since we'll use it three times in this package and maybe also somewhere else.

```
\@ifnotmw 3483 \long\def\@ifnotmw#1#2{\gm@ifundefined{mw@sectionxx}{#1}{#2}}
```

The kernel of MW's sectioning commands:

```
3508 \@ifnotmw{}{%
3509 \def\mw@sectionxx#1#2[#3]#4{%
3510 \edef\mw@HeadingLevel{\csname\#1@level\endcsname%
3511 \space}\% space delimits level number!
3512 \ifHeadingNumbered
3513 \ifnum\mw@HeadingLevel>\c@secnumdepth\%
3514 \HeadingNumberedfalse\fi
3515 line below is in \gm@ifundefined to make it work in classes other than mwbk
3516 \gm@ifundefined{if@mainmatter}{}{%
3517 \gm@dontnumbersectionsoutofmainmatter}
3518 \fi
3519 }
```

```
% \ifHeadingNumbered
% \refstepcounter{#1}%
% \protected@edef\HeadingNumber{\csname
% the#1\endcsname\relax}%
% \else
% \let\HeadingNumber\empty
% \fi
```

```
\HeadingRHeadText 3526 \def\HeadingRHeadText{#2}%
\HeadingTOCText 3527 \def\HeadingTOCText{#3}%
\HeadingText 3528 \def\HeadingText{#4}%
\mw@HeadingType 3529 \def\mw@HeadingType{#1}%
3530 \if\mw@HeadingBreakBefore
3531 \if@specialpage\else\thispagestyle{closing}\fi
3532 \gm@ifundefined{if@openright}{}{%
3533 \gm@clearpagesduetoopenright}%
3534 \if\mw@HeadingBreakAfter
3535 \thispagestyle{blank}\else
3536 \thispagestyle{opening}\fi
3537 \global\@topnum\z@
3538 \fi% of \if\mw@HeadingBreakBefore
```

placement of \refstep suggested by me (GM):

```
3540 \ifHeadingNumbered
3541 \refstepcounter{#1}%
3542 \protected@edef\HeadingNumber{\csname\#1\endcsname%
3543 \relax}%
3544 \else
3545 \let\HeadingNumber\empty
3546 \fi% of \ifHeadingNumbered
```

```

3548 \if\mw@HeadingRunIn
3549 \mw@runinheading
3550 \else
3551 \if\mw@HeadingWholeWidth
3552 \if@twocolumn
3553 \if\mw@HeadingBreakAfter
3554 \onecolumn
3555 \mw@normalheading
3556 \pagebreak\relax
3557 \if@twoside
3558 \null
3559 \thispagestyle{blank}%
3560 \newpage
3561 \fi% of \if@twoside
3562 \twocolumn
3563 \else
3564 \atopnewpage[\mw@normalheading]%
3565 \fi% of \if\mw@HeadingBreakAfter
3566 \else
3567 \mw@normalheading
3568 \if\mw@HeadingBreakAfter\pagebreak\relax\fi
3569 \fi% of \if@twocolumn
3570 \else
3571 \mw@normalheading
3572 \if\mw@HeadingBreakAfter\pagebreak\relax\fi
3573 \fi% of \if\mw@HeadingWholeWidth
3574 \fi% of \if\mw@HeadingRunIn
3575 }

```

### An improvement of MW's \SetSectionFormatting

A version of MW's \SetSectionFormatting that lets to leave some settings unchanged by leaving the respective argument empty ({} or []).

Notice: If we adjust this command for new version of MWCLS, we should name it \SetSectionFormatting and add issuing errors if the inner macros are undefined.

- [#1] the flags, e.g. `breakbefore`, `breakafter`;
- #2 the sectioning name, e.g. `chapter`, `part`;
- #3 preskip;
- #4 heading type;
- #5 postskip

```

3599 \relaxen\SetSectionFormatting
3600 \newcommand*\SetSectionFormatting[5][\empty] {%
3601 \ifx\empty#1\relax\else% empty (not \empty!) #1 also launches \else.
3602 \def\mw@HeadingRunIn{\def\mw@HeadingBreakBefore{}%
3603 \def\mw@HeadingBreakAfter{\def\mw@HeadingWholeWidth{%
3604 \def\mw@processflags{\relax}%
3605 \ifempty{#1} {} {\mw@processflags#1, \relax}%
3606 \ifempty{#1} {} {\def\mw@processflags{\relax}%
3607 \fi%
3608 \gm@ifundefined{#2} {\namedef{#2}{\mw@section{#2}}} {}%
3609 \mw@secdef{#2}{\preskip}{#3}{#2}{oblig.}}%
3610 }%
3611 }%
3612 }%
3613 }
```

```

3610 \mw@secdef{\#2}{@head}{\#4}{3\oblig.}%
3611 \mw@secdef{\#2}{@postskip}{\#5}{4\oblig.}%
3612 \ifx\empty\relax
3613 \mw@secundef{\#2@flags}{1 (optional)}%
3614 \else\mw@setflags{\#2}%
3615 \fi}

\mw@secdef 3617 \def\mw@secdef#1#2#3#4{%
 % #1 the heading name,
 % #2 the command distincer,
 % #3 the meaning,
 % #4 the number of argument to error message.
 3624 \@ifempty{#3}
 3625 {\mw@secundef{#1#2}{#4}}
 3626 {\@namedef{#1#2}{#3}}}

\mw@secundef 3628 \def\mw@secundef#1#2{%
 3629 \gm@ifundefined{#1}{%
 3630 \ClassError{mwcls/gm}{%
 3631 command\bslash#1\undefined\MessageBreak
 3632 after\bslash\SetSectionFormatting!!!\MessageBreak}%
 3633 Provide the #2 argument of \bslash
 3634 SetSectionFormatting.}}{}}

```

First argument is a sectioning command (wo. the backslash) and second the stuff to be added at the beginning of the heading declarations.

```

\addtoheading 3638 \def\addtoheading#1#2{%
 3639 \n@melet{\gmu@reserveda}{#1@head}%
 3640 \edef\gmu@reserveda{\unexpanded{#2}\@xa\unexpanded{%
 3641 \gmu@reserveda}}%
 3642 \n@melet{#1@head}{\gmu@reserveda}%
 3643 }
 3645 }% of \@ifnotmw's else.

```

### Negative \addvspace

When two sectioning commands appear one after another (we may assume that this occurs only when a lower section appears immediately after higher), we prefer to put the *smaller* vertical space not the larger, that is, the preskip of the lower sectioning not the postskip of the higher.

For that purpose we modify the very inner macros of MWCLS to introduce a check whether the previous vertical space equals the postskip of the section one level higher.

```
3657 \@ifnotmw{}{} We proceed only in MWCLS.
```

The information that we are just after a heading will be stored in the \gmu@prevsec macro: any heading will define it as the section name and \everypar (any normal text) will clear it.

```

\@afterheading 3662 \def\@afterheading{%
 3663 \nobreaktrue
 3664 \xdef\gmu@prevsec{\mw@HeadingType}% added now
 3665 \everypar{%
 3666 \relaxen\gmu@prevsec% added now. All the rest is original LATEX.
 3667 \if@nobreak

```

```

3668 \nobreakfalse
3669 \clubpenalty\@M
3670 \if@afterindent\else
3671 {\setbox\z@\lastbox}%
3672 \fi
3673 \else
3674 \clubpenalty\clubpenalty
3675 \everypar{}%
3676 \fi}%

```

If we are (with the current heading) just after another heading (one level lower I suppose), then we add the less of the higher header's post-skip and the lower header pre-skip or, if defined, the two-header-skip. (We put the macro defined below just before \addvspace in mwcls inner macros.)

```

\gmu@checkaftersec 3683 \def\gmu@checkaftersec{%
3684 \gm@ifundefined{\gmu@prevsec}{}{%
3685 \ifgmu@postsec% an additional switch that is true by default but may be
3686 turned into an \ifdim in special cases, see line 3721.
3687 {\@xa\mw@getflags\@xa{\gmu@prevsec}%
3688 \glet\gmu@reserved@{\mw@HeadingBreakAfter}%
3689 \if\mw@HeadingBreakBefore\def\gmu@reserved@{11}\fi% if the cur-
3690 rent heading inserts page break before itself, all the play with vskips is
3691 irrelevant.
3692 \if\gmu@reserved@else
3693 \penalty10000\relax
3694 \skip\z@=\csname\gmu@prevsec\@postskip\endcsname\relax
3695 \skip\tw@=\csname\mw@HeadingType\@preskip\endcsname\relax
3696 \gm@ifundefined{\mw@HeadingType\@twoheadskip}{%
3697 \ifdim\skip\z@>\skip\tw@
3698 \vskip-\skip\z@% we strip off the post-skip of previous header if it's
3699 bigger than current pre-skip
3700 \else
3701 \vskip-\skip\tw@% we strip off the current pre-skip otherwise
3702 \fi} % But if the two-header-skip is defined, we put it
3703 \penalty10000
3704 \vskip-\skip\z@
3705 \penalty10000
3706 \vskip-\skip\tw@
3707 \penalty10000
3708 \vskip-\skip\tw@
3709 \penalty10000
3710 \vskip\csname\mw@HeadingType\@twoheadskip\endcsname
3711 \relax}%
3712 \penalty10000
3713 \hrule\height\z@\relax% to hide the last (un)skip before
3714 subsequent \addvspace.
3715 \penalty10000
3716 \fi
3717 \fi
3718 }% of \gm@ifundefined{\gmu@prevsec} 'else'.
3719 }% of \def\gmu@checkaftersec.

\ParanoidPostsec 3721 \def\ParanoidPostsec{%
3722 this version of \ifgmu@postsec is intended for the
3723 special case of sections which may contain no normal text, as while gmdocing.
\ifgmu@postsec 3724 \def\ifgmu@postsec{%
3725 note this macro expands to an open \if.

```

```

3725 \skip\z@=\csname\gmu@prevsec\relax
3726 \ifdim\lastskip=\skip\z@\relax% we play with the vskips only if the
 last skip is the previous heading's postskip (a counter-example I met
 while gmdocing).
3730 }
3732 \let\ifgmu@postsec\iftrue
\gmu@getaddvs 3734 \def\gmu@getaddvs#1\addvspace#2\gmu@getaddvs{%
3735 \toks\z@={#1}%
3736 \toks\tw@={#2}}

```

And the modification of the inner macros at last:

```

\gmu@setheading 3739 \def\gmu@setheading#1{%
3740 \xa\gmu@getaddvs#1\gmu@getaddvs
3741 \edef#1{%
3742 \the\toks\z@\@nx\gmu@checkaftersec
3743 \@nx\addvspace\the\toks\tw@}}
3745 \gmu@setheading\mw@normalheading
3746 \gmu@setheading\mw@runinheading
\SetTwoheadSkip 3748 \def\SetTwoheadSkip#1#2{\@namedef{#1@twoheadskip}{#2}}
3750 }% of \@ifnotmw's else.

```

### My heading setup for mwcls

The setup of heading skips was tested in ‘real’ typesetting, for money that is. The skips are designed for 11/13 pt leading and together with my version of mw11.clo option file for mwcls make the headings (except paragraph and subparagraph) consist of an integer number of lines. The name of the declaration comes from my employer, “Wiedza Powszechna” Editions.

```

\@ifnotmw{}% We define this declaration only when in mwcls.
\WPheadings 3762 \DeclareCommand\WPheadings{T{\chapter}}{%
3763 \SetSectionFormatting[breakbefore,wholewidth]
3764 {part}{\z@\@plus1fill}{\z@\@plus3fill}%
3765 \IfValueF{#1}{%
3766 \gm@ifundefined{chapter}{%
3767 \SetSectionFormatting[breakbefore,wholewidth]
3768 {chapter}{%
3769 {66\p@}{67\p@} for Adventor/Schola 0,95.
3770 {\FormatHangHeading{\LARGE}}
3771 {27\p@\@plus0,2\p@\@minus1\p@}%
3772 }%
3773 }%
3774 }%
3775 }% of unless #1
3776 \SetTwoheadSkip{section}{27\p@\@plus0,5\p@}%
3777 \SetSectionFormatting{section}
3778 {24\p@\@plus0,5\p@\@minus5\p@}%
3779 {\FormatHangHeading{\Large}}
3780 {10\p@\@plus0,5\p@}%
3781 ed. Krajewska of “Wiedza Powszechna”, as we
 understand her, wants the skip between a heading and text to be rigid.
3782 \SetTwoheadSkip{subsection}{11\p@\@plus0,5\p@\@minus1\p@}%
3783 \SetSectionFormatting{subsection}

```

```

3787 {19\p@ \@plus0,4\p@ \@minus6\p@}
3788 {\FormatHangHeading{\large} }% 12/14 pt
3789 {6\p@ \@plus0,3\p@ }% after-skip 6 pt due to p.12, not to squeeze the
 before-skip too much.

3792 \SetTwoheadSkip{subsubsection}{10\p@ \@plus1,75\p@ \@minus1%
 \p@ }%
3793 \SetSectionFormatting{subsubsection}
3794 {10\p@ \@plus0,2\p@ \@minus1\p@ }
3795 {\FormatHangHeading{\normalsize} }
3796 {3\p@ \@plus0,1\p@ }% those little skips should be smaller than you cal-
 calculate out of a geometric progression, because the interline skip en-
 larges them.

3800 \SetSectionFormatting[runin]{paragraph}
3801 {7\p@ \@plus0,15\p@ \@minus1\p@ }
3802 {\FormatRunInHeading{\normalsize} }
3803 {2\p@ }%
3805 \SetSectionFormatting[runin]{ subparagraph}
3806 {4\p@ \@plus1\p@ \@minus0,5\p@ }
3807 {\FormatRunInHeading{\normalsize} }
3808 {\z@ }%
3809 }% of \WPheadings
3810 }% of \@ifnotmw

```

### Compatibilising standard and mwcls sectionings

If you use Marcin Woliński's document classes (`mwcls`), you might have met their little queerness: the sectioning commands take two optional arguments instead of standard one. It's reasonable since one may wish one text to be put into the running head, another to the toc and yet else to the page. But the order of optionalities causes an incompatibility with the standard classes: MW section's first optional argument goes to the running head not to toc and if you've got a source file written with the standard classes in mind and use the first (and only) optional argument, the effect with `mwcls` would be different if not error.

Therefore I counter-assign the commands and arguments to reverse the order of optional arguments for sectioning commands when `mwcls` are in use and reverse, to make `mwcls`-like sectioning optionals usable in the standard classes.

With the following in force, you may both in the standard classes and in `mwcls` give a sectioning command one or two optional arguments (and mandatory the last, of course). If you give just one optional, it goes to the running head and to toc as in `scls` (which is unlike in `mwcls`). If you give two optionals, the first goes to the running head and the other to toc (like in `mwcls` and unlike in `scls`).

(In both cases the mandatory last argument goes only to the page.)

What more is unlike in `scls`, it's that even with them the starred versions of sectioning commands allow optionals (but they still send them to the Gobbled Tokens' Paradise).

(In `mwcls`, the only difference between starred and non-starred sec commands is (not) numbering the titles, both versions make a contents line and a mark and that's not changed with my redefinitions.)

```

3851 \@ifnotmw{%
 we are not in mwcls and want to handle mwcls-like sectionings i.e.,
 those written with two optionals.
}
```

```
\gm@secini 3854 \def\gm@secini{\gm@la}%
```

```

3855 \StoreMacros {\partmark\chaptermark\sectionmark%
3856 \subsectionmark\subsubsectionmark\paragraphmark}%
\gm@secxx 3857 \def \gm@secxx#1#2[#3]#4 {%
3858 \ifx \gm@secstar \empty
 a little trick to allow a special version of the heading just to the running head.

3861 \@namedef{\#1mark}{\#1} % we redefine \secmark to gobble its argument
 and to launch the stored true marking command on the appropriate argument.
3864 \storedcsname{\#1mark}{\#2}%
3865 \RestoreMacro{\#1mark} % after we've done what we wanted we restore original \#1mark.
3867 }%
\gm@secstar 3868 \def \gm@secstar{[#3]}% if \gm@secstar is empty, which means the sectioning command was written starless, we pass the 'true' sectioning command #3 as the optional argument. Otherwise the sectioning command was written with star so the 'true' s.c. takes no optional.
3873 \fi
3874 \@xa\@xa\csname\gm@secini#1\endcsname
3875 \gm@secstar{\#4}%
3877 }% we are in mwcls and want to reverse MW's optionals order i.e., if there's just one optional, it should go both to toc and to running head.
\gm@secini 3880 \def \gm@secini{\gm@mw}%
3882 \let \gm@secmarkh \gobble% in mwcls there's no need to make tricks for special version to running headings.
\gm@secxx 3885 \def \gm@secxx#1#2[#3]#4 {%
3886 \@xa\@xa\csname\gm@secini#1\endcsname
3887 \gm@secstar{\#2}[#3]{\#4}%
3888 }

\gm@sec 3890 \def \gm@sec{\@dblarg{\gm@secx{\#1}}}
\gm@secx 3891 \def \gm@secx{\#2}%
3892 \@ifnextchar[\{\gm@secxx{\#1}{\#2}\}{\gm@secxx{\#1}{\#2}{\#2}}% if there's only one optional, we double it not the mandatory argument.

\gm@straightensec 3896 \def \gm@straightensec{\% the parameter is for the command's name.
3897 \gm@ifundefined{\#1}{}% we don't change the ontological status of the command because someone may test it.
3899 \n@melet{\gm@secini{\#1}{\#1}}%
3900 \@namedef{\#1}{%
\gm@secstar 3901 \gm@ifstar{\def \gm@secstar{\star}\gm@sec{\#1}}{%
3902 \def \gm@secstar{} \gm@sec{\#1}}}}%
3903 }%

3905 \let \do \gm@straightensec
3906 \do {part} \do {chapter} \do {section} \do {subsection} \do {%
3907 subsubsection} \do {%
3908 \@ifnotmw{}{\do {paragraph}}% this 'straightening' of \paragraph with the standard article caused the 'TeX capacity exceeded' error. Anyway, who on Earth wants paragraph titles in toc or running head?

```

### **enumerate\* and itemize\***

We wish the starred version of `enumerate` to be just numbered paragraphs. But `hyperref`

redefines \item so we should do it a smart way, to set the L<sup>A</sup>T<sub>E</sub>X's list parameters that is.

(Marcin Woliński in mwcls defines those environments slightly different: his item labels are indented, mine are not; his subsequent paragraphs of an item are not indented, mine are.)

```

enumerate* 3923 \@namedef{enumerate*}{%
3924 \ifnum\@enumdepth>\thr@@
3925 \@toodeep
3926 \else
3927 \advance\@enumdepth\@ne
3928 \edef\@enumctr{enum\romannumeral\the\@enumdepth}%
3929 \@xa\list\csname\label\@enumctr\endcsname{%
3930 \partopsep\topsep\topsep\z@\leftmargin\z@
3931 \itemindent\parindent\% \advance\itemindent\labelsep
3932 \labelwidth\parindent
3933 \advance\labelwidth-\labelsep
3934 \listparindent\parindent
3935 \usecounter\@enumctr
3936 \def\makelabel##1{\hfil##1\hfil}%
3937 \fi}
3938 \@namedef{endenumerate*}{\endlist}

itemize* 3941 \@namedef{itemize*}{%
3942 \ifnum\@itemdepth>\thr@@
3943 \@toodeep
3944 \else
3945 \advance\@itemdepth\@ne
3946 \edef\@itemitem{labelitem\romannumeral\the\@itemdepth}%
3947 \@xa\list\csname\@itemitem\endcsname{%
3948 \partopsep\topsep\topsep\z@\leftmargin\z@
3949 \itemindent\parindent
3950 \labelwidth\parindent
3951 \advance\labelwidth-\labelsep
3952 \listparindent\parindent
3953 \def\makelabel##1{\hfil##1\hfil}%
3954 \fi}
3955 \@namedef{enditemize*}{\endlist}

```

## The logos

We'll modify The L<sup>A</sup>T<sub>E</sub>X logo now to make it fit better to various fonts.

```

3964 \let\oldLaTeX\LaTeX
3965 \let\oldLaTeXe\LaTeXe

3967 \pdef\TeX{T\kern-.1667em\lower.5ex\hbox{E}\kern-.125emX\@}
3968 \StoreMacro\TeX
3969 \AtBeginDocument{\RestoreMacro\TeX}

\DeclareLogo 3971 \newcommand*\DeclareLogo[3]{\relax}%
 % [#1] is for non-LATEX spelling and will be used in the PD1 encoding (to
 % make pdf bookmarks);
 % #2 is the command, its name will be the PD1 spelling by default,
 % #3 is the definition for all the font encodings except PD1.

```

```

\gmu@reserveda 3979 \ifx\relax#1\def\gmu@reserveda {\@xa\@gobble\string#2}%
3980 \else
3981 \def\gmu@reserveda {#1}%
3982 \fi
3983 \edef\gmu@reserveda {%
3984 \nx\DeclareTextCommand\nx{PD1}{\gmu@reserveda}%
3985 \gmu@reserveda
3986 \DeclareTextCommandDefault{#2}{#3}%
\pdef 3987 \pdef{#3}%
3988 } added for LATEX.

\DeclareLogo 3991 \DeclareLogo\LaTeX{%
3992 {%
3993 L%
3994 \setbox\z@\hbox{\check@mathfonts
3995 \fontsize\sf@size\z@
3996 \math@fontsfalse\selectfont
3997 A}%
3998 \kern-.57\wd\z@
3999 \sbox\tw@\T%
4000 \vbox\to\ht\tw@{\copy\z@\vss}%
4001 \kern-.2\wd\z@% originally -.15em for T.
4002 }%
4003 {%
4004 \ifdim\fontdimen1\font=\z@
4005 \else
4006 \count\z@=\fontdimen5\font
4007 \multiply\count\z@ by 64\relax
4008 \divide\count\z@ by \p@
4009 \count\tw@=\fontdimen1\font
4010 \multiply\count\tw@ by \count\z@
4011 \divide\count\tw@ by 64\relax
4012 \divide\count\tw@ by \tw@
4013 \kern-\the\count\tw@\sp\relax
4014 }%
4015 \fi}%
4016 \TeX}

\LaTeXe 4018 \DeclareLogo\LaTeXe{\m@th\if
4019 b\expandafter\car\f@series\@nil\boldmath\fi
4020 \LaTeX\kern.15em2$_{\textstyle\backslash varepsilon}$}
4022 \StoreMacro\LaTeX
4023 \StoreMacro*\LaTeX

'(L)TeX' in my opinion better describes what I work with/in than just 'LATEX'.

```

```

\LaTeXpar 4029 \DeclareLogo[(La)TeX]{\LaTeXpar}{%
4030 {%
4031 \setbox\z@\hbox{(}%
4032 \copy\z@
4033 \kern-.2\wd\z@\L%
4034 \setbox\z@\hbox{\check@mathfonts
4035 \fontsize\sf@size\z@
4036 \math@fontsfalse\selectfont
4037 A}%

```

```

4038 \kern-.57\wd\z@%
4039 \sbox\tw@{\t%
4040 \vbox\to\ht\tw@{\box\z@%
4041 \vss}%
4042 }%
4043 \kern-.07em% originally -.15em for T.
4044 {%
4045 \sbox\z@%
4046 \kern-.2\wd\z@\copy\z@%
4047 \kern-.2\wd\z@\}TeX
4048 }

```

"Here are a few definitions which can usefully be employed when documenting package files: now we can readily refer to *AmSTeX*, *BibTeX* and *SLiTTeX*, as well as the usual *TeX* and *LATeX*. There's even a *PLAIN TeX* and a *Web*."

```

4055 \gm@ifundefined{AmSTeX}
\AmSTeX 4056 {\def\AmSTeX{\leavevmode\hbox{$\mathcal{A}\kern-.2em$%
4057 \lower.376ex%
4058 \hbox{\mathcal{M}}\kern-.2em\mathcal{S}\kern-.1em-\textrm{}}}{}
\BibTeX 4059 \DeclareLogo\BibTeX{{\rmfamily\kern-.05em%
4060 \textsc{i}\kern-.025em b}\kern-.08em% the kern is wrapped in braces
4061 for my \fakescaps' sake.
4062 \textrm{}}}
\SliTeX 4063 \DeclareLogo\SliTeX{{\rmfamily\kern-.06em L\kern-.18em%
4064 \raise.32ex\hbox%
4065 {\scshape i}\kern-.03em\textrm{}}}
\PlainTeX 4066 \DeclareLogo\PlainTeX{\textsc{Plain}\kern2pt\textrm{}}
\Web 4067 \DeclareLogo\Web{\textsc{Web}\textrm{}}

```

There's also the *(L)ATEX* logo got with the *\LaTeXpar* macro provided by *gutils*. And here *The TeXbook*'s logo:

```

\TeXbook 4073 \DeclareLogo[The\TeX\book]\TeXbook{\textsl{The\TeX\book}}
4074 \let\TB\TeXbook% TUG Boat uses this.
\eTeX 4075 \DeclareLogo[e-TeX]\eTeX{%
4076 \iffontchar\font"03B5{\itshape\varepsilon}\else
4077 \ensuremath{\varepsilon}\fi-\kern-.125em\textrm{}}% definition sent by
4078 Karl Berry from TUG Boat itself.
4079 \StoreMacro\eTeX
\pdfTeX 4080 \DeclareLogo[pdfe-TeX]\pdfeTeX{pdf\eTeX}
\pdfTeX 4081 \DeclareLogo[pdfTeX]\pdfTeX{pdf\textrm{TeX}}
\pdfLaTeX 4082 \DeclareLogo[pdfLaTeX]\pdfLaTeX{pdf\LaTeX}
\XeTeX 4083 \gm@ifundefined{XeTeX}{%
4084 \DeclareLogo\XeTeX{X\kern-.125em\relax
4085 \gm@ifundefined{reflectbox}{%
4086 \lower.5ex\hbox{E}\kern-.1667em\relax}%
4087 \lower.5ex\hbox{\reflectbox{E}}\kern-.1667em\relax}%
4088 \textrm{}}
4089 \gm@ifundefined{XeLaTeX}{%
4090 \DeclareLogo\XeLaTeX{X\kern-.125em\relax
4091 \gm@ifundefined{reflectbox}{%
4092 \lower.5ex\hbox{E}\kern-.1667em\relax}%
4093 \lower.5ex\hbox{\reflectbox{E}}\kern-.1667em\relax}%
4094 \textrm{}}
4095 \gm@ifundefined{XeLaTeX}{%

```

```
\XeLaTeX 4097 \DeclareLogo{\XeLaTeX}{\kern-.125em\relax
4098 \gm@ifundefined{reflectbox}{%
4099 \lower.5ex\hbox{E}\kern-.1667em\relax}%
4100 \lower.5ex\hbox{\reflectbox{E}}\kern-.1667em\relax}%
4101 }
```

As you see, if  $\text{\TeX}$  doesn't recognise  $\text{\reflectbox}$  (graphics isn't loaded), the first E will not be reversed. This version of the command is intended for non-X $\text{\TeX}$  usage. With X $\text{\TeX}$ , you can load the  $\text{xltextra}$  package (e.g. with the gutils  $\text{\XeTeXthree}$  declaration) and then the reversed E you get as the Unicode Latin Letter Reversed E.

```
\LuaTeX 4109 \DeclareLogo[\LuaTeX]{\textsc{Lua}}\TeX
```

### Expandable turning stuff all into ‘other’

A shorthand. Note that it takes an undelimited argument not requires *(balanced text)*.

```
\detoken@xa 4128 \long\def\detoken@xa#1{\detokenize@xa{#1}}
```

The next macro originates from the ancient era when I didn't know about  $\varepsilon$ - $\text{\TeX}$ 's  $\text{\detokenize}$ . A try to redefine it to  $\text{\detokenize@xa}{#1}$  resulted in error so (vo.991) I leave it and use as is.

Note however it acts different than  $\text{\detoken@xa}$  for a macro with parameters: while  $\text{\detoken@xa}$  produces and ‘extra }’ error,  $\text{\all@other}$  expands to the detokenised meaning.

```
\all@other 4138 \long\def\all@other#1{\@xa\gm@gobmacro\meaning#1}
```

The  $\text{\gm@gobmacro}$  macro above is applied to gobble the  $\text{\meaning}$ 's beginning, long  $\text{\macro:}->$  all ‘other’ that is.

```
\gm@gobmacro 4143 \edef\gm@tempa{%
4144 \def\@nx\gm@gobmacro##1\@xa\@gobble\string\macro:##2->{}}
4145 }
```

### Brave New World of X $\text{\TeX}$

```
\@ifXeTeX 4150 \def\@ifXeTeX{%
4151 \ifdefined\XeTeXversion
4152 \unless\ifx\XeTeXversion\relax\afterfifi\@firstoftwo\else%
4153 \afterfifi\@secondoftwo\fi
4154 \else\afterfi\@secondoftwo\fi}
```

```
\XeTeXifprefix 4155 \def\XeTeXifprefix{%
4156 \@ifXeTeX{}{\unless}}
```

```
\XeTeXthree 4158 \DeclareCommand{\XeTeXthree}{\o}{%
4162 \@ifXeTeX{%
4163 \IfValueT{#1}{\PassOptionsToPackage{#1}{fontspec}}%
4164 \@ifpackageloaded{gmverb}{\StoreMacro\verb}{}}%
4165 \RequirePackage{xltextra}% since v 0.4 (2008/07/29) this package re-
 defines \verb and verbatim*, and quite elegantly provides an option
 to suppress the redefinitions, but unfortunately that option excludes also
 a nice definition of \xxt@visiblespace which I fancy.
4172 \ifpackageloaded{gmverb}{\RestoreMacro\verb}{}
4173 \AtBeginDocument{}}
```

```

adddefaultfontfeatures 4174 \RestoreMacro\LaTeX\RestoreMacro*{LaTeX}%
4175 my version of
4176 the LATEX logo has been stored just after defining, in line 4023.
4177 \RestoreMacro\eTeX}%
4178
4179 \pdef\adddefaultfontfeatures##1{%
4180 \addtomacro\zf@default@options{\#1, }%
4181 }% of \@ifXeTeX's first argument,
4182 {}% \@ifXeTeX's second argument,
4183 }% of \XeTeXthree's body.

```

The \udigits declaration causes the digits to be typeset uppercase. I provide it since by default I prefer the lowercase (nautical) digits.

```

\udigits 4189 \AtBeginDocument{%
4190 \@ifpackageloaded{fontspec}{%
4191 \pdef\udigits{%
4192 \addfontfeature{Numbers=Uppercase}}%
4193 }{%
4194 \emptyify\udigits}}

```

### Fractions

```

\Xedeckfracc 4199 \def\Xedeckfracc{\gmu@ifstar\gmu@xedekfraccstar%
4200 \gmu@xedekfraccplain}

```

(plain) The starless version turns the font feature `frac` on.

(\*) But nor my modification of Minion Pro neither TeX Gyre Pagella doesn't feature the `frac` font feature properly so, with the starred version of the declaration we use the characters from the font where available (see the \namedef{s} below) and the `numr` and `dnom` features with the fractional slash otherwise (via \gmu@dekfracc).

(\*\*) But Latin Modern Sans Serif Quotation doesn't support the numerator and denominator positions so we provide the double star version for it, which takes the char from font if it exist and typesets with lowers and kerns otherwise.

```

\gmu@xedekfraccstar 4214 \def\gmu@xedekfraccstar{%
4215 \def\gmu@xefraccdef##1##2{%
4216 \iffontchar\font\#2{%
4217 \namedef{gmu@xefracc##1}{\char##2}%
4218 }{%
4219 \n@melet{gmu@xefracc##1}{relax}%
4220 }%
4221 }%
4222 \def\gmu@dekfracc##1##2{%
4223 {\addfontfeature{VerticalPosition=Numerator}##1}%
4224 \gmu@numeratorkern
4225 \char"2044\gmu@denominatorkern
4226 {\addfontfeature{VerticalPosition=Denominator}##2}}%

```

We define the fractional macros. Since Adobe Minion Pro doesn't contain  $\frac{n}{5}$  nor  $\frac{n}{6}$ , we don't provide them here.

```

4229 \gmu@xefraccdef{1/4}{BC}%
4230 \gmu@xefraccdef{1/2}{BD}%
4231 \gmu@xefraccdef{3/4}{BE}%
4232 \gmu@xefraccdef{1/3}{2153}%
4233 \gmu@xefraccdef{2/3}{2154}%
4234 \gmu@xefraccdef{1/5}{2155}%

```

```

4235 \gmu@xefraccdef{2/5}{\text{\textfrac{2}{5}}}
4236 \gmu@xefraccdef{3/5}{\text{\textfrac{3}{5}}}
4237 \gmu@xefraccdef{4/5}{\text{\textfrac{4}{5}}}
4238 \gmu@xefraccdef{1/6}{\text{\textfrac{1}{6}}}
4239 \gmu@xefraccdef{5/6}{\text{\textfrac{5}{6}}}
4240 \gmu@xefraccdef{1/8}{\text{\textfrac{1}{8}}}
4241 \gmu@xefraccdef{3/8}{\text{\textfrac{3}{8}}}
4242 \gmu@xefraccdef{5/8}{\text{\textfrac{5}{8}}}
4243 \gmu@xefraccdef{7/8}{\text{\textfrac{7}{8}}}
4244 \pdef\dekfracc@args##1##2{%
4245 \gm@ifundefined{\gmu@xefracc}{\detokenize{##1##2}}{%
4246 \gmu@dekfracc{##1}{##2}}{%
4247 \csname\gmu@xefracc\detokenize{##1##2}\endcsname}%
4248 \if@gmu@mmhbox\egroup\fi
4249 }% of \dekfracc@args.
4250 \gm@ifstar{\let\gmu@dekfracc\gmu@dekfraccsimple}{ }%
4251 }

\gmu@xedekfraccplain
\dekfracc@args
4253 \def\gmu@xedekfraccplain{\% 'else' of the main \gm@ifstar
4254 \pdef\dekfracc@args##1##2{%
4255 \ifmmode\hbox\fi{%
4256 \addfontfeature{Fractions=On}%
4257 ##1##2}%
4258 \if@gmu@mmhbox\egroup\fi
4259 }% of \dekfracc@args
4260 }

\if@gmu@mmhbox
4262 \newif\if@gmu@mmhbox% we'll use this switch for \dekfracc and also for \thous
(hacky thousand separator).

\dekfracc
4265 \pdef\dekfracc{%
4266 \ifmmode\hbox\bgroup\gmu@mmhboxtrue\fi
4267 \dekfracc@args}

\gmu@numeratorkern
\gmu@denominatorkern
4271 \def\gmu@numeratorkern{\kern-.055em\relax}
4272 \def\gmu@denominatorkern{\kern-.05em\relax}

```

What have we just done? We defined two versions of the `\Xefrac` declaration. The starred version is intended to make use only of the built-in fractions such as  $\frac{1}{2}$  or  $\frac{7}{8}$ . To achieve that, a handful of macros is defined that expand to the Unicodes of built-in fractions and `\dekfracc` command is defined to use them.

The unstarred version makes use of the Fraction font feature and therefore is much simpler.

Note that in the first argument of `\gm@ifstar` we wrote 8 (eight) `#`s to get the correct definition and in the second argument 'only' 4. (The L<sup>A</sup>T<sub>E</sub>X 2<sub>&</sub> Source claims that that is changed in the 'new implementation' of `\gm@ifstar` so maybe it's subject to change.)

A simpler version of `\dekfracc` is provided in line [5722](#).

### `\resizographics`

```

\resizographics
4294 \pdef\resizographics##1##2##3{%
 2009/11/17 works bad with a file whose
 name contains spaces so I return \XeTeXpicfile
4299 \resizebox{##1}{##2}{%
4300 \edef\gmu@tempa{\@nx\csname\XeTeX\@nx\@ifendswithpdf{%

```

```

4301 \@xa\string\csname#3\endcsname} {pdf} {pic}file\@nx%
4302 \endcsname}%
4302 \gmu@tempa\#3"\relax} }

4304 \edef\gmu@tempa{%
4305 \def\@nx\@ifendswithpdf##1{%
4306 \unexpanded{%
4307 \ifnum
4308 \if\relax\gmu@pdfdetector}\#\#1%
4309 \detokenize{pdf}\unexpanded{\relax\else\fi}%
4309 we expand to
4309 1 if #1 ends with lowercase 'pdf' of cat. 12
4311 \unexpanded{\if\relax\gmu@PDFdetector}\#\#1%
4312 \detokenize{PDF}\unexpanded{\relax\else\fi}%
4312 we expand to
4312 1 if #1 ends with uppercase 'PDF' of cat. 12
4314 >0
4315 \unexpanded{\@xa\@firstoftwo\else\@xa\@secondoftwo\fi}%
4316 }% of \@ifendswithpdf
4320 \def\@nx\gmu@pdfdetector##1\detokenize{pdf}{}
4321 \def\@nx\gmu@PDFdetector##1\detokenize{PDF}{}
4322 }\gmu@tempa

\textsuperscript@@
4325 \pdef\textsuperscript@@##1{%
4326 \selectfont##1%
4327 \def\textsuperscript##1{%
4328 {\m@th\ensuremath{^{\mbox{\scriptsize\sf@size\z@##1}}}}}
4330 \let\textsuperscript@@\textsuperscript

\GMtextsuperscript
4332 \def\GMtextsuperscript{%
4333 \ifXeTeX{%
4334 \DeclareCommand{textsuperscript}{sm}{%
4335 \IfValueTF{##1}{\textsuperscript@@{##2}}{%
4336 {%
4337 \begingroup
4338 \addfontfeature{VerticalPosition=Numerator}\#\#2%
4339 \endgroup}%
4340 }{\truetextsuperscript}}
4342 \def\truetextsuperscript{%
4343 \let\textsuperscript\textsuperscript@@
4344 }

```

### Settings for mathematics in main font

I used these terrible macros while typesetting E. Szarzyński's *Letters* in 2008. The `\gmath` declaration introduces math-active digits and binary operators and redefines Greek letters and parentheses, the `\garamath` declaration redefines the quantifiers and is more Garamond Premier Pro-specific.

`\gmath` So, when you set default fonts (in the preamble), put `\gmath` to set what possible from them to math. This sets the `normal` math version. If you want to use another set of fonts elsewhere in your document and have according math for them, set them 'for a while' in the preamble and in their scope declare `\gmath[<version>]`. Then put `\mathversion{normal}` right after `\begin{document}` and `\gmath[<version>]` where you want those other fonts.

\gmath takes second optional argument, in parentheses, which should be (sth. that expands to) a \fontspec font selecting command. A font selected by this command will be declared and used when some char in basic font is missing and only else the default math font will be left for such a char.

So,

\gmath[<version>] (<rescue font(specification)>)

Note that \gmath without first optional argument has always to come first because otherwise it overwrite settings of your math version.

```

\gmu@getfontstring 4378 \def\gmu@getfontstring{%
4379 \xdef\gmu@fontstring{%
4380 \gmu@fontstring@}}
\gmu@fontstring@ 4382 \def\gmu@fontstring@{%
4383 \@xa\@xa\@xa\gmu@fontstring@@\@xa\meaning\the\font\@nil}
\gmu@fontstring@@ 4391 \def\gmu@fontstring@@{\#1"\#2"\#3\@nil{"\#2"}}
\gmu@getfontscale 4393 \def\gmu@getfontscale{\#1Scale\#2=\#3, {%
4394 \ifx\gmu@getfontscale\#3\else
4395 \def\gmu@tempa{MatchLowercase}%
4396 \def\gmu@tempb{\#3}%
4397 \ifx\gmu@tempa\gmu@tempb
4398 \gmu@calc@scale{5}%
4399 \@xa\@firstoftwo
4400 \else
4401 \def\gmu@tempa{MatchUppercase}%
4402 \ifx\gmu@tempa\gmu@tempb
4403 \gmu@calc@scale{8}%
4404 \afterfifi\@firstoftwo
4405 \else\afterfifi\@secondoftwo
4406 \fi
4407 \fi
4408 {\xdef\gmu@fontscalebr{[\gmu@fontscale]_}\%}
4409 {\xdef\gmu@fontscalebr{[\#3]_}\%}
4410 \afterfi\gmu@getfontscale
4411 \fi
4412 }
\gmu@getfontdata 4415 \def\gmu@getfontdata{\#1{%
4416 \global\emptyify\gmu@fontscalebr
4417 \begingroup
4418 \#1%
4419 \@xa\@xa\@xa\gmu@getfontscale
4420 \csname_zf@family@options\f@family\endcsname
4421 ,Scale=\gmu@getfontscale,%
4422 \gmu@getfontstring
4423 \xdef\gmu@theskewchar{\the\skewchar\font}%
4424 \endgroup
4425 }
\gmu@stripchar 4428 \def\gmu@stripchar{\#1" "}

\gmath@getfamnum 4430 \DeclareCommand\gmath@getfamnum{C{\gmath@fam}}{%
4431 \edef\gmath@famnum{\@xa\gmu@stripchar\meaning\#1}%
4432 }

```



```

4488 \@nx\DeclareFontShape{\encodingdefault}{gmathIt}%
4489 \PutIfValue{#1}{m}{n}{%
4490 <-> \gmu@fontscalebr \gmu@fontstring}{}%
4491 } \gmu@tempa
4491 }% of if #2 given
4493 \gmu@getfontdata{\rmfamily\upshape}%
4494 \edef\gmu@tempa{%
4495 \IfValueTF{#1}{\@nx\SetSymbolFont{gmathroman}{#1}}{%
4496 \@nx\DeclareSymbolFont{gmathroman}{}{%
4497 \encodingdefault}{gmathrm}\PutIfValue{#1}{m}{n}{%
4498 \@nx\DeclareFontFamily{\encodingdefault}{gmathrm}%
4499 \PutIfValue{#1}{%
4500 \skewchar\font\gmu@theskewchar\space}%
4501 \@nx\DeclareFontShape{\encodingdefault}{gmathrm}%
4502 \PutIfValue{#1}{m}{n}{%
4503 <-> \gmu@fontscalebr \gmu@fontstring}{}%
4502 } \IfValueT{#1}{%
4503 \@nx\SetMathAlphabet{@nx\mathrm}{#1}{%
4504 \encodingdefault}{gmathrm#1}{m}{n}}%
4504 } \gmu@tempa
4507 \xa\font\csname\gmath@font\PutIfValue{#1}\endcsname
4508 =\gmu@fontstring\relax
4509 \gmathfamshook
4510 }
4512 \empty\gmathfamshook
\gmathbase 4514 \DeclareCommand\gmathbase{oC{\NoValue}}{%
4515 \gmathFams[#1](#2)%
\gmath@do 4516 \DeclareCommand\gmath@do{%
4517 m% (1) the character or CS to be declared,
4518 o% (2) the Unicode to be assigned,
4519 m% (3) math type (CS like \mathord etc.)
4520 C{\gmath@fam}% font family
4521 }%
4523 \gmath@getfamnum(##4)%
4524 \IfValueTF{##2}{%
4525 \edef\gmu@tempa{%
4526 =\mathchar@type##3\space
4527 \gmath@famnum\space
4529 "##2\relax}%
4530 \if\relax@nx##1%
4531 \gmu@ifstored{##1}{}{\StoreMacro##1}%
4532 \edef\gmu@tempa{%
4533 \XeTeXmathchardef\@nx##1\gmu@tempa}%
4534 \else
4535 \edef\gmu@tempa{%
4536 \XeTeXmathcode`##1\gmu@tempa}
4537 \fi%
4538 }%
4539 { % no value of ##2
4540 \edef\gmu@tempa{%
4541 \XeTeXmathcode`##1=
```

```

4542 \mathchar@type##3\space
4543 \gmath@famnum\space
4544 `##1\relax}%
4545 }%
4546 \gmu@tempa
4547 }% of \gmath@do
\gmath@doif 4552 \DeclareCommand{\gmath@doif}{%
4553 mooooooooo% (1) the Unicode hex of char enquired,
4554 moooooooo% (2) the char or CS to be declared,
4555 moooooooo% (3) math type CS(\mathord etc.),
4556 ooooooooo% (4) second-choice Unicode (taken if first-choice is absent),
4557 ooooooooo% (5) third-choice Unicode (as above if second-choice is absent
4558 from font).
4559 B {\gmath@fam} % (6) never used in this package. Why?
4560 C {\NoValue}
4561 }%
4562 \gmu@storeifnotyet{##2}%
4563 \@xa\let\@xa\gmath@ft\csname
4564 gmath@font%
4565 \ifx\gmath@fam##6\else\@t\fi
4566 \gmath@version
4567 \endcsname
4568 \iffontchar\gmath@ft"##1\gmath@do##2[##1]##3(##6)%
4569 \else
4570 \IfValueTF{##4}{%
4571 \iffontchar\gmath@ft"##4\%
4572 \gmath@do##2[##4]##3(##6)%
4573 \else
4574 \IfValueTF{##5}{%
4575 \iffontchar\gmath@ft"##5\%
4576 \gmath@do##2[##5]##3(##6)%
4577 \else
4578 \gmath@restore{##1}{##2}{##3}{##4}{##5}{##7}%
4579 \fi}%
4580 \gmath@restore{##1}{##2}{##3}{##4}{##5}{##7}%
4581 }% of \gmath@doif In the command above we try to define math char or
4582 a CS in the family given as ##6. If there're no respective chars, we try
4583 the same with the family given (as a word) in ##7.
\gmath@restore 4585 \def\gmath@restore##1##2##3##4##5##6{%
4586 \IfValueT{##6}{%
4587 \ifcsname##6\endcsname
4588 \edef\gmu@tempa{%
4589 \unexpanded{\gmath@doif{##1}{##2}{%
4590 ##3}[##4][##5]}%
4591 {\@xa\@nx\csname##6\endcsname}\relax
4592 }\gmu@tempa
4593 \@xa\@gobbletwo% if family ##6 is defined, we gobble the other
4594 branch

```

```

4594 \fi
4595 }%
4596 \firstofone
4597 {\if\relax\@nx##2%
4598 \RestoreMacro##2%
4599 \fi
4600 }%
4601 }%
4603 \iffalse \lrcorner doesn't work in a non-math font.
\gmath@delc 4604 \DeclareCommand{\gmath@delc}[1]{%
 % #1 the char or CS to be declared,
 % [#2] the Unicode (if not the same as the char).
 \gmath@getfamnum
 \IfValueTF{##2}{%
 \edef\gmu@tempa{%
 =\gmath@famnum\space"##2\relax}%
 \edef\gmu@tempa{%
 \XeTeXdelcode`##1\gmu@tempa}%
 }%
 {%
 \edef\gmu@tempa{%
 \XeTeXdelcode`##1=%
 \gmath@famnum\space
 `##1\relax}%
 }%
 \gmu@tempa
}%
4627 }% of \gmath@delc
\gmath@delcif 4629 \def{\gmath@delcif##1##2}{%
 % #1 the Unicode enquired,
 % #2 the char to be delcode-declared
 \iffontchar{\gmath@font"##1\gmath@delc##2[##1]\fi}
4636 \fi}% of ifffalse
\gmath@delimif 4638 \def{\gmath@delimif##1##2##3}{%
 % #1 the Unicode enquired,
 % #2 the CS defined as \XeTeXdelimiter,
 % #3 the math type CS (probably \mathopen or \mathclose).
 \iffontchar{\gmath@font"##1
 \gmath@getfamnum
 \protected\edef{##2}{\@nx\ensuremath{%
 \XeTeXdelimiter\mathchar@type##3\space
 \gmath@famnum\space"##1\relax}}%
 }%
}%
4650 }% of \gmath@delimif.
\rmopname 4652 \pdef{\rmopname##1##2##3}{%
 \mathop{\##1\kern\z@\mathrm{##3}}\csname n##2limits@\%
 \endcsname
}%
4654 }%
\gmu@dogmathbase 4656 \DeclareCommand{\gmu@dogmathbase}{oC{\NoValue}}{%
 \RestoreMacro\mathchar@type%
 \IfValueT{##1}{\mathversion{##1}}%
 \edef{\gmath@version}{\PutIfValue{##1}}%
 @xa\let@xa\gmath@fam\csname symgmathroman%
4662
4665

```

```

4666 \endcsname
4667 \edef\gmath@famm{\symgmathRoman\gmath@version}%
4668 % as you see, this
4669 % is not a font family (number) but a macro containing the name: of the
4670 % secondary ('rescue') family.
4671 \typeout{@@@\gmathutils.sty: taking some math chars from
4672 % the font ^J \gmu@fontstring@}
4673 \gmath@do+\mathbin
4674 \gmath@doif{2212}-\mathbin[2013](\gmath@famm)% minus sign if
4675 % present or else en dash
4676 \gmath@do=\mathrel
4677 \gmath@do0\mathord
4678 \gmath@do1\mathord
4679 \gmath@do2\mathord
4680 \gmath@do3\mathord
4681 \gmath@do4\mathord
4682 \gmath@do5\mathord
4683 \gmath@do6\mathord
4684 \gmath@do7\mathord
4685 \gmath@do8\mathord
4686 \gmath@do9\mathord
4687 \gmath@doif{2264}\le\mathrel(\gmath@famm)%
4688 \let\leq\le
4689 \let\leeng\le
4690 \gmath@doif{2265}\ge\mathrel(\gmath@famm)%
4691 \let\geq\ge
4692 \let\geeng\ge
4693 \gmath@doif{2A7D}\xleq\mathrel(\gmath@famm)%
4694 \gmath@doif{2A7E}\xgeq\mathrel(\gmath@famm)%
4695 %@ifpackageloaded{polski}{%
4696 % \ifdefined\xleq
4697 % \gmu@storeifnotyet\leq
4698 % \let\leq=\xleq
4699 % \let\le=\leq
4700 % \fi
4701 % \ifdefined\xgeq
4702 % \gmu@storeifnotyet\geq
4703 % \let\geq=\xgeq
4704 % \let\ge=\geq
4705 % \fi}{}%
4706 \gmath@do.\mathpunct
4707 \gmath@do,\mathpunct
4708 \gmath@do;\mathpunct
4709 \gmath@do...\mathpunct
4710 \gmath@do(\mathopen
4711 \gmath@do)\mathclose
4712 \gmath@do[\mathopen
4713 \gmath@do]\mathclose
4714 \gmath@doif{ooD7}*\mathbin(\gmath@famm)%
4715 \gmath@do:\mathrel
4716 \gmath@doif{ooB7}.\mathbin(\gmath@famm)%
4717 \gmath@doif{22C6}*\mathbin(\gmath@famm)% low star
4718 \gmath@doif{2300}\varnothing\mathord(\gmath@famm)%

```

```

4725 \gmath@doif{221E}\infty\mathord(\gmath@famm)%
4726 \gmath@doif{2248}\approx\mathrel(\gmath@famm)%
4727 \gmath@doif{2260}\neq\mathrel(\gmath@famm)%
4728 \let\neq\neq
4729 \gmath@doif{ooAC}\neg\mathbin(\gmath@famm)%
4730 \gmath@doif{ooAC}\neq\mathord(\gmath@famm)%
4731 \gmath@do/\mathop
4732 \gmath@do<\mathrel
4733 \gmath@do>\mathrel
4734 \gmath@doif{2329}\langle\mathopen(\gmath@famm)%
4735 \gmath@doif{232A}\rangle\mathclose(\gmath@famm)%
4736 \gmath@doif{2202}\partial\mathord(\gmath@famm)%
4737 \gmath@doif{ooB1}\pm\mathbin(\gmath@famm)%
4738 \gmath@doif{007E}\sim\mathrel(\gmath@famm)%
4739 \gmath@doif{2190}\leftarrow\mathrel(\gmath@famm)%
4740 \gmath@doif{2192}\rightarrow\mathrel(\gmath@famm)%
4741 \gmath@doif{2194}\leftrightarrow\mathrel(\gmath@famm)%
4742 \gmath@doif{2194}\leftrightarrow\mathrel(\gmath@famm)%
4743 \gmath@doif{2194}\leftrightarrow\mathrel(\gmath@famm)%
4744 if not
4745 present, \gmath further will take care of it if left and right arrows are
4746 present.
4747 \gmath@doif{2191}\uparrow\mathrel(\gmath@famm)%
4748 it should be
4749 a delimiter (declared with \gmath@delimif) but in a non-math font
4750 the delimiters don't work (2008/11/19) and I don't think I'll ever need
4751 up- and down- arrows as delimiters.
4752 \gmath@doif{2193}\downarrow\mathrel(\gmath@famm)%
4753 \gmath@doif{2208}\in\mathrel[\o3F5][\o454](\gmath@famm)%

```

As a fan of modal logics I allow redefinition of `\lozenge` and `\square` iff both are in the font. I don't accept the 'ballot box' U+2610.

```

4756 \if\iffontchar\gmath@font"25CA\else\fi
4757 \iffontchar\gmath@font"25FB\else\iffontchar%
4758 \gmath@font"25A1\else\fi\fi
4759 \gmath@do\lozenge[25CA]\mathord
4760 \gmath@doif{25FB}\square\mathord[25A1](\gmath@famm)%
4761 'medium
4762 white square (modal operator)' of just 'white square'.
4763 \fi
4764 \gmath@doif{EB08}\bigcircle\mathbin(\gmath@famm)%
4765 \gmath@doif{2227}\wedge\mathbin(\gmath@famm)%
4766 \gmath@doif{2228}\vee\mathbin(\gmath@famm)%
4767 \gmath@doif{0393}\Gamma\mathalpha(\gmath@famm)%
4768 \gmath@doif{0394}\Delta\mathalpha(\gmath@famm)%
4769 \gmath@doif{0398}\Theta\mathalpha(\gmath@famm)%
4770 \gmath@doif{039B}\Lambda\mathalpha(\gmath@famm)%
4771 \gmath@doif{039E}\Xi\mathalpha(\gmath@famm)%
4772 \gmath@doif{03A3}\Sigma\mathalpha(\gmath@famm)%
4773 \gmath@doif{03A5}\Upsilon\mathalpha(\gmath@famm)%
4774 \gmath@doif{03A6}\Phi\mathalpha(\gmath@famm)%
4775 \gmath@doif{03A8}\Psi\mathalpha(\gmath@famm)%
4776 \gmath@doif{03A9}\Omega\mathalpha(\gmath@famm)%
4777 \xa\let\x\gmath@fam\csname\x\gmath@version%
4778 \endcsname
4779 \edef\gmath@famm{\symgmathItalic\gmath@version}%
4780 \gmath@doif{03B1}\alpha\mathalpha(\gmath@famm)%
4781 \gmath@doif{03B2}\beta\mathalpha(\gmath@famm)%

```

```

4784 \gmath@doif{\o3B3}{\gamma\mathalpha{(\gmath@famm)}%}
4785 \gmath@doif{\o3B4}{\delta\mathalpha{(\gmath@famm)}%}
4786 \gmath@doif{\o3F5}{\epsilon\mathalpha{(\gmath@famm)}%}
4787 \gmath@doif{\o3B5}{\varepsilon\mathalpha{(\gmath@famm)}%}
4788 \gmath@doif{\o3B6}{\zeta\mathalpha{(\gmath@famm)}%}
4789 \gmath@doif{\o3B7}{\eta\mathalpha{(\gmath@famm)}%}
4790 \gmath@doif{\o3B8}{\theta\mathalpha{(\gmath@famm)}%}
4791 \gmath@doif{\o3D1}{\vartheta\mathalpha{(\gmath@famm)}%}
4792 \gmath@doif{\o3B9}{\iota\mathalpha{(\gmath@famm)}%}
4793 \gmath@doif{\o3BA}{\kappa\mathalpha{(\gmath@famm)}%}
4794 \gmath@doif{\o3BB}{\lambda\mathalpha{(\gmath@famm)}%}
4795 \gmath@doif{\o3BC}{\mu\mathalpha{(\gmath@famm)}%}
4796 \gmath@doif{\o3BD}{\nu\mathalpha{(\gmath@famm)}%}
4797 \gmath@doif{\o3BE}{\xi\mathalpha{(\gmath@famm)}%}
4798 \gmath@doif{\o3Co}{\pi\mathalpha{(\gmath@famm)}%}
4799 \gmath@doif{\o3Ao}{\Pi\mathalpha{(\gmath@famm)}%}
4800 \gmath@doif{\o3C1}{\rho\mathalpha{(\gmath@famm)}%}
4801 \gmath@doif{\o3C3}{\sigma\mathalpha{(\gmath@famm)}%}
4802 \gmath@doif{\o3DA}{\varsigma\mathalpha{(\gmath@famm)}% o3C2?}
4803 \gmath@doif{\o3C4}{\tau\mathalpha{(\gmath@famm)}%}
4804 \gmath@doif{\o3C5}{\upsilon\mathalpha{(\gmath@famm)}%}
4805 \gmath@doif{\o3D5}{\phi\mathalpha{(\gmath@famm)}%}
4806 \gmath@doif{\o3C7}{\chi\mathalpha{(\gmath@famm)}%}
4807 \gmath@doif{\o3C8}{\psi\mathalpha{(\gmath@famm)}%}
4808 \gmath@doif{\o3C9}{\omega\mathalpha{(\gmath@famm)}%}
4809 \if\u1\u1%
4810 \iffontchar\gmath@font"221A
4811 \fontdimen61\gmath@font=1pt
4812 \edef\sqrtsign{%
4813 \XeTeXradical\@xa\gmu@stripchar\meaning%
4814 \symgmathroman\space"221A\relax}%
4815 \fi
4816 \fi% of if 1 1.
\max 4817 \def\max{\rmopname\relax\m{max}}%
\min 4818 \def\min{\rmopname\relax\m{min}}%
\lim 4819 \def\lim{\rmopname\relax\m{lim}}%
\sin 4820 \def\sin{\rmopname\relax\o{sin}}%
\cos 4821 \def\cos{\rmopname\relax\o{cos}}%
\tg 4822 \def\tg{\rmopname\relax\o{tg}}%
\ctg 4823 \def\ctg{\rmopname\relax\o{ctg}}%
\tan 4824 \def\tan{\rmopname\relax\o{tan}}%
\ctan 4825 \def\ctan{\rmopname\relax\o{ctan}}%
4826 }% of \gmu@dogmathbase
4827 \AtBeginDocument{\gmu@dogmathbase[\#1](\#2)%
4828 \let\gmathbase\gmu@dogmathbase
4829 }% of atbd
4830 \not@onlypreamble\gmathbase
4831 }% of \gmathbase

```

The `\gmatbase` declaration defines a couple of gmath defining commands and then launches them for the default font at begin document and becomes only that launching.

```
4837 \onlypreamble\gmathbase
```

It's a bit tricky: if `\gmathbase` occurs first time in a document inside `document` then an error error is raised. But if `\gmathbase` occurs first time in the preamble, then it removes itself from the only-preamble list and redefines itself to be only the inner macro of the former itself.

```

\gmathfurther 4844 \pdef\gmathfurther{%
4851 \def\do##1##2##3{\gmu@storeifnotyet##1%
4852 \def##1{%
4853 \mathchoice{\hbox{%
4854 \rm
4855 \edef\gma@tempa{\the\fontdimen8\font}%
4856 \larger[3]%
4857 \lower\dimexpr(\fontdimen8\font-\gma@tempa)/2\relax%
4858 \hbox{##2}}}{\hbox{%
4859 \rm
4860 \edef\gma@tempa{\the\fontdimen8\font}%
4861 \larger[2]%
4862 \lower\dimexpr(\fontdimen8\font-%
4863 \gma@tempa)/2\relax%
4864 \hbox{##2}}}{%
4865 {\mathrm{##2}}{\mathrm{##2}}##3}}%
4866 \iffontchar\gmath@font"2211\do\sum{\char"2211}{}%
4867 \fi%
4868 \do\forall{\gma@quantifierhook\rotatebox[origin=c]{180}{A}}%
4869 \gmu@forallkerning
4870 }{\nolimits}%
4871 \def\gmu@forallkerning{\setbox0=\hbox{A}\setbox2=\hbox{%
4872 \scriptsize x}%
4873 \kern\dimexpr\ht2/3*2\relax-\wd0/2\relax} to be able to redefine it
4874 when the big quantifier is Bauhaus-like.
4875 \do\exists{\rotatebox[origin=c]{180}{\gma@quantifierhook E}}{\nolimits}%
4876 \def\do##1##2##3{\gmu@storeifnotyet##1%
4877 \def##1##3{%
4878 \mathchoice{\hbox{\rm##2}}{\hbox{\rm##2}}%
4879 {\hbox{\rm\scriptsize##2}}{\hbox{\rm\tiny##2}}}%
4880 \unless\iffontchar\gmath@font"2227
4881 \do\vee{\rotatebox[origin=c]{90}{<}}{\mathbin{}}%
4882 \fi
4883 \unless\iffontchar\gmath@font"2228
4884 \do\wedge{\rotatebox[origin=c]{-90}{<}}{\mathbin{}}%
4885 \fi
4886 \unless\iffontchar\gmath@font"2194
4887 \if\iffontchar\gmath@font"2190\else1\fi
4888 \iffontchar\gmath@font"2192\else2\fi
4889 \do\leftrightarrow{\char"2190\kern-0.1em}{%
4890 \char"2192}\mathrel{%
4891 }\fi\fi
4892 \def\do##1##2##3{\gmu@storeifnotyet##1%
4893 \def##1##3{%
4894 \hbox{%
4895 \rm
4896 }}%
4897 \unless\iffontchar\gmath@font"2228
4898 \do\wedge{\rotatebox[origin=c]{-90}{<}}{\mathbin{}}%
4899 \fi
4900 \unless\iffontchar\gmath@font"2227
4901 \do\vee{\rotatebox[origin=c]{90}{<}}{\mathbin{}}%
4902 \fi
4903 }
```

```

4896 \setboxo=\hbox{####1}%
4897 \edef\gma@tempa{\the\hto}%
4898 \edef\gma@tempb{\the\dp0}%
4899 %%3%
4900 \setboxo=\hbox{####1}%
4901 \lower\dimexpr(\hto+\dp0)/2-\dp0-((%
4902 \gma@tempa+\gma@tempb)/2-\gma@tempb)%
4903 \boxo}}}}%
4904 \do\bigr\mathopen\larger
4905 \do\bigr\mathclose\larger
4906 \do\Bigl\mathopen\largerr
4907 \do\Bigr\mathclose\largerr
4908 \do\biggl\mathopen{\larger[3]}%
4909 \do\biggr\mathclose{\larger[3]}%
4910 \do\Biggl\mathopen{\larger[4]}%
4911 \do\Biggr\mathclose{\larger[4]}%
4912 \addtotoks\everymath{%
4913 \def\do##1##2{\gmu@storeifnotyet##1%
4914 \def##1{\ifmmode##2{\mathchoice
4915 {\hbox{\rm\char`##1}}{\hbox{\rm\char`##1}}%
4916 {\hbox{\rm\scriptsize\char`##1}}{\hbox{\rm\tiny%
4917 \char`##1}}}}%
4918 \else\char`##1\fi}}%
4919 \do\{\mathopen
4920 \do\}\mathclose
4921 \def\={\mathbin{=}}%
4922 \def\neqb{\mathbin{\neq}}%
4923 \let\neb\neqb
4924 \def\do##1{\gmu@storeifnotyet##1%
4925 \edef\gma@tempa{%
4926 \def\@xa\@nx\csname\@xa\gobble\string##1r%
4927 \endcsname{%
4928 \@nx\mathrel{\@nx##1}}}}%
4929 \gma@tempa}%
4930 \do\vee\do\wedge\do\neg
4931 \def\fakern{\mkern-3mu}%
4932 \thickmuskip=8mu plus 4mu\relax
4933 \gma@gmathhook
4934 }% of \everymath.
4935 \everydisplay\everymath
4936 \ifdefined\Url
4937 \ampulexdef\Url{\let\do\@makeother
4938 \everymath{}\let\do\@makeother}%
4939 I don't know why but the url
4940 package's \url typesets the argument inside a math which caused
4941 digits not to be typewriter but Roman and lowercase.
4942 \fi% of \ifdefined\Url.
4943 }% of \def\gmathfurther.
4944 \StoreMacro\mathchar@type
4945 \DeclareCommand\gmath{oC{\NoValue}}{%
4946 \gmathbase[#1](#2)%
4947 \gmathfurther

```

```

4954 \IfValueT{#1}{\csname gmathhook#1\endcsname}%
 this allows adding
 version-specific stuff (I first used this for Fell fonts rescued with Garamond
 Premier)
4957 }

\gmathscripts 4959 \pdef\gmathscripts{%
4960 \addtotoks\everymath{\catcode`^=\relax\catcode`_=8%
4961 \relax}%
4962 \everydisplay\everymath}

\gmathcats 4963 \pdef\gmathcats{%
4964 \addtotoks\everymath{\gmu@septify}%
4965 \everydisplay\everymath}

\quantifierhook 4967 \emptify\gma@quantifierhook
4968 \def\quantifierhook#1{%
4969 \def\gma@quantifierhook{#1}%

\gma@quantifierhook 4971 \emptify\gma@gmathhook
4972 \def\gmathhook#1{\addtomacro\gma@gmathhook{#1}}%

\gma@dollar 4975 \def\gma@dollar$#1${\gmath$#1$}%
\gma@bare 4976 \def\gma@bare#1{\gma@dollar$#1$}%
\gma@checkbracket 4977 \def\gma@checkbracket{@ifnextchar[%
4978 \gma@bracket\gma@bare}%
\gma@bracket 4979 \def\gma@bracket[#1]{{\gmath[#1]}\@ifnextchar\par{}{%
4980 \noindent}}%
\gma 4980 \def\gma{@ifnextchar$%
4981 \gma@dollar\gma@checkbracket}%

\garamath 4987 \DeclareCommand\garamath{%
4988 O{\rm}%
4989 }{%

```

Before 2009/10/19 all the stuff was added to `\everymath` which didn't work.

```

\gma@arrowdash 4992 \quantifierhook{\addfontfeature{OpticalSize=800}}%
4994 \def\gma@arrowdash{%
4995 \setboxo=\hbox{\char"2192}\copyo\kern-0.6\wd0
4996 \bgcolor\rule[-\dpo]{0,6\wd0}{\dimexpr1.07\ht0+\dpo}%
 \kern-0.6\wd0}%
\gma@gmathhook 4998 \def\gma@gmathhook{%
4999 \def\do####1#####2#####3{\gmu@storeifnotyet####1%
5000 \def####1{####3{%
5001 \mathchoice{\hbox{####1####2}}{\hbox{####1####2}}%
5002 {\hbox{####1\scriptsize####2}}{\hbox{####1%
 \tiny####2}}}}}%
5003 \do\mapsto{\rule{0,4ex}{0,1ex}\kern-0.05em}%
5004 \gma@arrowdash\kern-0.05em\char"2192\mathrel
5005 \do\cup{\scshape u}\mathbin
5006 \do\varnothing{\setboxo=\hbox{\gma@quantifierhook%
 \addfontfeature{Scale=1.272727}0}%
5007 \setbox2=\hbox{\char"2044}%
5008 \copyo\kern-0.5\wd0\kern-0.5\wd2\lower0.125\wd0\%
 \copy2
5009 \kerno,5\wd0\kern-0.5\wd2}{}% of \varnothing

```

```

5010 \do\leftarrow{\char"2190\kern-0,.05em\gma@arrowdash}%
 \mathrel
5011 \do\shortleftarrow{\char"2190}\mathrel
5012 \do\rightarrow{\gma@arrowdash\kern-0,.05em\char"2192}%
 \mathrel
5013 \do\shortrightarrow{\char"2192\relax}\mathrel
5014 \do\in{\gma@quantifierhook\char"0454}\mathbin
5015 \do\prec{\gma@quantifierhook
 \rotatebox[origin=c]{-90}{\bf\scriptsize\texttt{glyphname{u03A5.a}}}}\mathrel% added 2009/9/11
5016 }% of \gma@gmathhook
5017 }% of \garamath.
5018 }% of \garamath.

```

### Minion and Garamond Premier kerning and ligature fixes

»Ws« shall not make long »s« because long »s« looks ugly next to »W«.

```

5028 \def\gmu@tempa{\kern-0,.08em\penalty10000\hskiposp\relax
5029 s\penalty10000\hskiposp\relax}
5031 \protected\edef\V{\gmu@tempa}
5033 \protected\edef\W{\gmu@tempa}
\Wz 5035 \pdef\Wz{\W\kern-0,.05em\penalty10000\hskiposp\relax_z}

```

### A left-slanted font

Or rather a left Italic *and* left slanted font. In both cases we sample the skewness of the itshape font of the current family, we reverse it and apply to \itshape in \litshape and \textlit and to \sl in \lsl. Note a slight asymmetry: \litshape and \textlit take the current family while \lsl and \textlsl the basic Roman family and basic (serif) Italic font. Therefore we introduce the \lit declaration for symmetry, that declaration left-slants \it.

I introduced them first while typesetting E. Szarzyński's *Letters* to follow his (elaborate) hand-writing and now I copy them here when need left Italic for his *Albert Camus' The Plague* to avoid using bold font.

Of course it's rather esoteric so I wrap all that in a declaration.

```

\leftslanting@%
\litdimen 5057 \pdef\leftslanting@{%
 \def\litdimen{\strip@pt\fontdimen1\font_ex}%
\litcorrection 5058 \def\litcorrection{%
 \ifhmode\null\nobreak\hskip\litdimen\relax\fi}%
5059 \def\litkern{%
 note it's to be used inside the left slanted font, unlike \lit%
 correction, intended to be used before switching to left slant/italic.
\litkern 5060 \leavevmode\null
 \kern-\litdimen\relax}%
5061 \def\dilitkern{\kern\litdimen\litkern}%
\dilitkern 5062 \pdef\litshape{%
 \litcorrection
 \itshape
 \tempdima=-2\fontdimen1\font
 \advance\leftskip_by\strip@pt\fontdimen1\font_ex% to assure
 at least the lowercase letters not to overshoot to the (left) margin. Note
 this has any effect only if there is a \par in the scope.
5063
5064
5065
5066
5067
5068
5069
5070
5071
5072
5073
5074
5075
5076
5077
5078
5079
5080
5081
5082
5083
5084
5085
5086
5087
5088
5089
5090
5091
5092
5093
5094
5095
5096
5097
5098
5099
5100
5101
5102
5103
5104
5105
5106
5107
5108
5109
5110
5111
5112
5113
5114
5115
5116
5117
5118
5119
5120
5121
5122
5123
5124
5125
5126
5127
5128
5129
5130
5131
5132
5133
5134
5135
5136
5137
5138
5139
5140
5141
5142
5143
5144
5145
5146
5147
5148
5149
5150
5151
5152
5153
5154
5155
5156
5157
5158
5159
5160
5161
5162
5163
5164
5165
5166
5167
5168
5169
5170
5171
5172
5173
5174
5175
5176
5177
5178
5179
5180
5181
5182
5183
5184
5185
5186
5187
5188
5189
5190
5191
5192
5193
5194
5195
5196
5197
5198
5199
5199
5200
5201
5202
5203
5204
5205
5206
5207
5208
5209
5210
5211
5212
5213
5214
5215
5216
5217
5218
5219
5220
5221
5222
5223
5224
5225
5226
5227
5228
5229
5230
5231
5232
5233
5234
5235
5236
5237
5238
5239
5240
5241
5242
5243
5244
5245
5246
5247
5248
5249
5250
5251
5252
5253
5254
5255
5256
5257
5258
5259
5259
5260
5261
5262
5263
5264
5265
5266
5267
5268
5269
5270
5271
5272
5273
5274
5275
5276
5277
5278
5279
5280
5281
5282
5283
5284
5285
5286
5287
5288
5289
5289
5290
5291
5292
5293
5294
5295
5296
5297
5298
5299
5299
5300
5301
5302
5303
5304
5305
5306
5307
5308
5309
5309
5310
5311
5312
5313
5314
5315
5316
5317
5318
5319
5319
5320
5321
5322
5323
5324
5325
5326
5327
5328
5329
5329
5330
5331
5332
5333
5334
5335
5336
5337
5338
5339
5339
5340
5341
5342
5343
5344
5345
5346
5347
5348
5349
5349
5350
5351
5352
5353
5354
5355
5356
5357
5358
5359
5359
5360
5361
5362
5363
5364
5365
5366
5367
5368
5369
5369
5370
5371
5372
5373
5374
5375
5376
5377
5378
5379
5379
5380
5381
5382
5383
5384
5385
5386
5387
5388
5389
5389
5390
5391
5392
5393
5394
5395
5396
5397
5398
5399
5399
5400
5401
5402
5403
5404
5405
5406
5407
5408
5409
5409
5410
5411
5412
5413
5414
5415
5416
5417
5418
5419
5419
5420
5421
5422
5423
5424
5425
5426
5427
5428
5429
5429
5430
5431
5432
5433
5434
5435
5436
5437
5438
5439
5439
5440
5441
5442
5443
5444
5445
5446
5447
5448
5449
5449
5450
5451
5452
5453
5454
5455
5456
5457
5458
5459
5459
5460
5461
5462
5463
5464
5465
5466
5467
5468
5469
5469
5470
5471
5472
5473
5474
5475
5476
5477
5478
5479
5479
5480
5481
5482
5483
5484
5485
5486
5487
5488
5489
5489
5490
5491
5492
5493
5494
5495
5496
5497
5498
5499
5499
5500
5501
5502
5503
5504
5505
5506
5507
5508
5509
5509
5510
5511
5512
5513
5514
5515
5516
5517
5518
5519
5519
5520
5521
5522
5523
5524
5525
5526
5527
5528
5529
5529
5530
5531
5532
5533
5534
5535
5536
5537
5538
5539
5539
5540
5541
5542
5543
5544
5545
5546
5547
5548
5549
5549
5550
5551
5552
5553
5554
5555
5556
5557
5558
5559
5559
5560
5561
5562
5563
5564
5565
5566
5567
5568
5569
5569
5570
5571
5572
5573
5574
5575
5576
5577
5578
5579
5579
5580
5581
5582
5583
5584
5585
5586
5587
5588
5589
5589
5590
5591
5592
5593
5594
5595
5596
5597
5598
5599
5599
5600
5601
5602
5603
5604
5605
5606
5607
5608
5609
5609
5610
5611
5612
5613
5614
5615
5616
5617
5618
5619
5619
5620
5621
5622
5623
5624
5625
5626
5627
5628
5629
5629
5630
5631
5632
5633
5634
5635
5636
5637
5638
5639
5639
5640
5641
5642
5643
5644
5645
5646
5647
5648
5649
5649
5650
5651
5652
5653
5654
5655
5656
5657
5658
5659
5659
5660
5661
5662
5663
5664
5665
5666
5667
5668
5669
5669
5670
5671
5672
5673
5674
5675
5676
5677
5678
5679
5679
5680
5681
5682
5683
5684
5685
5686
5687
5688
5689
5689
5690
5691
5692
5693
5694
5695
5696
5697
5698
5698
5699
5699
5700
5701
5702
5703
5704
5705
5706
5707
5708
5709
5709
5710
5711
5712
5713
5714
5715
5716
5717
5718
5719
5719
5720
5721
5722
5723
5724
5725
5726
5727
5728
5729
5729
5730
5731
5732
5733
5734
5735
5736
5737
5738
5739
5739
5740
5741
5742
5743
5744
5745
5746
5747
5748
5749
5749
5750
5751
5752
5753
5754
5755
5756
5757
5758
5759
5759
5760
5761
5762
5763
5764
5765
5766
5767
5768
5769
5769
5770
5771
5772
5773
5774
5775
5776
5777
5778
5779
5779
5780
5781
5782
5783
5784
5785
5786
5787
5788
5789
5789
5790
5791
5792
5793
5794
5795
5796
5797
5798
5798
5799
5799
5800
5801
5802
5803
5804
5805
5806
5807
5808
5809
5809
5810
5811
5812
5813
5814
5815
5816
5817
5818
5819
5819
5820
5821
5822
5823
5824
5825
5826
5827
5828
5829
5829
5830
5831
5832
5833
5834
5835
5836
5837
5838
5839
5839
5840
5841
5842
5843
5844
5845
5846
5847
5848
5849
5849
5850
5851
5852
5853
5854
5855
5856
5857
5858
5859
5859
5860
5861
5862
5863
5864
5865
5866
5867
5868
5869
5869
5870
5871
5872
5873
5874
5875
5876
5877
5878
5879
5879
5880
5881
5882
5883
5884
5885
5886
5887
5888
5889
5889
5890
5891
5892
5893
5894
5895
5896
5897
5898
5898
5899
5899
5900
5901
5902
5903
5904
5905
5906
5907
5908
5909
5909
5910
5911
5912
5913
5914
5915
5916
5917
5918
5919
5919
5920
5921
5922
5923
5924
5925
5926
5927
5928
5929
5929
5930
5931
5932
5933
5934
5935
5936
5937
5938
5939
5939
5940
5941
5942
5943
5944
5945
5946
5947
5948
5949
5949
5950
5951
5952
5953
5954
5955
5956
5957
5958
5959
5959
5960
5961
5962
5963
5964
5965
5966
5967
5968
5969
5969
5970
5971
5972
5973
5974
5975
5976
5977
5978
5979
5979
5980
5981
5982
5983
5984
5985
5986
5987
5988
5989
5989
5990
5991
5992
5993
5994
5995
5996
5997
5998
5998
5999
5999
6000
6001
6002
6003
6004
6005
6006
6007
6008
6009
6009
6010
6011
6012
6013
6014
6015
6016
6017
6018
6019
6019
6020
6021
6022
6023
6024
6025
6026
6027
6028
6029
6029
6030
6031
6032
6033
6034
6035
6036
6037
6038
6039
6039
6040
6041
6042
6043
6044
6045
6046
6047
6048
6049
6049
6050
6051
6052
6053
6054
6055
6056
6057
6058
6059
6059
6060
6061
6062
6063
6064
6065
6066
6067
6068
6069
6069
6070
6071
6072
6073
6074
6075
6076
6077
6078
6079
6079
6080
6081
6082
6083
6084
6085
6086
6087
6088
6089
6089
6090
6091
6092
6093
6094
6095
6096
6097
6098
6098
6099
6099
6100
6101
6102
6103
6104
6105
6106
6107
6108
6109
6109
6110
6111
6112
6113
6114
6115
6116
6117
6118
6119
6119
6120
6121
6122
6123
6124
6125
6126
6127
6128
6129
6129
6130
6131
6132
6133
6134
6135
6136
6137
6138
6139
6139
6140
6141
6142
6143
6144
6145
6146
6147
6148
6149
6149
6150
6151
6152
6153
6154
6155
6156
6157
6158
6159
6159
6160
6161
6162
6163
6164
6165
6166
6167
6168
6169
6169
6170
6171
6172
6173
6174
6175
6176
6177
6178
6179
6179
6180
6181
6182
6183
6184
6185
6186
6187
6188
6189
6189
6190
6191
6192
6193
6194
6195
6196
6197
6198
6198
6199
6199
6200
6201
6202
6203
6204
6205
6206
6207
6208
6209
6209
6210
6211
6212
6213
6214
6215
6216
6217
6218
6219
6219
6220
6221
6222
6223
6224
6225
6226
6227
6228
6229
6229
6230
6231
6232
6233
6234
6235
6236
6237
6238
6239
6239
6240
6241
6242
6243
6244
6245
6246
6247
6248
6249
6249
6250
6251
6252
6253
6254
6255
6256
6257
6258
6259
6259
6260
6261
6262
6263
6264
6265
6266
6267
6268
6269
6269
6270
6271
6272
6273
6274
6275
6276
6277
6278
6279
6279
6280
6281
6282
6283
6284
6285
6286
6287
6288
6289
6289
6290
6291
6292
6293
6294
6295
6296
6297
6298
6298
6299
6299
6300
6301
6302
6303
6304
6305
6306
6307
6308
6309
6309
6310
6311
6312
6313
6314
6315
6316
6317
6318
6319
6319
6320
6321
6322
6323
6324
6325
6326
6327
6328
6329
6329
6330
6331
6332
6333
6334
6335
6336
6337
6338
6339
6339
6340
6341
6342
6343
6344
6345
6346
6347
6348
6349
6349
6350
6351
6352
6353
6354
6355
6356
6357
6358
6359
6359
6360
6361
6362
6363
6364
6365
6366
6367
6368
6369
6369
6370
6371
6372
6373
6374
6375
6376
6377
6378
6379
6379
6380
6381
6382
6383
6384
6385
6386
6387
6388
6389
6389
6390
6391
6392
6393
6394
6395
6396
6397
6398
6398
6399
6399
6400
6401
6402
6403
6404
6405
6406
6407
6408
6409
6409
6410
6411
6412
6413
6414
6415
6416
6417
6418
6419
6419
6420
6421
6422
6423
6424
6425
6426
6427
6428
6429
6429
6430
6431
6432
6433
6434
6435
6436
6437
6438
6439
6439
6440
6441
6442
6443
6444
6445
6446
6447
6448
6449
6449
6450
6451
6452
6453
6454
6455
6456
6457
6458
6459
6459
6460
6461
6462
6463
6464
6465
6466
6467
6468
6469
6469
6470
6471
6472
6473
6474
6475
6476
6477
6478
6479
6479
6480
6481
6482
6483
6484
6485
6486
6487
6488
6489
6489
6490
6491
6492
6493
6494
6495
6496
6497
6498
6498
6499
6499
6500
6501
6502
6503
6504
6505
6506
6507
6508
6509
6509
6510
6511
6512
6513
6514
6515
6516
6517
6518
6519
6519
6520
6521
6522
6523
6524
6525
6526
6527
6528
6529
6529
6530
6531
6532
6533
6534
6535
6536
6537
6538
6539
6539
6540
6541
6542
6543
6544
6545
6546
6547
6548
6549
6549
6550
6551
6552
6553
6554
6555
6556
6557
6558
6559
6559
6560
6561
6562
6563
6564
6565
6566
6567
6568
6569
6569
6570
6571
6572
6573
6574
6575
6576
6577
6578
6579
6579
6580
6581
6582
6583
6584
6585
6586
6587
6588
6589
6589
6590
6591
6592
6593
6594
6595
6596
6597
6598
6598
6599
6599
6600
6601
6602
6603
6604
6605
6606
6607
6608
6609
6609
6610
6611
6612
6613
6614
6615
6616
6617
6618
6619
6619
6620
6621
6622
6623
6624
6625
6626
6627
6628
6629
6629
6630
6631
6632
6633
6634
6635
6636
6637
6638
6639
6639
6640
6641
6642
6643
6644
6645
6646
6647
6648
6649
6649
6650
6651
6652
6653
6654
6655
6656
6657
6658
6659
6659
6660
6661
6662
6663
6664
6665
6666
6667
6668
6669
6669
6670
6671
6672
6673
6674
6675
6676
6677
6678
6679
6679
6680
6681
6682
6683
6684
6685
6686
6687
6688
6689
6689
6690
6691
6692
6693
6694
6695
6696
6697
6698
6698
6699
6699
6700
6701
6702
6703
6704
6705
6706
6707
6708
6709
6709
6710
6711
6712
6713
6714
6715
6716
6717
6718
6719
6719
6720
6721
6722
6723
6724
6725
6726
6727
6728
6729
6729
6730
6731
6732
6733
6734
6735
6736
6737
6738
6739
6739
6740
6741
6742
6743
6744
6745
6746
6747
6748
6749
6749
6750
6751
6752
6753
6754
6755
6756
6757
6758
6759
6759
6760
6761
6762
6763
6764
6765
6766
6767
6768
6769
6769
6770
6771
6772
6773
6774
6775
6776
6777
6778
6779
6779
6780
6781
6782
6783
6784
6785
6786
6787
6788
6789
6789
6790
6791
6792
6793
6794
6795
6796
6797
6798
6798
6799
6799
6800
6801
6802
6803
6804
6805
6806
6807
6808
6809
6809
6810
6811
6812
6813
6814
6815
6816
6817
6818
6819
6819
6820
6821
6822
6823
6824
6825
6826
6827
6828
6829
6829
6830
6831
6832
6833
6834
6835
6836
6837
6838
6839
6839
6840
6841
6842
6843
6844
6845
6846
6847
6848
6849
6849
6850
6851
6852
6853
6854
6855
6856
6857
6858
6859
6859
6860
6861
6862
6863
6864
6865
6866
6867
6868
6869
6869
6870
6871
6872
6873
6874
6875
6876
6877
6878
6879
6879
6880
6881
68
```

```

5077 \litcorrection
5078 \edef\gmu@tempa{%
5079 \@nx\addfontfeature{FakeSlant=\strip@pt\@tempdima}%
 when not \edefed, it caused an error, which is perfectly
 understandable.
5082 \gmu@tempa}%
5083 \pdef\textlit##1{%
5084 {\litshape##1}}%
5085 \pdef\lit{\rm\litshape}%
5086 \pdef\lsl{%
5087 \litcorrection
5088 \it
5089 \rm\@tempdima=\fontdimen1\font
5090 \litcorrection
5091 \xdef\gmu@tempa{%
5092 \@nx\addfontfeature{RawFeature={slant=\strip@pt%
5093 \@tempdima}}}%
5094 \rm\% Note in this declaration we left-slant the basic Roman font not the
5095 itshape of the current family.
5096 \gmu@tempa}%

```

Now we can redefine \em and \emph to use left Italic for nested emphasis. In Polish typesetting there is bold in nested emphasis as I have heard but we don't like bold since it perturbs homogeneous greyness of a page. So we introduce a three-cycle instead of two:- Italic, left Italic, upright.

```

\em 5110 \pdef\em{%
5111 \ifdim\fontdimen1\font=\z@\itshape
5112 \else
5113 \ifdim\fontdimen1\font>\z@\itshape
5114 \else\upshape
5115 \fi
5116 \fi}%
5117 \pdef\emph##1{%
5118 {\em##1}}%
5119 }% of \leftslanting@.
\leftslanting 5123 \pdef\leftslanting{\AtBeginDocument{\leftslanting@}}
5124 \AtBeginDocument{\let\leftslanting\leftslanting@}

```

### Fake Old-style Numbers

While preparing documentation of this package I faced an æsthetic problem of lack of old-style numbers in a font I fancy. The font is for the sans serif and the digits occur only in the date in title so it would be a pity not too use a nice font when only one or two numbers are needed.

```

\romorzero 5136 \def\romorzero#1{%
5137 \ifnum#1=0\zero\else\romannumeral#1\fi}
\fakeonum 5139 \DeclareCommand\fakeonum{%
5140 \o\% fake bold for the digit »2« (for which emboldening improves look),
5141 >Pm\% the text to fake old-style numbers in.
5142 }% I tried to use this command as a declaration but active digits are very uncom-
 fortable, e.g. you can't define macros with arguments.

```

```

5152 \gmu@if@onum{\#2} {%
5153 \begingroup
5154 \edef\gmu@tempa{\#2}%
5155 \makeatletter%
5156 \IfValueT{\#1}{%
5157 \prependtomacro\fake@onum@ii{%
5158 \begingroup\addfontfeature{FakeBold=\#1}}%
5159 \addtomacro\fake@onum@ii\endgroup
5160 }%
5161 \endlinechar\m@ne% to suppress the line end added by \scantokens,
5162 especially in active ^M's scopes.
5163 \gmu@dofakeonum
5164 \xa\scantokens\@xa{\gmu@tempa}%
5165 \endgroup
5166 }% of \gmu@ifonum 'else'.
5167 }% of \fakeonum.

\gmu@dofakeonum 5169 \def\gmu@dofakeonum{%
5170 \def\do##1{%
5171 \catcode`##1\active
5172 \scantokens{%
5173 \xa\let\@xa##1%
5174 \csname\fake@onum@\@xa\romorzero\string##1\endcsname%
5175 \empty}%
5176 }%
5177 \def\do##1##2{%
5178 \namedef{fake@onum@\romorzero##1}{##2}%
5179 }
5180 \def\gmu@tempa##1{%
5181 \do##1{\leavevmode
5182 \gmu@calculateslant{\#1}% uses \gmu@tempa and \gmu@tempb, therefore goes first. And defines \gmu@tempd.
5183 \gmu@measurewd{\#1}% the width of char #1 is in \gmu@tempa without kerning and in \gmu@tempb with kerning.
5184 \edef\gmu@tempc{\the\fontcharht\font`\#1}%
5185 \hbox{to\gmu@tempb{%
5186 \hss\resizebox{\gmu@tempa}{%
5187 \dimexpr\fontdimen5\font+\gmu@tempc-\fontdimen8\font}%
5188 {\gmu@tempd\#1}\hss}}}
5189 }
5190 \def\gmu@measurewd{\#1}{%
5191 \edef\gmu@tempa{\the\fontcharwd\font`\#1}%
5192 \settowidth{\@tempdimb}{% to preserve kerning
5193 \char`\#1\char`\#1\char`\#1\char`\#1\char`\#1\char`\#1%
5194 \char`\#1\char`\#1\char`\#1\char`\#1\char`\#1\char`\#1%
5195 \char`\#1\char`\#1\char`\#1\char`\#1\char`\#1\char`\#1%
5196 \char`\#1\char`\#1\char`\#1\char`\#1\char`\#1\char`\#1%
5197 }%
5198 \edef\gmu@tempb{\the\dimexpr(\@tempdimb-\gmu@tempa)/20}%
5199 }
5200 \gmu@tempao% \fake@onum@zero
5201 \gmu@tempa1% \fake@onum@i

```

```

5209 \gmu@tempa2 \% \fake@onum@ii
5210 \def\gmu@tempa#1{%
5211 \do#1{\leavevmode
5212 \gmu@measurewd{#1}%
5213 \lower
5214 \dimexpr\fontdimen8\font-\fontdimen5\font\relax
5215 \hbox{to }\gmu@tempb{\hss#1\hss}}%
5216 }
5217 }

5218 \gmu@tempa3 \% \fake@onum@iii
5219 \gmu@tempa4 \% \fake@onum@iv
5220 \gmu@tempa5 \% \fake@onum@v
5221 \gmu@tempa7 \% \fake@onum@vii
5222 \gmu@tempa9 \% \fake@onum@ix

5223 \def\gmu@tempa#1{%
 to preserve pseudo-kerning in digits sequences.
5224 \do#1{\leavevmode
5225 \gmu@measurewd{#1}%
5226 \hbox{to }\gmu@tempb{\hss#1\hss}}}

5227 \gmu@tempa6 \% \fake@onum@vi
5228 \gmu@tempa8 \% \fake@onum@viii

\gmu@if@onum 5229 \protected\def\gmu@if@onum{%
5230 \edef\gmu@tempa{\@xa\meaning\the\font}%
5231 \@xa\ifinmeaning\detokenize{+onum}\of\gmu@tempa
5232 }
5233 }

Thus \gmu@if@onum becomes a two-argument command that executes its #1 if there is +onum in current font specification or its #2 if +onum is absent.

One could easily generalise \gmu@if@onum to \@if@fontfeature, i.e. to a test for an arbitrary font feature, probably with employing that very nice feature specification of fontspec, so that you could write \IfFontFeature {Numbers=OldStyle} {} {\fake old-style digits}.

\gmu@getslant 5234 \pdef\gmu@getslant{%
 we define \gmu@tempa to the (fake) slant of current
 font.
5235 \edef\gmu@tempa{\@xa\meaning\the\font\detokenize{%
 slant=0,}}%
5236 \edef\gmu@tempb{%
5237 \def\@nx\gmu@tempb##\#1%
5238 \detokenize{slant=}%
5239 #####2,#####3}%
5240 \gmu@tempb\@nil{##2}%
5241 \edef\gmu@tempa{\@xa\gmu@tempb\gmu@tempa\@nil\pt}%
5242 }
5243 }

\gmu@calculateslant 5244 \def\gmu@calculateslant#1{%
5245 \gmu@getslant
5246 \edef\gmu@tempa{\the\numexpr\dimexpr\fontdimen1\font+%
5247 \gmu@tempa\% \gmu@tempa bears the number of scaled points of total
5248 slant (\fontdimen1\font+slant=... if present) per 1pt of #1.}
5249 \edef\gmu@tempa{\the\numexpr\gmu@tempa*%
5250 \numexpr\fontdimen5\font\relax/\numexpr\fontcharht\%
5251 \font`#1%
5252 \relax\% we scale the total slant of #1 by the ratio of original and scaled
5253 height of #1.}
5254 }
```

```

5271 \edef\gmu@tempd{%
5272 \the\dimexpr\gmu@tempa_sp-\fontdimen1\font}%
5273 and we subtract slant-fontdimen from the scaled total slant.
5274 \ifdim\gmu@tempd=\z@\emptyify\gmu@tempd
5275 \else\edef\gmu@tempd{%
5276 \nx\addfontfeature{FakeSlant=\strip@pt\dimexpr%
5277 \gmu@tempd} }%
5278 \fi}
5280 \DeclareCommand\gmu@cepstnof{0{\gmu@tempa}}%
5281 a cs to be \xdefed the font specification,
5282 s% not used really,
5283 m% \fontspec token or name of feature font (Italic, Bold, SmallCaps,
5284 % BoldItalic),
5285 O{, \Scale=MatchLowercase}%
5286 fontspec font features (key=val)
5287 { \gmu@cepstnof's body
5288 \def\gmu@reservedc##1:##2:\@nil{%
5289 \ifx:##2\else\RawFeature={\gmu@maybestripcomma##2,,%
5290 \@nil}\fi}%
5291 \def\gmu@reservedd##1/##2:\@nil{%
5292 to check whether font name contains / (which may not be true!)
5293 \ifx##2\@xa\@firstoftwo\else\@xa\@secondoftwo\fi}%
5295 \def\gmu@reservede##1:##2:\@nil{%
5296 to check whether the font name contains : when it doesn't contain /.
5297 \ifx##2\@xa\@firstoftwo\else\@xa\@secondoftwo\fi}%
5299 \edef\gmu@reserveda{%

```

now, reserved B parses the font name and features. It uses an auxiliary reserved C because after / may be or may not be features specification.

```

5303 \@xa\@xa\@xa\gmu@reservedd\@xa\meaning\the\font//\@nil
5304 { font name doesn't contain a slash
5305 \@xa\@xa\@xa\gmu@reservede\@xa\meaning\the\font::%
5306 \@nil
5307 { nor does it contain a colon
5308 \def\nx\gmu@reservedb\detokenize{select\font
5309 "}##1\detokenize{"}##2\@nx\@nil{%
5310 \ifx\fontspec#3%
5311 \@nx\@nx\@nx\fontspec[\@gobble#4\empty] {##1}%
5312 gobble
5313 a comma
5314 \else
5315 #3Font={##1}, \#3Features={\@gobble\#4%
5316 \empty}%
5317 \fi
5318 }%
5319 { no slash but there is a colon
5320 \def\nx\gmu@reservedb\detokenize{select\font
5321 "}##1:##2\detokenize{"}##3\@nx\@nil{%
5322 \ifx\fontspec#3%
5323 \@nx\@nx\@nx\fontspec[\@nx\gmu@reservedc##2::%
5324 \@nx\@nil#4]{##1}%

```

```

5321 \else
5322 #3Font={####1}, #3Features={\@nx%
5323 \gmu@reservedc##2::\@nx\@nil#4}%
5324 \fi
5325 }%
5326 }%
5327 }% of 'no slash' case
5328 {%
5329 font name contains a slash
5330 \def\@nx\gmu@reservedb\detokenize{select_font
5331 " }####1####2\detokenize{" }####3\@nx\@nil{%
5332 \ifx\fontspec#3%
5333 \@nx\@nx\@nx\fontspec[\@nx\gmu@reservedc##2::%
5334 \@nx\@nil#4]{####1}%
5335 \else
5336 #3Font={####1}, #3Features={\@nx%
5337 \gmu@reservedc##2::\@nx\@nil#4}%
5338 \fi
5339 }% of \gmu@reservedb
5340 }% of 'slash present' case
5341 } \gmu@reserveda
5342 \xdef#1{\@xa\@xa\@xa\gmu@reservedb\@xa\meaning\the\font%
5343 \@nil}%
5344 }%
5345 \def\gmu@maybestripcomma#1,,#2\@nil{#1}
5346 \gmu@setbasefont
5347 \pdef\gmu@setbasefont{\@xa\let\@xa\gmu@basefont\the\font}
5348 \let\setbasefont\gmu@setbasefont
5349 \gmu@calc@scale
5350 \DeclareCommand\gmu@calc@scale{%
5351 O{1}%
5352 a factor,
5353 m%
5354 number of the fontdimen
5355 }
5356 {\begingroup

We 'descale' the current font:
5357 \gmu@cepstnof\fontspec[, Scale=1]\gmu@tempa
5358 \@xa\let\@xa\gmu@currfont@descaled\the\font
5359 \gmu@basefont
5360 \gmu@cepstnof\fontspec[, Scale=1]\gmu@tempa
5361 now also the base font is descaled.

5362 \xdef\gmu@fontscale{%
5363 \strip@pt
5364 \dimexpr\@ipt\@*
5365 \numexpr\dimexpr#1\fontdimen#2\font\relax\relax\@/
5366 \numexpr\fontdimen#2\gmu@currfont@descaled\relax\relax\@/
5367 \relax}%
5368 {\endgroup}

```

Varia

A very neat macro provided by doc. I copy it ~verbatim.

\gmu@tilde 5380 \def\gmu@tilde {%

```
5381 \leavevmode\lower.8ex\hbox{$\backslashwidetilde{\mbox{~}}$}
```

Originally there was just `\~` instead of `\mbox{~}` but some commands of ours do redefine `\~`.

```
5387 \AtBeginDocument{\% to bypass redefinition of \~ as a text command with various encodings}
```

```
\texttilde 5389 \pdef\texttilde{\%
```

```
5396 \ifnextchar/{\gmu@tilde\kern-0.1667em\relax}{\gmu@tilde}}
```

We prepare the proper kerning for “~/”.

The standard `\obeyspaces` declaration just changes the space’s `\catcode` to <sub>13</sub> (‘active’). Usually it is fairly enough because no one ‘normal’ redefines the active space. But we are *not* normal and we do *not* do usual things and therefore we want a declaration that not only will `\active`ate the space but also will (re)define it as the `\~` primitive. So define `\gmobeyspaces` that obeys this requirement.

(This definition is repeated in `gmverb.`)

```
5408 \foone{\catcode`\~\active}{%
```

```
\gmobeyspaces 5409 {\def\gmobeyspaces{\let\~\catcode`\~\active}}
```

While typesetting poetry, I was surprised that sth. didn’t work. The reason was that original `\obeylines` does `\let` not `\def`, so I give the latter possibility.

```
5416 \foone{\catcode`^\M\active}{% the comment signs here are crucial.
```

```
\defobeylines 5417 {\def\defobeylines{\catcode`^\M=\13\def^\M{\par}}}}
```

Another thing I dislike in L<sup>A</sup>T<sub>E</sub>X yet is doing special things for `\...skip`’s, ‘cause I like the Knuthian simplicity. So I sort of restore Knuthian meanings:

```
\deksmallskip 5426 \def\deksmallskip{\vskip\smallskipamount}
```

```
\undecksmallskip 5427 \def\undecksmallskip{\vskip-\smallskipamount}
```

```
\dekmedbigskip 5428 \def\dekmedbigskip{\vskip\glueexpr\medskipamount+%\smallskipamount}
```

```
\dekmedskip 5429 \def\dekmedskip{\vskip\medskipamount}
```

```
\dekbigs skip 5430 \def\dekbigs skip{\vskip\bigskipamount}
```

```
\hfillneg 5433 \def\hfillneg{\hskip\opt\plus\hskip\relax}
```

A mark for the **TO-DO!**s:

```
\TODO 5438 \newcommand*{\TODO}[1][]{\%
```

```
5439 \sffamily\bfseries\huge\texttt{TO-DO!}\if\relax\#1\relax\else\space\fi\#1}}
```

I like two-column tables of contents. First I tried to provide them by writing `\begin{multicols}{2}` and `\end{multicols}` out to the .toc file but it worked wrong in some cases. So I redefine the internal L<sup>A</sup>T<sub>E</sub>X macro instead.

```
\twocoltoc 5474 \newcommand*{\twocoltoc}{%
```

```
5475 \RequirePackage{multicol}%
```

```
\@starttoc 5476 \def\@starttoc##1{%
```

```
5477 \begin{multicols}{2}\makeatletter\@input{\jobname.\#\#1}%
```

```
5478 \if@filesw\@xa\newwrite\csname\@tfname\endcsname
```

```
5479 \immediate\openout\csname\@tfname\endcsname%
```

```
\jobname.\#\#1\relax
```

```
5480 \fi
```

```
5481 \nobreakfalse\end{multicols}}}
```

```
5483 \onlypreamble\twocoltoc
```

The macro given below is taken from the `multicol` package (where its name is `\enough@room`). I put it in this package since I needed it in two totally different works.

```
\enoughpage 5488 \DeclareCommand\enoughpage {%
5491 Q{+-0123456789}%
5492 B{z\baselineskip}%
5493 (optional short version (number of \baselineskip)s)
5494 (2) optional (formerly mandatory) long version of re-
5495 quired room on a page
5496 >is
5497 B{} % (3) what if the room is enough
5498 >isB{\newpage} % (4) what if there's too little room on a page
5499 } %
5500 \if
5501 \ifdim\dimexpr\pagegoal-\pagetotal<\dimexpr
5502 \IfValueTF{\#1}{\#1\baselineskip}{\#2}\relax
5503 1\elseo\fi
5504 \unless\ifdim\dimexpr\pagegoal-\pagetotal<\z@
5505 1\else2\fi
5506 }
```

We check both whether space left on page is larger than we check *and* whether it's nonnegative (the latter happens when we are already on the next page). Therefore this test will not work properly for large values of `#1` or values close to `\textwidth`.

```
5518 \xa\@firstoftwo
5519 \else
5520 \xa\@secondoftwo
5521 \fi
5522 {\#4}{\#3}%
5523 }
```

An equality sign properly spaced:

```
\equals 5532 \pdef\equals{\hunskip${}={\$}\ignorespaces}
```

And for the L<sup>A</sup>T<sub>E</sub>X's pseudo-code statements:

```
\eequals 5534 \pdef\eequals{\hunskip${}==\$}\ignorespaces
\cdot 5536 \pdef\cdot{\hunskip${}.\$}\ignorespaces
```

While typesetting a UTF-8 ls-R result I found a difficulty that follows: UTF-8 encoding is handled by the `inputenc` package. It's O.K. so far. The UTF-8 sequences are managed using active chars. That's O.K. so far. While writing such sequences to a file, the active chars expand. You feel the blues? When the result of expansion is read again, it sometimes is again an active char, but now it doesn't star a correct UTF-8 sequence.

Because of that I wanted to 'freeze' the active chars so that they would be `\written` to a file unexpanded. A very brutal operation is done: we look at all 256 chars' catcodes and if we find an active one, we `\let` it `\relax`. As the macro does lots and lots of assignments, it shouldn't be used in `\edefs`.

```
\freeze@actives 5556 \def\freeze@actives{%
5557 \count\z@\z@
5558 \@whilenum\count\z@<\@ccclvi\do{%
5559 \ifnum\catcode\count\z@=\active
5560 \uccode`\~=\count\z@
5561 \uppercase{\let`\~\relax}%
5562 \fi
5563 \advance\count\z@\@ne}}
```

A macro that typesets all 256 chars of given font. It makes use of `\@whilenum`.

```
\ShowFont 5570 \newcommand*{\ShowFont}[1][6]{%
 5571 \begin{multicols}{#1}[The current font (the f@encoding\encoding):]
 5572 \parindent\z@
 5573 \count\z@\m@ne
 5574 \@whilenum\count\z@<\@ccnv\do{%
 5575 \advance\count\z@\@ne
 5576 \the\count\z@:~\char\count\z@\par}
 5577 \end{multicols}}
```

A couple of macros for typesetting liturgical texts such as psalmody of Liturgia Horarum. I wrap them into a declaration since they'll be needed not every time.

```
\liturgiques 5585 \newcommand*{\liturgiques}[1][red]{% Requires the color package.
 5586 \gmu@RPfor{xcolor}\color%
\czerwo 5587 \newcommand*{\czerwo}{\small\color{#1}}% environment
\czer 5588 \newcommand{\czer}[1]{\leavevmode{\czerwo#1}}% we leave vmode
 because if we don't, then verse's \everypar would be executed in
 a group and thus its effect lost.
 5591 \StoreMacro*%
* 5592 \def*{\czer{\storedcsname{*}}}%
\+ 5593 \def\+{\czer{\+}}%
\niecz 5594 \newcommand*{\niecz}[1]{\textcolor{black}{##1}}%
 5595 }
```

After the next definition you can write `\gmu@RP[<options>]{<package>} {<CS>}` to get the package #2 loaded with options #1 if the CS#3 is undefined.

```
\gmu@RPfor 5600 \newcommand*{\gmu@RPfor}[3][]{%
 5602 \ifx\relax#1\relax\emptyify\gmu@resa
\gmu@resa 5603 \else\def\gmu@resa{[#1]}%
 5604 \fi
 5605 \xa\RequirePackage\gmu@resa{#2}}
```

Since inside document we cannot load a package, we'll redefine `\gmu@RPfor` to issue a request before the error issued by undefined CS.

```
\gmu@RPfor 5611 \AtBeginDocument{%
 5612 \renewcommand*{\gmu@RPfor}[3][]{%
 5613 \unless\ifdefined#3%
 5614 \@ifpackageloaded{#2}{}{%
 5615 \typeout{^^J! Package `#2' not loaded!!! (%
 \on@line)^^J}}%
 5616 \fi}}
```

It's very strange to me but it seems that `c` is not defined in the basic math packages. It is missing at least in the *Symbols* book.

```
\continuum 5622 \pprovide\continuum{%
 5623 \gmu@RPfor{eufrak}\mathfrak\ensuremath{\mathfrak{c}}}}
```

And this macro I saw in the *ltugproc* document class and I liked it.

```
\iteracro 5627 \def\iteracro{%
\acro 5628 \pdef\acro##1{%
 5629 \begingroup
```

```

5630 \acropresetting
5631 \gmu@acrospace{\#1}\gmu@acrospace
5632 \endgroup
5633 }%
5634 }

5636 \empty\acropresetting
5638 \iteracro

\gmu@acrospace 5640 \def\gmu@acrospace{\#1\#2\gmu@acrospace}{%
5641 \gmu@acroinner{\#1\gmu@acroinner
5642 \ifx\relax\#2\relax\else
5643 \space
5644 \afterfi{\gmu@acrospace{\#2\gmu@acrospace}}% when #2 is nonempty,
5645 it is ended with a space. Adding one more space in this line resulted in
5646 an infinite loop, of course.
5647 \fi}
5648 }

\gmu@acroinner 5651 \def\gmu@acroinner{\#1{%
5652 \ifx\gmu@acroinner{\#1\relax\else
5653 \ifcat_a\@nx{\#1\relax}%
5654 \ifnum`#1=\uccode`#1%
5655 {\acroc{\#1}}%
5656 \else{\#1} tu bylo \smallerr
5657 \fi
5658 \else{\#1}%
5659 \fi
5660 \afterfi\gmu@acroinner
5661 \fi}

```

We extract the very thing done to the letters to a macro because we need to redefine it in fonts that don't have small caps.

```
\acroc{\#1} was: \scshape\lowercase
5665 \pdef\acroc{\smaller}% was: \scshape\lowercase
5666 }
```

Since the fonts I am currently using do not support required font feature, I skip the following definition.

```
\IMHO 5671 \def\IMHO{\acroc{IMHO}}
\AKA 5672 \def\AKA{\acroc{AKA}}
\usc 5674 \pdef\usc{\{\addfontfeature{Letters=UppercaseSmallCaps}\#1\}}
\uscacro 5676 \def\uscacro{\let\acroc\usc}
```

Probably the only use of it is loading gmdocc.cls 'as second class'. This command takes first argument optional, options of the class, and second mandatory, the class name. I use it in an article about gmdoc.

```
\secondclass 5694 \def\secondclass{%
\ifSecondClass 5695 \newif\ifSecondClass
5696 \SecondClasstrue
5697 \@fileswithoptions{\@clsextension}{[outeroff,gmeometric]}{gm-
 docc} it's loading gmdocc.cls with all the bells and whistles except the error
 message.
```

Cf. *The T<sub>E</sub>Xbook* ex. 11.6.

A line from L<sup>A</sup>T<sub>E</sub>X:

```
% \check@mathfonts \fontsize\sf@size\z@\math@fontsfalse \selectfont
```

didn't work as I would wish: in a \footnotesize's scope it still was \scriptsize, so too large.

```
\gmu@dekfraccsimple 5709 \def\gmu@dekfraccsimple#1/#2{\leavevmode\kern.1em
5710 \raise.7ex\hbox{ $\frac{\gmu@fracfontsetup#1}{\gmu@numeratorkern}$
5711 \gmu@fracfontsetup#1}\gmu@numeratorkern
5712 \gmu@fracfontsetup#2}\gmu@denominatorkern
5713 { $\frac{\gmu@fracfontsetup#1}{\gmu@fracfontsetup#2}$
5714 \if@gmu@mmhbox\egroup\fi}

\gmu@fracfontsetup 5718 \def\gmu@fracfontsetup{ $\frac{\smaller[3]{\addfontfeature{FakeBold=1}}}$
5719 }

\dekfraccsimple 5722 \def\dekfraccsimple{ $\frac{\let\dekfracc@args\gmu@dekfraccsimple}$
5723 }

\dekfraccslash 5725 \@ifXeTeX{\def\dekfraccslash{\char"2044}}
5726 {\def\dekfraccslash{/}}% You can define it as the fraction
 slash, \char"2044
5728 \dekfraccsimple
```

A macro that acts like \, (thin and unbreakable space) except it allows hyphenation afterwards:

```
\ikern 5736 \newcommand*\ikern{\, \penalty\@M\hskip\z@skip\relax}
```

And a macro to forbid hyphenation of the next word:

```
\nohy 5740 \pdef\nohy{\leavevmode\kernosp\relax}
\yeshy 5741 \pdef\yeshy{\leavevmode\penalty\@M\hskip\z@skip}
```

In both of the above definitions 'osp' not \z@ to allow their writing to and reading from files where @ is 'other'.

### \include not only .tex's

\include modified by me below lets you to include files of any extension provided that extension in the argument.

If you want to \include a non-.tex file and deal with it with \includeonly, give the latter command full file name, with the extension that is.

```
\gmu@gettext 5756 \def\gmu@gettext#1.#2@@nil{ $\def\gmu@filename{#1}$
5757 \def\gmu@fileext{#2}
5758 }

5760 \def\include#1{\relax
5761 \ifnum\@auxout=\@partaux
5762 \@latex@error{\string\include\space cannot be nested} \relax
5763 \else \@include#1\fi}

\@include 5765 \def\@include#1{ $\def\gmu@gettext#1.\@nil$
5766 \ifx\gmu@fileext\empty\def\gmu@fileext{tex}\fi
5767 \clearpage
5770 \if@files
```

```

5771 \immediate\write\@mainaux{\string\@input{%
5772 \gmu@filename.aux} }%
5773 \fi
5774 \if@tempswattrue
5775 \if@partsw
5776 \edef\reserved@b{\#1}%
5777 \for\reserved@a:=\partlist\do{%
5778 \ifx\reserved@a\reserved@b\@tempswattrue\fi}%
5779 \fi
5780 \if@tempswa
5781 \let\@auxout\@partaux
5782 \if@files w
5783 \immediate\openout\@partaux\gmu@filename.aux
5784 \immediate\write\@partaux{\relax}%
5785 \fi
5786 \input{\gmu@filename.\gmu@fileext}%
5787 \inlasthook
5788 \clearpage
5789 \writeckpt{\gmu@filename}%
5790 \if@files w
5791 \immediate\closeout\@partaux
5792 \fi
5793 \else

```

If the file is not included, reset \include deadcycles, so that a long list of non-included files does not generate an 'Output loop' error.

```

5797 \deadcycles\z@
5798 \nameuse{cp@\gmu@filename}%
5799 \fi
5800 \let\@auxout\@mainaux}

\whenonly \gmu@whonly 5803 \newcommand\whenonly[3]{%
5804 \def\gmu@whonly{\#1,}%
5805 \ifx\gmu@whonly\partlist\afterfi{\#2}\else\afterfi{\#3}\fi}

```

I assume one usually includes chapters or so so the last page style should be closing.

```
\inlasthook 5809 \def\inlasthook{\thispagestyle{closing}}
```

### Switching on and off parts of one file

The \include facility is very nice only it forces you to split your source in many files. Therefore I provide a tool analogous to \include and using the same \includeonly mechanism/list to switch on and off parts of the same source file.

```

\filepart 5818 \def\filepart#1{\relax
5819 \ifnum\@auxout=\@partaux
5820 \latext@error{\string\filepart\space cannot be nested}%
 \eha
5821 \else\afterfi{\@filepart#1}\fi}

\@filepart 5823 \def\@filepart#1{%
5824 \clearpage
5825 \edef\gmu@filepartname{\#1}%
 we'll use it later
5826 \if@files w

```

```

5827 \immediate\write\@mainaux{\string\@input{#1.aux} }%
5828 \fi
5829 \tempswatrue
5830 \if@partsw
5831 \tempswafalse
5832 \for\reserved@a:=\partlist\do{%
5833 \ifx\reserved@a\gmu@filepartname\tempswatrue\fi}%
5834 \fi
5835 \if@tempswa
5836 \let\auxout\partaux
5837 \if@files w
5838 \immediate\openout\partaux\#1.aux
5839 \immediate\write\partaux{\relax}%
5840 \fi
5841 \xa\@firstoftwo
5843 \else

```

If the file is not included, reset \include deadcycles, so that a long list of non-included files does not generate an 'Output loop' error.

```

5847 \deadcycles\z@
5848 \nameuse{cp@\gmu@filepartname}%
5849 \let\auxout\mainaux
5850 \xa\@secondoftwo
5851 \fi
5852 {\iftrue}%
5853 {\let\endfilepart\fi
5854 \csname\gm@skipped\#1\endcsname
5855 \def\next{\RestoreMacro\endfilepart
5856 \ifnextchar\bgroup{\show\NextBgroup\@gobble}{}}
5857 \xa\next\iffalse}%
5858 }

\endfilepart 5861 \DeclareCommand\endfilepart{b}{% Note the argument is not used really.
 Maybe later we'll use it for checking of proper matching. Or maybe not.
5863 \inlasthook
5864 \clearpage
5865 \writeckpt{\gmu@filepartname}%
5866 \if@files w
5867 \immediate\closeout\partaux
5868 \fi
5869 \fi% this \fi closes \Iftrue put by line 5841.
5870 \let\auxout\mainaux
5871 }

5873 \StoreMacro\endfilepart

\nofileparts 5875 \def\nofileparts{%
5876 \let\filepart\gobble
\endfilepart 5877 \DeclareCommand\endfilepart{b}{%}
5878 }

```

#### **Fix of including when fontspec is used**

The fontspec package creates counters for font families. If a fontspec command is used in a part of a document and then such a part is skipped, an error occurs 'No counter

`zf@fam@... defined'.` Now we fix that by ensuring all the counters are defined before they are set.

Note it's a draft version which doesn't support resetting of one counter within another.

```
\includecountfix 5890 \def\includecountfix{%
 \wckptelt 5891 \def\@wckptelt##1{%
 \immediate\write\@partaux{%
 \providecounter{##1}%
 to provide the font counters defined in parts
 of the document.
 \string\setcounter{##1}{\the\@nameuse{c##1}}}}%
 5897 }

\providecounter 5899 \pdef\providecounter#1{%
 #1 5900 \unless\ifcsname_c#1\endcsname\newcounter{#1}\fi}
```

### Faked small caps

```
\gmu@scapLetters 5905 \def\gmu@scapLetters#1{%
 5906 \ifx#1\relax\relax\else% two \relaxes to cover the case of empty #1.
 5907 \ifcat\@nx#1\relax
 5908 \ifnum\the\lccode`#1=\#1\relax
 5909 {\fakescapscore\MakeUppercase{#1}}%
 not Plain \uppercase
 because that works bad with inputenc.
 5911 \else#1%
 5912 \fi
 5913 \else#1%
 5914 \fi%
 5915 \gmu@scapLetters
 5916 \fi}%
\gmu@scapSpaces 5918 \def\gmu@scapSpaces#1\@nil{%
 5919 \ifx#1\relax\relax
 5920 \else\gmu@scapLetters#1\relax
 5921 \fi
 5922 \ifx#2\relax\relax
 5923 \else\afterfi{\gmu@scapSpaces#2\@nil}%
 5924 \fi}
\gmu@scapss 5926 \def\gmu@scapss#1\@nil{{\def~{\nobreakspace}}%
\nobreakspace 5927 \gmu@scapSpaces#1\@nil}%
 %\def\\{\newline}\relax adding
 redefinition of \\ caused stack overflow. Note it disallows hyphenation
 except at \-.}

\fakescaps 5931 \pdef\fakescaps#1{{\gmu@scapss#1\@nil}}
 5933 \let\fakescapscore\gmu@scalematchX
 Experimente z akcentami patrz no3.tex.

\tinycae 5936 \def\tinycae{{\tiny AE}}%
 to use in \fakescaps[\tiny]...
 5938 \RequirePackage{calc}
 wg \zf@calc@scale pakietu fontspec.

\gmu@scalar 5942 \@ifpackageloaded{fontspec}{%
 5943 \def\gmu@scalar{1.0}%
 5944 \def\zf@scale{}%
\gmu@scalematchX 5945 \def\gmu@scalematchX{%
```

```

5946 \begingroup
5947 \ifx\zf@scale\empty\def\gmu@scalar{1.0}%
5948 \else\let\gmu@scalar\zf@scale\fi
5949 \setlength{\tempdima}{\fontdimen5\font}%
5950 \setlength{\tempdimb}{\fontdimen8\font}%
5951 5—ex height
5952 8— $\text{X}\ddot{\text{T}}\text{E}\text{X}$ synthesised
5953 uppercase height.
5954 \divide{\tempdimb}{1000}\relax
5955 \divide{\tempdima}{\tempdimb}
5956 \setlength{\tempdima}{\tempdima*\real{\gmu@scalar}}%
5957 \gm@ifundefined{fakesc@extrascale}{}{%
5958 \setlength{\tempdima}{\tempdima*\real{%
5959 \fakesc@extrascale}}%
5960 }%
5961 \tempcnta=\tempdima
5962 \divide{\tempcnta}{1000}\relax
5963 \tempcntb=-1000\relax
5964 \multiply{\tempcntb}{\tempcnta}
5965 \advance{\tempcntb}{\tempdima}
5966 \xdef\gmu@scscale{\the\tempcnta.%
5967 \ifnum\tempcntb<100\relax\fi
5968 \ifnum\tempcntb<10\relax\fi
5969 \the\tempcntb}%
5970 \endgroup
5971 \addfontfeature{Scale=\gmu@scscale}%
5972 } \let\gmu@scalematchX\smallerr
5973 \def\fakescextrascale#1{\def\fakesc@extrascale{#1}}
5974 \fakesc@extrascale

```

#### See above/see below

To generate a phrase as in the header depending of whether the respective label is before or after.

```

\wyzejnizej 5977 \newcommand*\wyzejnizej[1]{%
5978 \edef\gmu@tempa{\gm@ifundefined{r@#1}{\arabic{page}}{%
5979 \xa\x\@xa\@xa\@secondoftwo\csname r@#1\endcsname}%
5980 \ifnum\gmu@tempa<\arabic{page}\relax\wy\.zej\fi
5981 \ifnum\gmu@tempa>\arabic{page}\relax\ni\.zej\fi
5982 \ifnum\gmu@tempa=\arabic{page}\relax\@xa\ignorespaces\fi
5983 }

```

#### **luzniej** and **napapierki**—environments used in page breaking for money

The name of first of them comes from Polish typesetters' phrase “rozbijać [skład] na papierki”—‘to broaden [leading] with paper scratches’.

```

\napapierkistretch 5993 \def\napapierkistretch{0,3pt}%
5994 It's quite much for 11/13pt leading.
\napapierkicore 5995 \def\napapierkicore{\advance\baselineskip{%
5996 \optplus\napapierkistretch}\relax}
5997
6003 \DeclareEnvironment{napapierki}[s]{%
6004 \par\IfValueT{#1}{\global
6005 }%
6006 \napapierkicore}
6007 {%
6008 \par

```

```

6010 \IfValueT{#1}{\global\baselineskip=1\baselineskip\relax
6011 }%
6012 }% so that you can use \napapierki*... \endnapapierki* in interlacing environments.

\gmu@luzniej 6017 \newcount\gmu@luzniej
\luzniejcore 6019 \newcommand*\luzniejcore[1][1]{%
6020 \advance\gmu@luzniej@\ne% We use this count to check whether we open
 the environment or just set \looseness inside it again.
6022 \ifnum\gmu@luzniej=\@ne\multiply\tolerance by 2\fi
6023 \looseness=#1\relax}

```

After `\begin{luzniej}` we may put the optional argument of `\luzniejcore`

```
luzniej 6027 \newenvironment*\luzniej}{\par\luzniejcore}{\par}
```

The starred version sets `\looseness` in `\everypar`, which has its advantages and disadvantages.

```
luzniej* 6032 \newenvironment*\luzniej*}[1][1]{%
6033 \multiply\tolerance by 2\relax
6034 \everypar{\looseness=#1\relax}}{\par}
```

`\nawj` 6036 `\newcommand*\nawj{\kern .1em\relax}`% a kern to be put between parentheses and letters with descendants such as *j* or *y* in certain fonts.

The original `\pauza` of polski has the skips rigid (one is even a kern). We make the skips flexible. Moreover, our `\pauza` begins with `\ifhmode` to be usable also at the beginning of a line where it marks a part of a dialogue.

```
\pauza@skipcore 6045 \def\pauza@skipcore{\hskip.2em plus.1em\relax
6046 \pauzacore
6047 \@ifnextchar,{% 2009/11/22 added a special case of a comma following
 pauza
6048 }{\hskip.2em plus.1em\relax\ignorespaces}%
\ppauza@skipcore 6050 \def\ppauza@skipcore{\unskip\penalty10000\hskip.2em plus.1em\relax
6051 \ppauza@dash\hskip.2em plus.1em\ignorespaces}
6054 \AtBeginDocument{%
\pauza 6055 \pdef\pauza{%
6056 \ifhmode
6057 \unskip\penalty10000
6058 \hskip.2em plus.1em\relax
6059 \pauzacore\hskip.2em plus.1em\relax\ignorespaces%
6060 \else
6061 \pauzadial
6062 \fi}%

```

According to *Instrukcja technologiczna. Skład ręczny i maszynowy* the dialogue dash (in Polish) should be followed by a rigid hskip of  $\frac{1}{2}$ em.

```
\pauzadial 6067 \pdef\pauzadial{%
6068 \leavevmode\pauzacore\penalty10000\hskip.5em%
 \ignorespaces}
```

And a version with no space at the left, to begin a `\noindent`ed paragraph explaining e.g. a quotation:

```
\lpauza 6072 \pdef\lpauza{%
```

```

6073 \leavevmode
6074 \pauzacore\hskip.2em plus 0.1em\ignorespaces}%

```

We define `\ppauza` as an en dash surrounded with thin stretchable spaces and sticking to the upper line or bare but discretionary depending on the next token being `space_0`. Of course you'll never get such a space after a literal CS so an explicit `\ppauza` will always result with a bare discretionary en dash, but if we `\let - \ppauza...`

```

\ppauza 6083 \pdef\ppauza{%
6084 \ifvmode\PackageError{gmutils}{%
6085 command\bslash\ppauza (en dash) not intended for%
6086 vmode.}%
6087 Use\bslash\ppauza (en dash) only in number and%
6088 numeral ranges.}%
6089 \else
6090 \unskip\discretionary
6091 {\ppauza@dash}{\ppauza@dash}{\ppauza@dash}%
6092 \fi}%
6093 }% of at begin document
6094
6095 \ifdefined\XeTeXversion
6096 \AtBeginDocument{%
6097 \pdef\—{%
6098 \ifhmode
6099 \unskip\penalty10000
6100 \afterfi{%
6101 \@ifnextspace{\pauza@skipcore}{%
6102 \@\ifnextchar,{\pauza@skipcore}{%
6103 \pauzacore\penalty\hyphenpenalty\hskip%
6104 \z@skip}}}}%
6105 }% of \afterfi's argument
6106 \else
6107 According to Instrukcja technologiczna. Skład ręczny i maszynowy the dialogue dash
6108 should be followed by a rigid hskip of $\frac{1}{2}$ em.
6109 \leavevmode\pauzacore\penalty10000\hskip0,5em%
6110 \ignorespaces
6111 \fi}%

```

The next command's name consists of letters and therefore it eats any spaces following it, so `\@ifnextspace` would always be false, therefore we don't use it.

```

\— 6114 \pdef\—{%
6115 \ifvmode\PackageError{gmutils}{%
6116 command\bslash\ppauza (en dash) not intended for%
6117 vmode.}%
6118 Use\bslash\ppauza (en dash) only in number and%
6119 numeral ranges.}%
6120 \else
6121 \afterfi{%
6122 \@ifnextspace{\ppauza@skipcore}{%
6123 \@ifnextMac{\ppauza@skipcore}{%
6124 \unskip\discretionary
6125 {\ppauza@dash}{\ppauza@dash}{\ppauza@dash}}}}%
6126 }%

```

```

6125 \fi
6126 }%
\emdash 6128 \def\emdash{\char`— }
6129 }% of at begin document

\longpauza 6131 \def\longpauza{\def\pauzacore{—}}
\pauzacore 6132 \longpauza
\shortpauza 6133 \def\shortpauza{%
\pauzacore 6134 \def\pauzacore{\hbox{—\kern,.23em\relax\llap{—}}}}%
\ppauza@dash 6135 \def\ppauza@dash{—}%
6138 \else % not XETEX
\longpauza 6139 \def\longpauza{\def\pauzacore{---}}
\pauzacore 6140 \longpauza
\shortpauza 6141 \def\shortpauza{%
\pauzacore 6142 \def\pauzacore{--\kern,.23em\relax\llap{--}}}}%
\ppauza@dash 6143 \def\ppauza@dash{--}%
6146 \fi% of if XETEX.

```

If you have all the three dashes on your keyboard (as I do), you may want to use them for short instead of `\pauza`, `\ppauza` and `\dywiz`. The shortest dash is defined to be smart in math mode and result with `—`.

```

6154 \ifdefined\XeTeXversion
6155 \foone{\catcode`—\active\catcode`—\active\catcode`-\active}{%
\adashes 6156 \def\adashes{\AtBeginDocument\adashes} % because \pauza is defined
 at begin document.
\adashes 6158 \AtBeginDocument{\def\adashes{%
\— 6159 \catcode`—\active\def—{\—}%
\— 6160 \catcode`—\active\def—{\—}%
6162 \addtomacro\dospecials{\do\—\do\—}%
6163 \addtomacro\@sanitize{\@makeother\—\@makeother\—}%
6164 \addtomacro\gmu@septify{\do\—\do\—\relax}%
6165 } }%
6166 \else
6167 \relaxen\adashes
6168 \fi

```

The hyphen shouldn't be active IMHO because it's used in T<sub>E</sub>X control such as `\hskip-2pt`. Therefore we provide the `\ahyphen` declaration reluctantly, because sometimes we need it and always use it with caution. Note that my active hyphen in vertical and math modes expands to `—12`.

```

\gmu@dywiz 6177 \def\gmu@dywiz{\ifmmode-\else
6178 \ifvmode-\else\afterfifi\dywiz\fi\fi}%
6180 \foone{\catcode`-\active}{% aktivnyj diefis aktywny dywiz active hyphen
\ahyphen 6181 \def\ahyphen{\let-\gmu@dywiz\catcode`-\active}}

```

To get current time. Works in  $\epsilon$ -T<sub>E</sub>Xs, including X<sub>E</sub>T<sub>E</sub>X. `\czas` typesets 8.18 and `\czas[:]` typesets 8:18.

```

\czas 6186 \newcommand*\czas[1][.]{%
6187 \the\numexpr(\time-30)/60\relax#1%
6188 \tempcnta=\numexpr\time-(\time-30)/60*60\relax
6189 \ifnum\tempcnta<10\fi\the\tempcnta}

```

```

6192 \@ifXeTeX{%
\textbullet 6193 \pdef\textbullet{%
6196 \iffontchar\font"2022\char"2022\else\ensuremath{%
6197 \bullet}\fi}%
\glyphname 6198 \pprovide\glyphname#1{%
6200 \XeTeXglyph\numexpr\XeTeXglyphindex"#1"\relax\relax}%
6201 XETX ... \numexpr is redundant.
6202 }
\textbullet 6203 {\def\textbullet{\ensuremath{\bullet}}}

tytulowa 6205 \newenvironment*{tytulowa}{\newpage}{\par\thispagestyle{%
6206 empty}\newpage}

```

To typeset peoples' names on page 4 (the editorial page):

```
\nazwired 6208 \def\nazwired{\quad\textrm{}}
```

### Typesetting dates in my memoirs

A date in the YYYY-MM-DD format we'll transform into 'DD mmmm YYYY' format or we'll just typeset next two tokens/ {...} if the arguments' string begins with --. The latter option is provided to preserve compatibility with already used macros and to avoid a starred version of \thedate and the same time to be able to turn \datef off in some cases (for SevSev04.tex).

```

\polskadata 6222 \pdef\polskadata{%
\gmu@datefs1 6223 \DeclareCommand\gmu@datefs1{%
6224 Q{0123456789\bgroup}>iT{/ -}\ % (1) year
6225 Q{0123456789\bgroup}>iT{/ -}\ % (2) month
6226 Q{0123456789\bgroup}\ % (3) day
6227 T{, }\K{\#\!\!1\gmu@datefs1}\ % (4, 5) additional stuff after comma
6228 }\ %
6229 \IfValueF{\#\!\!2}{\PutIfValue{\#\!\!3}}%
6230 \IfValueT{\#\!\!2}{%
6231 \tempcnta=0\#\!\!3\relax\the\tempcnta
6232 \ifcase\#\!\!2\relax\or__stycznia\or__lutego\%
6233 \or__marca\or__kwietnia\or__maja\or__czerwca\or__
6234 lipca\or__sierpnia\%
6235 \or__września\or__października\or__listopada\or__
6236 grudnia\else
6237 \fi\ %
6238 \IfValueT{\#\!\!1}{\space\#\!\!1}%
6239 \PutIfValue{\#\!\!4}\IfValueT{\#\!\!5}{\#\!\!5}%
6240 }\% of \gmu@datefs1.
6241 }% of \polskadata
6242 \polskadata

```

For documentation in English:

```

\englishdate 6245 \pdef\englishdate{%
\gmu@datefs1 6246 \DeclareCommand\gmu@datefs1{%
6247 Q{0123456789\bgroup}>iT{/ -}\ % (1) year
6248 Q{0123456789\bgroup}>iT{/ -}\ % (2) month
6249 Q{0123456789\bgroup}\ % (3) day
6250 T{, }\K{\#\!\!1\gmu@datefs1}\ % (4, 5) additional stuff after comma

```

```

6251 }{%
6252 \IfValueF{\#2}{\PutIfValue{\#3}}%
6253 \IfValueT{\#2}{%
6254 \ifcase\#2\relax\or\January\or\February%
6255 \or\March\or\April\or\May\or\June\or\July\or%
6256 August%
6257 \or\September\or\October\or\November\or\December%
6258 \else
6259 {}%
6260 \fi}%
6261 \space
6262 \tempcnta=\#3\relax\the\tempcnta,
6263 \IfValueT{\#1}{\#1}%
6264 \PutIfValue{\#4}\IfValueT{\#5}{\#5}%
6265 }% of \gmu@datefs1.
6266 }%

```

Dates for memoirs to be able to typeset them also as diaries.

```

\ifdate 6269 \newif\ifdate
\bidate 6271 \pdef\bodate#1{%
6272 \gmu@datefs1#1\gmu@datefs1
6273 }

\linedate 6275 \pdef\linedate{\gm@ifstar\linedate@@\linedate@}
\linedate@ 6276 \pdef\linedate@{\#1{\linedate@{--{}{}#1}}}
\linedate@ 6277 \pdef\linedate@{\par
6278 \linedate@hook{\#1}%
6279 \ifdate\addvspace{\dateskipamount}%
6280 \possvfil% if we put it before \addvspace, the v-space is always added.
6281 \date@line{\footnotesize\itshape\bidate{\#1}}%
6282 \nopagebreak
6283 \else% %\ifnum\arabic{dateinsection}>0\dekbigsip\fi
6284 \addvspace{\bigskipamount}\possvfil
6285 \fi}% end of \linedate.

6286 \let\linedate@hook\@gobble
6287 \let\dateskipamount\medskipamount
\date 6291 \pdef\rdate{\let\date@line\rightline\linedate}
\date@left 6294 \def\date@left#1{\par{%
6295 \raggedright#1%
6296 \leftskip\z@skip
6301 \@@par}}}

\ldate 6303 \pdef\ldate{%
6305 \let\date@line\date@left
6306 \linedate}

\runindate 6308 \newcommand*\runindate[1]{%
6309 \paragraph{\footnotesize\itshape\gmu@datefs1\gmu@datef}%
6310 \stepcounter{dateinsection}}

```

I'm not quite positive which side I want the date to be put to so let's `let` for now and we'll be able to change it in the very documents.

```
6313 \let\thedata\ldate
```

```

\zwrobcy 6316 \pdef\zwrobcy#1{\emph{#1}}% ostinato, allegro con moto, garden party etc.,
 także kompliment

\tytul 6319 \pdef\tytul#1{\emph{#1}}

Maszynopis w świecie justowanym zrobi delikatną chorągiewkę. (The maszynopis
environment will make a delicate ragged right if called in a justified world.)

maszynopis 6325 \newenvironment{maszynopis}[1][]{#1\ttfamily
6326 \hyphenchar\font=45\relax% this assignment is global for the font.
6327 \tempskipa=\glueexpr\rightskip+\leftskip\relax
6328 \ifdim\gluestretch\tempskipa=\z@%
6329 \tolerance900
 it worked well with tolerance = 900.
6331 \advance\rightskip\by\z@plus0,5em\relax\fi
6332 \fontdimen3\font=\z@% we forbid stretching spaces...
% \fontdimen4\font=\z@ but allow shrinking them.
6334 \hyphenpenalty0% not to make TeX nervous: in a typewriting this marvelous
 algorithm of hyphenation should be turned off and every line broken
 at the last allowable point.

\pauzacore 6337 \StoreMacro\pauzacore
6338 \def\pauzacore{-\rlap{\kern-0,3em-}-%
6339 }{\par}

\justified 6343 \pdef\justified{%
6344 \leftskip=1\leftskip% to preserve the natural length and discard stretch
 and shrink.
6346 \rightskip=1\rightskip
6347 \parfillskip=1\parfillskip
6348 \advance\parfillskip\by\ospplus\ifil\relax
6350 \let\\@\normalcr}

To conform Polish recommendation for typesetting saying that a paragraph's last line
leaving less than \parindent should be stretched to fill the text width:

\fullpar 6355 \DeclareCommand\fullpar{%
6356 T{+-}%
6357 Q{+-0123456789}% optional looseness (most probably negative)
6358 }{%
6359 \begingroup
6360 \IfValueT{#1}{\looseness=#1\IfValueTF{#2}{#2}{1}\relax
6361 \multiply\tolerance\by\tw@
6362 }%
6363 \fullparcore
6364 \par
6365 \endgroup}

\fullparcore 6367 \pdef\fullparcore{%
6368 \hunskip
6369 \parfillskip\z@skip}

To conform Polish recommendation for typesetting that says that the last line of
a paragraph has to be 2\parindent long at least. The idea is to set \parfillskip naturally
rigid and long as \textwidth-2\parindent, but that causes non-negligible
shrinking of the inter-word spaces so we provide a declaration to catch the proper glue
where the parindent is set (e.g. in footnotes parindent is 0pt)

\twoparinit 6379 \newcommand*\twoparinit{%
 the name stands for 'last paragraph line's length
}

```

```

minimum two \parindent.

\def\twopar@defts{%
 \hsize-\leftskip-\rightskip-\fontcharwd\font`...}%
\def\twopar@atleast{2\parindent}%
\DeclareCommand\twopar{%
 T{+-}%
 (1) you can specify loosening the paragraph by one only by typing
 single + and tightening by one by typing single -.
 Q{+-0123456789}%
 A{\twopar@atleast}%
 >iT{\cipo{lagwa}}%
 \begingroup
 \IfValueT{\#1}{%
 \looseness=\#1\IfValueTF{\#2}{\#2}{1}\relax
 \multiply\tolerance by2
 }%
 \twoparcore<\#3>%
 \par
 \endgroup
}%
of \twopar.

\ifdefined\XeTeXversion
\DeclareCommand\twoparcore{%
 A{\twopar@atleast}>iT{\cipo{lagwa}}%
 \hunskip%
 it's O.K. it's in a group, it'll work anyway.
 \edef\gmu@tempa{\the\dimexpr\twopar@defts-\#1\relax}%
 \parfillskip=\glueexpr\gmu@tempa minus \gmu@tempa
 \relax% to delimit \glueexpr.
 \relax% to delimit the assignment.
}%
\else%
not XETEX—doesn't use \fontcharwd.
\DeclareCommand\twoparcore{%
 A{\twopar@default}>iT{\cipo{lagwa}}%
 \hunskip%
 it's O.K. it's in a group, it'll work anyway.
 {\setbox0=\hbox{\dots}%
 \xdef\gmu@tempa{\the\wd0}%
 \edef\gmu@tempa{%
 \the\dimexpr\hsize-\leftskip-\rightskip
 -\gmu@tempa 2\parindent\relax}%
 \parfillskip=\glueexpr\gmu@tempa minus \gmu@tempa
 \relax% to delimit \glueexpr.
 \relax% to delimit the assignment.
}%
\fi
\AtBeginDocument{%
 \unless\ifdefined\@parindent
 \newskip\@parindent
 \parindent=\parindent
 \fi
 \def\restoreparindent{\parindent\@parindent}%
}%
of \AtBeginDocument.
}%
of \twoparinit.

```

### For dati under poems

Or explanations under results of time.

```
\wherncore 6439 \DeclareCommand\wherncore{om} {%
 % [#1] optional value of \hskip of (left) indent of the parbox. If absent,
 % parbox is aligned right;
 % [#2] optional text for the datum parbox.
6445 \IfValueTF{#1}{\leftline{%
6446 \whernfont
6447 \hskip#1\relax\parbox
6448 {\dimexpr\textwidth-\leftskip-\rightskip-#1}%
6449 {#2}\% of \parbox,
6450 }\% of \leftline,
6451 }\% of ValueT{#1}.
6452 {\% ValueF{#1}:
6453 \rightline
6454 {\whernfont
6455 \whern@parbox{#2}\%
6456 }\% of \rightline,
6457 \setprevdepth
6458 }\% of ValueF{#1},
6459 }\% of \wherncore.

\whern@parbox 6462 \DeclareCommand\whern@parbox{%
6463 S{\leftskip\rightskip}\% horizontal alignment of resulting box (the side to
 be ragged)
6464 O{t}\% vertical alignment of parbox
6465 >is\% separator
6466 O{o,7666\textwidth}\% (3) width of parbox
6467 m\% (4) parbox contents
6468 }{%
6469 }{%
 % #1 S the skip of the ragged side,
 % #2 S the \parbox's contents.

6474 \parbox[#2]{#3}{%
6475 \IfValueTF{#1}{#1}{\leftskip}=osp\plus\textwidth
6476 \parfillskip=osp\relax
6477 \let\\\linebreak
6478 \disobeylines
6479 \whernfont\#4\unskip\strut\endgraf
6480 \getprevdepth
6481 }\% of \parbox,
6482 }\% of \whern@parbox.

\whern 6484 \def\whern{%
6485 \endgraf\nopagebreak
6486 \gm@ifstar{\wherncore}\%
6487 {\vskip\whernskip\wherncore}}}

6489 \let\whernfont\footnotesize

\whernskip 6491 \newskip\whernskip
6492 \whernskip2\baselineskip minus 2\baselineskip\relax

\whernup 6494 \DeclareCommand\whernup{%
```

```

6495 o \ a vskip before
6496 >is \ separating star (ignored)
6497 o \ (2) custom width of parbox
6498 >Pm} {\par
6499 \IfValueT{\#1}{\vskip\relax}%
6500 \leftline{%
6501 \IfValueTF{\#2}{\whern@parbox\rightskip[b][#2]}{%
6502 \whern@parbox\rightskip[b]}%
6503 {\#3}%
6504 }%
6505 \setprevdepth
6506 \nopagebreak\relax
6507 \@ifenvir{quote}{\noindent\ignorespaces}{}}

```

### Thousand separator

\thousep 6513 \pdef \thousep{\% a macro that'll put the thousand separator between every  
two three-digit groups.

First we check whether we have at least five digits.

```

6517 \gmu@thou@fiver\relax\relax\relax\relax\relax% we
 put five \relaxes after the parameter to ensure the string will
 meet \gmu@thou@fiver's definition.
6520 \gmu@thou@fiver{\#1}{% if more than five digits:
6521 \emptyify\gmu@thou@put
6522 \relaxen\gmu@thou@o\relaxen\gmu@thou@i\relaxen%
6523 \gmu@thou@ii
6524 \atempcnta\z@
6525 \gmu@thou@putter\#1\gmu@thou@putter
6526 \gmu@thou@put
6526 }

```

\gmu@thou@fiver 6528 \def \gmu@thou@fiver{\#2\#3\#4\#5\gmu@thou@fiver\#6\#7}{% this macro only
checks if the text delimited with itself consists of at least five tokens/braces
6530 \ifx\relax\relax\@xa\@firstoftwo
6531 \else\@xa\@secondoftwo
6532 \fi{\#6}{\#7}}

\gmu@thou@putter 6534 \def \gmu@thou@putter{\#2}{% we are sure to have at least five tokens before the
sentinel \gmu@thou@putter.
6536 \advance\atempcnta\@ne
6537 \atempcntb\atempcnta
6538 \divide\atempcntb\relax
6539 \atempcnta=\numexpr\atempcnta-\atempcntb\*3
6540 \edef\gmu@thou@put{\@xau{\gmu@thou@put}\unexpanded{\#1}%
6541 \ifx\gmu@thou@putter\@else
6542 \ifcase\atempcnta
6543 \gmu@thou@o\or\gmu@thou@i\or\gmu@thou@ii% all three CSes
are yet \relax so we may put them in an \edef safely.
6544 \fi
6545 \fi}% of \edef
6546 \ifx\gmu@thou@putter\@endoftext% if we are at end of the digits...
6547 \edef\gmu@tempa{%
6548 \ifcase\atempcnta
6549 \gmu@thou@o\or\gmu@thou@i\or\gmu@thou@ii
6550
6551

```

6552 \fi}%
6553 \@xa\let\gmu@tempa\gmu@thousep% ... we set the proper CS...
6554 \else% or ...
6555 \afterfi{%
6556 \gmu@thou@putter#2}%
6557 \fi% of if end of digits.
6558 }% of \gmu@thou@putter.

\gmu@thousep 6560 \def\gmu@thousep{\,}%
in Polish the recommended thousand separator is a thin space.

```

So you can type `\thousep{7123123123123}` to get 7123 123 123 123. But what if you want to apply `\thousep` to a count register or a `\numexpr`? You should write one or two `\expandafters` and `\the`. Let's do it only once for all:

```

\xathousep 6568 \pdef\xathousep#1{\@xa\thousep\@xa{\the#1}}
Now write \xathousep{\numexpr 10*9*8*7*6*120} to get 3628800.

```

```

\shortthousep 6572 \def\shortthousep{%
\thous 6573 \DeclareCommand\thous{Q{+-0123456789}}{%
we declare it as a command with Q-type argument to allow spaces between digits.

```

```

6576 \ifmmode\hbox\bgroup\gmu@mmhboxtrue\fi
6577 \IfValueTF{\#1}{%
6578 we are given a sequence of digits
6579 \tempcnta=\#1\relax
6580 \ifnum\tempcnta<0\$\$-
6581 \tempcnta=-\tempcnta
6582 \fi
6583 \xathousep\tempcnta
6584 \if@gmu@mmhbox\egroup
6585 \else\@xa\spifletter
6586 \fi
6587 }%
6588 {%
6589 no bare digits given, then we assume the argument is braced.
6590 \thousep
6591 }%
6592 }%
6593 }%
6594 }%
6595 of \thous.
6596 }%
6597 of \shortthousep.

```

And now write `\thous 3628800` to get 3628800 even with a blank space (beware of the range of `\TeX`'s counts).

#### `hyperref's \nolinkurl into \url*`

```

\urladdstar 6599 \def\urladdstar{%
6600 \AtBeginDocument{%
6601 \ifpackageloaded{hyperref}{%
6602 \StoreMacro\url
6603 \pdef\url{\gmu@ifstar{\nolinkurl}{\storedcsname{url}}}{%
6604 }{}}
6605 \onlypreamble\urladdstar

```

#### `Footnotes suggested by Andrzej Tomaszewski`

```

\ATfootnotes 6611 \DeclareCommand\ATfootnotes{s}{%
We make the footnote mark in the footnote \scriptsize not \scriptscriptsize.

```

```

6617 \IfValueT{#1}{ the following setting is suitable for old style numbers in foot-
 note marks, therefore I place it in the starred version of the command.
6620 {\prependtomacro\gmu@ATfootnotes{%
6621 \pdef{\@makefnmark}{%
6622 \mbox{\normalfont\textsuperscript{\smaller[3]{%
6623 \@thefnmark}}}}%
6624 }% of prepend,
6625 }% of \IfValueT.
6626 \gmu@ATfootnotes
6627 \gmu@AT@ampulex\maketitle% without hyperref
6628 \ifdefined\HyOrg@maketitle
6629 \afterfi{\gmu@AT@ampulex\HyOrg@maketitle}% with hyperref
6630 \fi
6631 }

\gmu@AT@ampulex 6633 \pdef{\gmu@AT@ampulex#1}{%
6634 \ampulexdef#1{\def{\@makefnmark}{%
6635 \if@twocolumn
6636 {\gmu@ATfootnotes\if@twocolumn}{ Ampulex redefinition of \maketitle for the standard classes.
6637 \ampulexdef#1{\long\def{\@makefntext}{%
6638 \if@twocolumn{\gmu@ATfootnotes\if@twocolumn}{ Ampulex redefinition of \maketitle for mwcls.
6639 }%
6640 }%
6641 }}

\gmu@ATfootnotes 6643 \pdef{\gmu@ATfootnotes}{%
6644 \long\pdef{\@makefntext##1}{%
6645 \ifdefined\@parindent\@parindent\@parindent
6646 \else\@parindent\@em\relax
6647 \fi
6648 \indent{\ATf@font\scriptsize%
6649 {\@thefnmark}}%
6650 \gmu@fnhook
6651 \enspace\ignorespaces##1}%
6652 }%
6653 \let\ATf@font\normalfont
6654 \empty\gmu@fnhook
6655 }

\rrthis 6657 \pdef{\rrthis}{ rag right this': make only the current paragraph ragged right
6658 (e.g. if the paragraph consists of a long URL).
6659 \begingroup\rightskip=0sp\plus\hspace\endgraf\endgroup}

\centerthis 6666 \pdef{\centerthis}{ 2009/12/15
6667 \begingroup
6668 \rightskip=1\rightskip\plus\hspace
6669 \leftskip=1\leftskip\plus\hspace
6670 \parfillskip=\z@skip
6671 \endgraf\endgroup}

\balsmiley 6674 \def{\balsmiley#1}{ }% to balance parentheses and brackets in smileys. ;)

```

```

% \balsmiley(;-) .

\scantnoline 6681 \long\pdef\scantnoline#1{%
 'rescan tokens without adding line end'
 {\endlinechar\m@ne\scantokens{#1}}}

```

### A fix to the url package

It happened that a URLs typeset with the \url command of the url package came out sort of spaced because kerning was off because of the math mode. So I provide a redefinition of the internal macros of the url package which in my version uses not math mode but \scantokens and not \relpenalty and \binoppenalty but \hyphenpenalty (as it is in the paragraph) and \discretionary. I tried putting explicit penalties after the symbols but that spoiled kerning.

The rules of line breaking are somewhat different, too: in the original url package line breaks are forbidden between any two symbols listed in \UrlBigBreaks. In my version line breaks are forbidden between any two *identical* 'URL Breaks' and 'URL Big Breaks'.

There are some more differences in formatting some chars, i.a. ~, % and angle brackets which I don't treat specially and just take from font assuming the font provides ASCII chars and checking whether it provides the angle brackets.

```

6707 \@ifXeTeX{%
\UrlFix 6708 \pdef\UrlFix{\AtBeginDocument{%
6709 \@ifpackageloaded{url}{\gm@UrlFix}{}{}}%
6710 \relaxen\UrlFix}%
6712 \AtBeginDocument{%
\UrlFix 6713 \pdef\UrlFix{%
6714 \@ifpackageloaded{url}{\gm@UrlFix}{}{}}%
6715 \relaxen\UrlFix}%
6716 }
6717 {%
\UrlFix 6718 \pdef\UrlFix{\PackageWarning{gmu}{!!! The \string%
6719 \UrlFix\space
6720 declaration works only with XeTeX}}%
6721 }

6723 \@ifXeTeX{}{%
6724 \edef\gmu@restoreUpUpUp{\catcode`\\nx\^{\the\catcode`%
6725 \AtEndOfPackage\gmu@restoreUpUpUp
6726 \catcode`\\nx=9}%
\gm@UrlFix 6728 \def\gm@UrlFix{%
 default style assignments

\UrlBreaks 6731 \def\UrlBreaks{\do\.\do@\do\\\do/\do\!\do_\\do\|\do\%;%
 \do\]%
6732 \do\)\do\,\do\?\do\'\do\"\\do\+\do\=\do\#\do\%\do\~\do\%
 _\do\|%
6733 \do\{\do\}\do\$}%
\UrlBigBreaks 6734 \def\UrlBigBreaks{\do\:\;}%
\UrlNoBreaks 6735 \def\UrlNoBreaks{\do\\(\do\[\\do\{}\;}%
\UrlSpecials 6736 \def\UrlSpecials{%
6737 \do\{\hbox{\visible{}}\}\do\^\^M{\hbox{%
 \visible{}}}\}%

```

```

\Url@Format 6741 \def\Url@Format##1{%
6742 \UrlFont
6743 \ifdefined\verbatim@specials
6744 \catcode`>\active
6745 \verb@specials
6746 \verb@mathhack
6747 \fi % setting of the escape char, begin and end group and optionally math
 shift, defined in gmverb.
6748 \gm@UrlSetup
6749 \UrlLeft
6750 \edef\gmu@theendlinechar{\the\endlinechar}%
6751 \endlinechar\m@ne
6752 \kern\z@% to forbid hyphenating the first word if the URL begins with
6753 a word
6754 \hyphenchar\font=\UrlHyphenchar\relax
6755 \let\-\gmu@discretionaryhyphen
6756 \scantokens{\##1}%
6757 \endlinechar\gmu@theendlinechar\relax
6758 \UrlRight
6759 }% of \Url@Format.

6760 \edef\UrlHyphenchar{%
6761 \ifdefined\gmv@hyphenchar\gmv@hyphenchar
6762 \else"A6\fi}% broken bar, | or the same as provided in gmverb for verba-
6763 tims. You can redefine it as you please. This char is used as the hyphen-
6764 ation char in URLs and therefore should be different from – (hyphen),
6765 which is often a part of an URL. The broken bar seems to be quite unlikely
6766 in URLs and/or file names.

\verbatim@mathhack 6773 \def\verbatim@mathhack{%
6774 \ifdefined\verbatim@specials@list
6775 \@xa\verbatim@mathhack@\verbatim@specials@list
6776 \fi
6777 }%

\verbatim@mathhack@ 6779 \def\verbatim@mathhack@##1##2##3##4##5##6{%
6780 \IfValueT{##4}{%
6781 \edef\gmu@thinmuskip{\the\thinmuskip}%
6782 \edef\gmu@medmuskip{\the\medmuskip}%
6783 \edef\gmu@thickmuskip{\the\thickmuskip}%
6784 \begingroup
6785 \lccode`\~=\##4\lowercase{%
6786 \endgroup\def~##1~}%
6787 {$\thinmuskip\gmu@thinmuskip\relax
6788 \medmuskip\gmu@medmuskip\relax
6789 \thickmuskip\gmu@thickmuskip\relax
6790 ##1%
6791 $}%
6792 \catcode`##4\active
6793 }%
6794 }%

\gm@UrlSetup 6796 \def\gm@UrlSetup{%
6797 \medmuskip\Urulemuskip\thickmuskip\medmuskip\thinnmuskip\thinnmuskip%

```

```

6798 \relpenalty\UrlBigBreakPenalty\binoppenalty%
 \UrlBreakPenalty
6801 \def\do{\gmu@doUrlMath\UrlBreakPenalty}\UrlBreaks% bin(\hy-
 phenpenalty anyway)
6803 \def\do{\gmu@doUrlMath\UrlBigBreakPenalty}\UrlBigBreaks% rel
 (\hyphenpenalty anyway)
6805 \def\do{\gmu@doUrlMath@M}\UrlNoBreaks% open (no break)
6806 \def\do{\gmu@doUrlMathAc\UrlBreakPenalty}(\hyphenpenalty)
6807 \UrlSpecials
6808 \if\iffontchar\font"2329\else\fi\iffontchar%
 \font"232A\else\fi
we check whether the font provides both left and right angle brackets.
6811 \gmu@measurewd{^^^2329}%
6812 \edef\gmu@tempa{%
6813 @nx\gmu@doUrlMathAc@M@nx<{%
6814 \hbox\to\gmu@tempb{\unexpanded{\hss\char"2329\hss}}%
6815 }\gmu@tempa
6816 \gmu@measurewd{^^^232A}%
6817 \edef\gmu@tempa{%
6818 @nx\do\@nx>{%
6819 \hbox\to\gmu@tempb{\unexpanded{\hss\char"232A\hss}}%
6820 }\gmu@tempa
6821 \else
6822 \gmu@doUrlMathAc@M<{\langle}\do\>{\rangle}%
6823 \fi
6824 \iffontchar\font"22C6% low star
6825 \do*\hbox{\char"22C6}%
6826 \else\do**%
6827 \fi
6828 \ifx\do@url@hyp@\empty
6829 \gmu@measurewd{-}% this macro is defined in line 5197.
6830 \edef\gmu@tempa{%
6831 \unexpanded{\gmu@doUrlMathAc@M-}%
6832 {\hbox\to\gmu@tempb{\unexpanded{\hss-\hss}}%
6833 @nx\-% hyphen is a good point for hyphenation, but the hyphen-
 ation char should be sth. else, and it is indeed: | (broken bar,
 \char"A6). See also line 6765
6834 }\gmu@tempa
6835 \fi
6836 \addfontfeature{Ligatures=NoCommon, Mapping=none}%
 instead of
 'doing' \verb@im@nolig@list.
6839 }% of \gm@UrlSetup.

\gmu@doUrlMath 6849 \def\gmu@doUrlMath##1##2{%
 % #1 value of the penalty (used as a Boolean: if < 10 000,
 % \hyphenpenalty will be used anyway, if ≥ 10 000, there will be no
 % \discretionary),
 % #2 the char, given as \langle char \rangle.
6850 \begingroup
6851 \lccode`\~`##2\lowercase{%
6852 \endgroup\def~{\@ifnextchar~}%

```

```

6864 \@xa\addtomacro{\@xa~} of \lowercase.
6865 \ifnum##1<\@M
6866 {%
6867 {\char`##2\csname\gmu@dbl\string##2kern\endcsname} if next
6868 is the same char
6869 {\ifmmode\char`##2% else
6870 \else\gmu@urlbreakable{##1}{##2}%
6871 \fi}%
6872 }% of \addtomacro's argument \ifnum true.
6873 \else
6874 {%
6875 {\char`##2\csname\gmu@dbl\string##2kern\endcsname} {%
6876 \char`##2}%
6877 }% of \addtomacro's argument \ifnum false.
6878 \fi
6879 \catcode`##2=\active
6880 }% of \gmu@doUrlMath.

\gmu@doUrlMathAc 6880 \def\gmu@doUrlMathAc##1##2##3{%
 % #1 (value of) a penalty (see the remark to ##1 of the previous macro),
 % #2 the char (as \char),
 % #3 the definition.
 \begingroup
 \lccode`\~=`##2\lowercase{%
 \endgroup\def~{\@ifnextchar~{%
 \gmu@addtomacro{\@xa~} of \lowercase.
 \ifnum\##1<\@M
 {%
 {\ifmmode\char`##2\else$##3\m@th$\fi}%
 {\ifmmode\char`##2%
 \else\discretionary{\hbox{$##3\m@th$}}{}{\hbox{$##3$}
 \m@th$}}%
 \fi}%
 }% of \addtomacro's argument if num true.
 \else
 {%
 {\ifmmode\char`##2\else$##3\m@th$\fi} {\ifmmode%
 \char`##2\else$##3\m@th$\fi}%
 }% of \addtomacro's argument if num false.
 \fi
 \catcode`##2=\active
 }% of \gmu@doUrlMathAc.

\gmu@url@rigidbreak 6906 \pdef\gmu@url@rigidbreak##1##2{\discretionary{\char`##2}{%
 }{\char`##2}%
\gmu@url@flexbreak 6908 \pdef\gmu@url@flexbreak##1##2{\penalty\@M\hskip\z@\u
 pluso,03em
 \char`##2\penalty##1\hskip\z@\u pluso,03em\relax}%
6909
6911 \let\gmu@urlbreakable\gmu@url@flexbreak
\url@z 6913 \def\Url@z##1{%
 Do any hyper referencing due to hyperref (or perform a url-def)
 6915 \Url@HyperHook

```

Now do the formatting in a group (can also have \Url@HyperHook take this as an argument).

```

6918 {\Url@Format {##1}}%
6919 \endgroup}%
6921 \DeclareUrlCommand\file{\urlstyle{sf}}%
6923 \emptyify\Url@moving% with our settings \url is pretty allowed in moving
 arguments, I hope.
6925 }% of \gmu\UrlFix.

\UrlSlashKern 6927 \DeclareCommand\UrlSlashKern{0{tt}m}%
6928 {\AtBeginDocument{%
6929 \nameedef{\url#1style}{\def\@nx\UrlFont{%
6930 \@xa\@nx\csname#1family\endcsname
6931 \def\@xa\@nx\csname\gmu\dbl\string\kern\endcsname
6932 {\kern#2\relax}%
6933 }% of \UrlFont
6935 }% of \url#1style
6936 \urlstyle{#1}%
6937 }% of \AtBeginDocument
6938 }% of \UrlSlashKern

```

### Conditional tilde

Polish typesetting standards say that for 12 dd and 10 dd *[czcionki](#)* if leading is narrower than  $3\frac{1}{2}$  *[kwadratu](#)*,  $3\frac{1}{2} \times 48$  dd, then hanging letters are allowed, which also applies to 8 and 6 dd *[czcionki](#)* in less than 3 *[kwadraty](#)* leading. I treat this recommendation not strictly but as an inspiration, that is I translate »dd« to »pt«.

```

\TrzaskaTilde 6952 \def\TrzaskaTilde{%
6953 \@xa\DeclareCommand\@xa\gm@smarttilde
6954 \@xa{\@xa\S\@xa{\all@stars~}}{%
6955 \IfValueTF{##1}{\nobreakspace{}{}}{%
6956 \ifdim\dimexpr\hsize-\leftskip-\rightskip
6957 -\ifdim\hangindent<\z@\fi\hangindent\% the last parameter is
 used with respect to the floatfl package.
6959 >%
6960 \ifdim\f@size\pt>\dimexpr\opt-1sp\relax
6961 168dd
6962 \else
6963 144dd
6964 \fi
6965 \nobreakspace\%
6966 \else
6967 \%
6968 \fi
6969 }% of \gm@ifstar's else,
6970 }% of \gm@smarttilde,
6971 \let~\gm@smarttilde
6972 }% of \TrzaskaTilde.

\getprevdepth 6975 \pdef\getprevdepth{%
6976 \endgraf
6977 \xdef\setprevdepth{\prevdepth=\the\prevdepth\relax}%

```

6978 }

### Storing the catcode of line end

```
\StoreCatM 6981 \def\StoreCatM{%
 6982 \protected\edef\RestoreCatM{%
 6983 \catcode`\\@nx\^^M=\the\catcode`\^^M\relax}%
 6984 }

\RestoreCatM 6986 \pdef\RestoreCatM{\PackageE{gmu}{first store the catcode
 6987 of
 6988 ^\empty^\empty M with \string\StoreCatM.}%
 6988 }
```

### A really empty page

Copied from Marcin Woliński's macros.

```
\clearemptydoublepage 6994 \newcommand{\clearemptydoublepage}{%
 6995 \newpage{\pagestyle{empty}\cleardoublepage}%

\setspacekip 6998 \DeclareCommand\setspacekip{%
 6999 A{}% optional factor for all three components
 7000 O{\fontdimen2\font}%
 7001 >is
 7002 O{\fontdimen3\font}%
 7003 >is
 7004 O{\fontdimen4\font}%
 7005 {\spaceskip=#1#2plus#1#3minus#1#4\relax}

 7007 \foone\obeylines{%
\disobeylines 7008 \def\disobeylines{%
 7009 for arguments in which line end is active to simulate
 7010 normal behaviour
 7011 \ifnum\catcode`\^^M=\active%
 7012 \pdef^\^^M{\ifnextgroup{\ifhmode\unskip\space\fi}{%
 7013 \gmu@disMiner}}%
\gmu@disMinner 7012 \def\gmu@disMinner##1{%
 7013 \ifx^\^^M##1\endgraf%
 7014 \else\afterfi{\ifhmode\unskip\space\fi}\fi##1}%
 7015 }%
 7016 }

\makestarlow 7022 \def\makestarlow{%
 7023 \begingroup\lccode`\\~=`*\lowercase{%
 7024 \endgroup\def~{\gmu@lowstar}}% 2009/10/19 \let changed to \def
 * 7024 to allow redefinitions of \gmu@lowstar.
 7026 \catcode`*=active
 7027 \defLowStarFake
 7028 }

\defLowStarFake 7030 \DeclareCommand\defLowStarFake{%
 7031 Q{+-0123456789,.}{0,5}% fraction of fontchar depth of the star glyph
 7032 }%
 7033 {%
```

```

\gmu@lowstarfake 7034 \def\gmu@lowstarfake{%
7035 \leavevmode\vbox{\hbox{*}\kern#1\fontchardp\font`*}%
7036 }%
7037 }

\gmu@lowstarfake 7040 \def\gmu@lowstarfake{*} % useful for next command where normal star is
 low.

```

The `\*` CS and active `*` should be defined different to make them distinguishable by tests, especially with `\gm@ifstar` in mind.

```

* 7048 \DeclareCommand*\Q{0123456789}{1}
7050 {\gmu@star@loopo{\#1}}

```

```

\gmu@star@loop 7052 \def\gmu@star@loop#1#2{%
 this is an expandable loop as in The ϵ -TEX Manual
 p. 9.
7054 \ifnum#1<\numexpr#2\relax%
7055 \gmu@lowstar
7056 \@xa\gmu@star@loop
7057 \@xa{\number\numexpr#1+1\@xa}%
7058 \@xa{\number#2\@xa}%
7059 \fi}
7063 \endinput

```

## d. The gmiflink Package<sup>1</sup>

Written by Grzegorz ‘Natrór’ Murzynowski,  
natror at o2 dot pl

© 2005, 2006 by Grzegorz ‘Natrór’ Murzynowski.

This program is subject to the LATEX Project Public License.

See <http://www.ctan.org/tex-archive/help/Catalogue/licenses.lppl.html> for the details of that license.

LPPL status: “author-maintained”.

```
45 \NeedsTeXFormat {LaTeX2e}
46 \ProvidesPackage {gmiflink}
47 [2006/08/16 v0.97 Conditionally hyperlinking package
 (GM)]
```

### Introduction, usage

This package protects you against an error when a link is dangling and typesets some plain text instead of a hyperlink then. It is intended for use with the hyperref package. Needs two LATEX runs.

I used it for typesetting the names of the objects in a documentation of a computer program. If the object had been defined a \hyperlink to its definition was made, otherwise a plain object’s name was typeset. I also use this package in authomatic making of hyperlinking indexes.

The package provides the macros \gmiflink, \gmiref and \gmihypertarget for conditional making of hyperlinks in your document.

\gmihypertarget [*<name>*] {*<text>*} makes a \hypertarget {@name} {*<text>*} and a \label {@name}.

\gmiflink [*<name>*] {*<text>*} makes a \hyperlink {@name} {*<text>*} to a proper hypertarget if the corresponding *label* exists, otherwise it typesets *<text>*.

\gmiref [*<name>*] {*<text>*} makes a (hyper-) \ref {@name} to the given label if the label exists, otherwise it typesets *<text>*.

The *@name* argument is just *<name>* if the *<name>* is given, otherwise it’s *<text>* in all three macros.

For the example(s) of use, examine the gmiflink.sty file, lines 45–58.

The remarks about installation and compiling of the documentation are analogous to those in the chapter gmdoc.sty and therefore omitted.

### Contents of the gmiflink.zip archive

The distribution of the gmiflink package consists of the following three files and a TDS-compliant archive.

---

<sup>1</sup> This file has version number v0.97 dated 2006/08/16.

```

gmiflink.sty
README
gmiflink.pdf
gmiflink.tds.zip

```

## The code

```

145 \@ifpackageloaded{hyperref} {} {\message{^^J^^J gmiflink
 package:
146 There's no use of me without hyperref package, I end
 my input.^^J} \endinput}

148 \providecommand{\empty}{}

A new counter, just in case

GMlabel 150 \newcounter{GMlabel}
151 \setcounter{GMlabel}{0}

```

The macro given below creates both hypertarget and hyperlabel, so that you may reference both ways: via `\hyperlink` and via `\ref`. Its pattern is the `\label` macro, see `LATEX Sourceze`, file `x`, line 32.

But we don't want to gobble spaces before and after. First argument will be a name of the hypertarget, by default the same as typeset text, i.e., argument #2.

```

\gmhypertarget 161 \DeclareRobustCommand*\gmhypertarget{%
162 \@ifnextchar{}{\gmahypertarget}{\@dblarg{%
 \gmahypertarget}}}

\gmahypertarget 165 \def\gmahypertarget[#1]{% If argument #1 = \empty, then we'll use #2,
 i.e., the same as name of hypertarget.
166 \refstepcounter{GMlabel}% we \label{\gmht@firstpar}
167 \hypertarget{#1}{#2}%
168 \protected@write\auxout{}{%
169 \string\newlabel{#1}{#2}\the\page\relax{GMlabel.%%
170 \arabic{GMlabel}}}}%
171 }% end of \gmahypertarget.
172 }% end of \gmhypertarget.

```

We define a macro such that if the target exists, it makes `\ref`, else it typesets ordinary text.

```

\gmiref 178 \DeclareRobustCommand*\gmiref{\@ifnextchar{}{\gmairef}{%
179 \@dblarg{\gmairef}}}

\gmairef 181 \def\gmairef[#1]{%
182 \expandafter\ifx\csname r@#1\endcsname\relax\relax%
183 #2\else\ref{#1}\fi%
184 }% end of \gmairef

\gmiflink 187 \DeclareRobustCommand*\gmiflink{\@ifnextchar{}{\gmaiflink}{%
188 \@dblarg{\gmaiflink}}}

\gmaiflink 190 \def\gmaiflink[#1]{%
191 \expandafter\ifx\csname r@#1\endcsname\relax\relax%
192 #2\else\hyperlink{#1}{#2}\fi%
193 }% end of \gmaiflink

```

It's robust because when just `\newcommand*`ed, use of `\gmiflink` in an indexing macro resulted in errors: `\@ifnextchar` has to be `\noexpanded` in `\edefs`.

<sup>199</sup> \end{input}

The old version — all three were this way primarily.

```
\newcommand*\gmiflink[2][\empty] {{%
 \def\gmht@test{\empty}\def\gmht@firstpar{\#1}%
 \ifx\gmht@test\gmht@firstpar\def\gmht@firstpar{\#2}\fi%
 \expandafter\ifx\csname
 r@\gmht@firstpar\endcsname\relax\relax%
 #2\else\hyperlink{\gmht@firstpar}{#2}\fi%
} }
```

## e. The gmverb Package<sup>1</sup>

March 4, 2010

This is (a documentation of) file gmverb.sty, intended to be used with L<sup>A</sup>T<sub>E</sub>X 2<sub>E</sub> as a package for a slight redefinition of the \verb macro and verbatim environment and for short verb marking such as | \mymacro | .

Written by Natror (Grzegorz Murzynowski),  
natror at o2 dot pl

© 2005, 2006, 2007, 2008, 2009, 2010 by Natror (Grzegorz Murzynowski).

This program is subject to the L<sup>A</sup>T<sub>E</sub>X Project Public License.

See [http://www.ctan.org/tex--archive/help/Catalogue/licenses.lppl.html](http://www.ctan.org/tex-archive/help/Catalogue/licenses.lppl.html) for the details of that license.

LPPL status: "author-maintained".

Many thanks to my T<sub>E</sub>X Guru Marcin Woliński for his T<sub>E</sub>Xnical support.

```
79 \NeedsTeXFormat {LaTeX2e}
80 \ProvidesPackage{gmverb}
81 [2010/03/04 v0.93 After shortvrb (FM) but my way (GM)]
```

### Intro, usage

This package redefines the \verb command and the verbatim environment so that the verbatim text can break into lines, with % (or another character chosen to be the comment char) as a 'hyphen'. Moreover, it allows the user to define their own verbatim-like environments provided their contents would be not *horribly* long (as long as a macro's argument may be at most).

This package also allows the user to declare a chosen char(s) as a 'short verb' e.g., to write | \a \verbatim \example | instead of \verb| \a \verbatim \example | .

The gmverb package redefines the \verb command and the verbatim environment in such a way that , { and \ are breakable, the first with no 'hyphen' and the other two with the comment char as a hyphen. I.e. {<subsequent text>} breaks into { % <subsequent text>} and <text>\mymacro breaks into <text>% \mymacro.

\nobreakbslash  
\nobreaklbrace  
(If you don't like line breaking at backslash, there's the \nobreakbslash declaration (observing the common scoping rules, hence OCSR) and an analogous declaration for the left brace: \nobreaklbrace.)

\VerbHyphen  
The default 'hyphen' is % since it's the default comment char. If you wish another char to appear at the line break, use the \VerbHyphen declaration that takes \<char> as the only argument. This declaration is always global.

\verbeolOK  
Another difference is the \verbeolOK declaration (OCSR). Within its scope, \verb allows an end of a line in its argument and typesets it just as a space.

---

<sup>1</sup> This file has version number v0.93 dated 2010/03/04.

|                                                                                                         |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                           |
|---------------------------------------------------------------------------------------------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
|                                                                                                         | As in the standard version(s), the plain <code>\verb</code> typesets the spaces blank and <code>\verb*</code> makes them visible.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                         |
| <code>\MakeShortVerb</code>                                                                             | Moreover, <code>gmverb</code> provides the <code>\MakeShortVerb</code> macro that takes a one-char control sequence as the only argument and turns the char used into a short verbatim delimiter, e.g., after <code>\MakeShortVerb* \  </code> (as you guess, the declaration has its starred version, which is for visible spaces, and the non-starred for the spaces blank) you may type <code>  \mymacro </code> to get <code>\mymacro</code> instead of typing <code>\verb+ \mymacro+</code> . Because the char used in this example is my favourite and used just this way by DEK in the <i>The T<sub>E</sub>Xbook's</i> format, <code>gmverb</code> provides a macro <code>\dekclubs</code> as a shorthand for <code>\MakeShortVerb(*) \  </code> . |
| <code>\DeleteShortVerb</code>                                                                           | Be careful because such active chars may interfere with other things, e.g. <code> </code> with the <code>tikz</code> package. If this happens, you can declare <code>\DeleteShortVerb \  </code> and the previous meaning of the char used shall be restored.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                             |
| <code>\OldMakeShortVerb</code>                                                                          | One more difference between <code>gmverb</code> and <code>shortverb</code> is that the chars <code>\active</code> ated by <code>\MakeShortVerb</code> in the math mode behave as if they were 'other', so you may type e.g., <code>\$2   o \$</code> to get <code>2o</code> and <code>+ \active</code> ated this way is in the math mode typeset properly etc.                                                                                                                                                                                                                                                                                                                                                                                            |
| <code>\dekclubs</code><br><code>\dekclubs*</code><br><code>\olddekclubs</code><br><code>\edverbs</code> | However, if you don't like such a conditional behaviour, you may use <code>\Old\MakeShortVerb</code> instead, what I do when I like to display short verbatims in display-math.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                           |
|                                                                                                         | There's one more declaration provided by <code>gmverb</code> : <code>\dekclubs</code> , which is a shorthand for <code>\MakeShortVerb \  </code> , <code>\dekclubs*</code> for <code>\MakeShortVerb* \  </code> and <code>\old\dekclubs</code> for <code>\OldMakeShortVerb \  </code> .                                                                                                                                                                                                                                                                                                                                                                                                                                                                   |
|                                                                                                         | There's one more declaration, <code>\edverbs</code> that makes <code>\ [</code> checks if the next token is an active char and opens an <code>\hbox</code> if so. That is done so that you can write (in <code>\edverbs'</code> and <code>\dekclubs'</code> scope)                                                                                                                                                                                                                                                                                                                                                                                                                                                                                        |
|                                                                                                         | <code>\[   &lt;verbatim stuff&gt;   \]</code>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                             |
|                                                                                                         | instead of                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                |
|                                                                                                         | <code>\[ \hbox{   &lt;verbatim stuff&gt;   } \]</code>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                    |

to get a displayed shortverb.

Both versions of `\dekclubs` OCSR.

The `verbatim` environment inserts `\topsep` before and after itself, just as in standard version (as if it was a `list`).

In August 2008 Will Robertson suggested grey visible spaces for `gmdoc`. I added a respective option to `gmdoc` but I find them so nice that I want to make them available for all verbatim environments so I bring here the declaration `\VisSpacesGrey`. It redefines only the visible spaces so affects `\verb*` and `verbatim*` and not the unstarred versions. The colour of the visible spaces is named `visspacesgrey` and you can redefine it `xcolor` way.

We also provide the `\verbatimspecials` declaration that takes six arguments:

- #1 m a char for verbatim escape char (for catcode 0), has to be unbraced<sup>2</sup>,
- #2 m a char for group starter (for catcode 1), has to be unbraced,
- #3 m a char for group ender (for catcode 2), has to be unbraced,
- [#4] (optional) a char for verbatim math shift (for catcode 3); it has to be in square brackets if present. If absent, nothing is set for the verbatim math shift,
- [#5] (optional) a char for the shorthand for `\metachar`; it has to be in square brackets if present.

---

<sup>2</sup> To be precise, the arguments cannot be wrapped in curly braces because those are recatcoded to 'other'. But if you make some other pair of chars category 1 and 2 that are not on the `\dospecials` list, then you can wrap the arguments in those chars, but what for?

{#6} b optional in curly braces, additional stuff (commands) to be executed in a verbatim.

For example, after telling TeX

```
\verb@imspecials@{<`<>`[`>`]{\def\|{\vert}}}
```

(the slash is Unicode Fractional Slash, spaces are ignored) you can write

```
| \macro{arg<arg. {n+1}>}|[No]Value|(T|F)|
```

to get

```
\macro{<arg. n+1>}|[No]Value(T|F)
```

To get a ‘verbatim special’ verbatim, precede it with the ‘verbatim escape’: //`<`>`[`>` in the example above.

Note also that `|` is a control sequence so it doesn’t delimit the short verbatim |’s argument.

The \verb@imspecials@ declaration OCSR. Subsequent uses of it override the previous settings. If you specified the optionals at first and then specify \verb@im!specials@ without optionals, the previous optional settings are forgotten.

To turn the ‘verbatim specials’ off write \verb@noverbatimspecials@, which OCSR too.

Note that although we don’t provide a ‘verbatim superscript’ nor ‘verbatim subscript’, you have the \verb@sups@ and \verb@subs@ CS’es defined by gutils.

The 4th argument for the math shift is optional because you can use LATEX’s \(` and \)`.

The \verb@imspecials@ declaration goes a step further than LATEX’s alltt and Til Tantau’s beamer’s semiverbatim environments. To get their effect, declare

```
\verb@imspecials@{}
```

There is something for verbatims in arguments of commands: \verb@scanverb[@]{<text>}@. However there are some limitations: if % is the comment char (which is usual situation), then you cannot use % in <text>, or rather, % will act as comment char anyway. Moreover, spaces are ignored. This last limitation may be worked around if you declare \verb@im!timspecials@, say `/ (fraction slash) as the escape char. Then you can use `/ to put a space which will be typeset blank in the unstarred version and visible with star.

Not so long ago I started to use the ‘broken bar’ (U+00A6, |) character as a hyphen in hyperlinks, because it seems not to occur in hyperlinks at all unlike hyphen. I suggest the same char for verbatims, but if you don’t like it, there’s the \verb@DiscretionaryHyphen@ declaration that takes two arguments. Broken bar is declared as

```
\verb@DiscretionaryHyphen@{"A6}{|}
```

### The package options

As many good packages, this also does not support any options.

The remarks about installation and compiling of the documentation are analogous to those in the chapter gmdoc.sty and therefore omitted.

### Contents of the gmverb.zip archive

The distribution of the gmverb package consists of the following three files and a TDS-compliant archive.

```
gmverb.sty
README
```

gmverb.pdf  
gmverb.tds.zip

This package requires another package of mine, `gmutils`, also available on CTAN.

## The code

### Preliminaries

331 \RequirePackage{gmutils}[2008/10/08]

For `\firstofone`, `\afterfi`, `\gmobeyspaces`, `\@ifnextcat`, `\foone` and `\noexpand`'s and `\expandafter`'s shorthands `\@nx` and `\@xa` resp.

Someone may want to use another char for comment, but we assume here 'orthodoxy'. Other assumptions in `gmdoc` are made. The 'knowledge' what char is the comment char is used to put proper 'hyphen' when a `verbatim` line is broken.

\verbhyphen 345 \let\verbhyphen\xiipercent

Provide a declaration for easy changing it. Its argument should be of `\<char>` form (a `\<char>`<sub>12</sub> is also allowed).

\VerbHyphen 351 \def\VerbHyphen#1{  
352 {\escapechar\m@ne  
353 \@xa\gdef\@xa\verbhyphen\@xa{\string#1}}}

As you see, it's always global.

### The breakables

Let's define a `\discretionary` left brace such that if it breaks, it turns `{%` at the end of line. We'll use it in almost Knuthian `\ttverbatim`—it's part of this 'almost'.

\breakbrace 362 \def\breakbrace{  
363 \discretionary{\type@lbrace\verbhyphen}{}{\type@lbrace}{%  
364 \yeshy}  
  
367 \foone{\catcode`\[=1\catcode`\{=\active\catcode`\]=2\} %  
368 [%  
369 \def\dobreakbrace[\catcode`\{=\active  
370 \def{  
371 [\breakbrace\gm@lbracehook]]%  
372 ]

Now we only initialise the hook. Real use of it will be made in `gmdoc`.

376 \relaxen\gm@lbracehook

The `\bslash` macro defined below I use also in more 'normal' TEXing, e.g., to `\typeout` some `\outer` macro's name.

381 \foone{\catcode`\!=o\@makeother\\}%  
382 {  
383 !def!\bslash{}%  
384 }% of \foone.  
  
\breakbslash 387 \def\breakbslash{  
388 \discretionary{\verbhyphen}{%

```

389 {\type@bslash} {\type@bslash} \yeshy% it seems that we allow hyphen-
 ation after backslash but hyphenation will be allowed iff \hyphenchar
 % \font is nonnegative.
392 }% of \breakbslash.

```

Sometimes line breaking at a backslash may be unwelcome. The basic case, when the first CS in a verbatim breaks at the line end leaving there %, is covered by line 801. For the others let's give the user a counter-crank:

```

\nobreakbslash
\breakbslash
398 \pdef\nobreakbslash{\def\breakbslash{\type@bslash\yeshy}}%
 due to the common scoping rules. But for the special case of a backslash opening a verbatim scope, we deal specially in the line 801.

```

Analogously, let's provide a possibility of 'nobreaking' the left brace:

```

\nobreakbrace
\breakbrace
405 \pdef\nobreakbrace{\def\breakbrace{\type@lbrace\yeshy}}
410 \foone{\catcode`!=\o\catcode`\=\active}%
412 {%
413 !def!dobreakbslash{!catcode`!=\active\!def\{%
 !breakbslash}}%
414 }

```

The macros defined below, \visiblebreakspaces and \xiiclus we'll use in the almost Knuthian macro making verbatim. This 'almost' makes a difference.

```

\breakablevisspace
423 \def\breakablevisspace{\discretionary{\visiblespace}{\visiblespace}{%
 \visiblespace}}

```

The \visiblespace macro is \let in gutils to \xiispace or to \xxt@visiblespace of xltextra if available.

```

427 \foone\obeyspaces% it's just re\catcode'ing.
428 {%
429 \newcommand*\dobreakvisible{\def\breakablevisspace{\obeyspaces}%
 \obeyspaces}% \defining it caused a stack overflow disaster with
 gmdoc.

```

```

\dobreakblankspace
431 \newcommand*\dobreakblankspace{\let\space=\obeyspaces}%
432 {%
435 \foone{@makeother\|}{%
436 \def\xiiclus{}}

```

### Almost-Knuthian \ttverbatim

\ttverbatim comes from *The T<sub>E</sub>Xbook* too, but I add into it a L<sup>A</sup>T<sub>E</sub>X macro changing the \catcodes and make spaces visible and breakable and left braces too.

```

\ttverbatim
445 \pdef\ttverbatim{%
446 \let\do=\do@noligs\verbatim@nolig@list
447 \let\do=@makeother\dospecials
448 \dobreakbrace\dobreakbslash
449 \dobreakspace
450 \makeatletter
454 \ifhmode
455 \setspacekip
456 \fi
457 \tt
458 \edef\gmv@storedhyphenchar{\the\hyphenchar\font}%

```

```

459 \hyphenchar\font=\gmv@hyphenchar
460 \ttverbatim@hook}

```

While typesetting stuff in the QX fontencoding I noticed there were no spaces in verbatims. That was because the QX encoding doesn't have any reasonable char at position 32. So we provide a hook in the very core of the verbatim making macros to set proper fontencoding for instance.

```

468 \@emptify\ttverbatim@hook
471 \def\VerbT1{\def\ttverbatim@hook{\fontencoding{T1}%
 \selectfont}}

```

\ttverbatim@hook

We wish the visible spaces to be the default.

```

475 \let\dobreakspace=\dobreakvisibleSpace

```

### The core: from shortverb

The below is copied verbatim ;-) from doc.pdf and then is added my slight changes.

```

\MakeShortVerb
484 \def\MakeShortVerb{%
485 \gm@ifstar
486 { \def\@shortverbdef{\verb*@\MakeShortVerb}%
487 { \def\@shortverbdef{\verb@\MakeShortVerb} }

\@MakeShortVerb
490 \def\@MakeShortVerb#1{%
491 \xa\ifx\csname_cc\string#1\endcsname\relax
492 \atshortverbinfo{Made }{#1}\@shortverbdef
493 \add@special{#1}%
494 \AddtoPrivateOthers#1% a macro to be really defined in gmdoc.
495 \xa
496 \xdef\csname_cc\string#1\endcsname{\the\catcode`#1}%
497 \begin{group}
498 \catcode`\~\active\lccode`\~=`#1%
499 \lowercase{%
500 \global\xdef\csname_ac\string#1\endcsname{\the\catcode`#1}%
501 \global\let\csname_ac\string#1\endcsname\relax
502 \xa\gdef\x{%
503 \xa\ifmmode\x\string\x\else\x\afterfi{\@shortverbdef~}\fi}%
504 \xa\else\x\afterfi{\@shortverbdef~}\fi}%
505 \xa\else\x\afterfi{\@shortverbdef~}\fi}%
506 This terrible number of \expandafters is to make the shortverb char just other in the
507 math mode (my addition).
508 \end{group}
509 \global\catcode`#1\active
510 \else
511 \atshortverbinfo{\empty{#1 already}}{\empty{\verb(*)}}%
512 \fi}

\DeleteShortVerb
515 \def\DeleteShortVerb#1{%
516 \xa\ifx\csname_cc\string#1\endcsname\relax
517 \atshortverbinfo{\empty{#1 not}}{\empty{\verb(*)}}%
518 \else
519 \atshortverbinfo{Deleted }{#1 as}{\empty{\verb(*)}}%
520 \rem@special{#1}%
521 \global\catcode`#1\csname_cc\string#1\endcsname
522 \global\x\let\x\csname_cc\string#1\endcsname\relax

```

```

523 \ifnum\catcode`#1=\active
524 \begingroup
525 \catcode`\~\active\lccode`\~`#1%
526 \lowercase{%
527 \global\@xa\let\@xa~%
528 \csname\ac\string#1\endcsname}%
529 \endgroup\fi\fi}

```

My little addition

```

\gmv@packname @ifpackageloaded{gmdoc} {%
534 \def\gmv@packname{gmdoc} }{%
535 \def\gmv@packname{gmverb} }

\@shortvrbinfo \def\@shortvrbinfo#1#2#3{%
539 \PackageInfo{\gmv@packname}{%
540 ^^J\empty\@xa\gobble\string#2\aa short reference
541 for\@xa\string#3} }

\add@special \def\add@special#1{%
545 \rem@special{#1}%
546 \@xa\gdef\@xa\dospecials\@xa
547 {\dospecials\do{#1}%
548 \@xa\gdef\@xa\@sanitize\@xa
549 {\@sanitize\makeother{#1}}}

```

For the commentary on the below macro see the `doc` package's documentation. Here let's only say it's just amazing: so tricky and wicked use of `\do`. The internal macro `\rem@special` defines `\do` to expand to nothing if the `\do`'s argument is the one to be removed and to unexpandable CSes `\do` and `\do's argument` otherwise. With `\do` defined this way the entire list is just globally expanded itself. Analogous hack is done to the `\@sanitize` list.

```

\rem@special \def\rem@special#1{%
561 \def\do##1{%
562 \ifnum`#1='##1\else\@nx\do\@nx##1\fi}%
563 \xdef\dospecials{\dospecials}%
564 \begingroup
565 \def\makeother##1{%
566 \ifnum`#1='##1\else\@nx\makeother\@nx##1\fi}%
567 \xdef\@sanitize{\@sanitize}%
568 \endgroup}

```

And now the definition of `verbatim` itself. As you'll see (I hope), the internal macros of it look for the name of the current environment (i.e., `\@currenvir`'s meaning) to set their expectation of the environment's `\end` properly. This is done to allow the user to define his/her own environments with `\verb+` inside them. I.e., as with the `verbatim` package, you may write `\verb+` in the begin definition of your environment and then necessarily `\endverb+` in its end definition. Of course (or maybe *surprisingly*), the commands written in the begin definition after `\verb+` will also be executed at `\begin{<environment>}`.

```

\verb+ \def\verb+{%
582 \edef\gmv@hyphenpe{\the\hyphenpenalty}%
583 \edef\gmv@exhyphenpe{\the\exhyphenpenalty}%
584 \begin{parpenalty}\predisplaypenalty\@verb+
585 \frenchspacing\gmobeyspaces\@xverb+

```

```

586 \hyphenpenalty=\gmv@hyphenpe\relax
587 \exhyphenpenalty=\gmv@exhyphenpe
588 \hyphenchar\font=\m@ne
589 }% in the LATEX version there's \vobeyspaces instead of \gmobeyspaces.
590 \c@namedef{verbatim*}{\begin{parpenalty}\predisplaypenalty\%
591 \begin{verbatim}
592 \end{parpenalty}\end{macro}%
593 \end{macro}%
594 \c@sxverbatim\% it's the same as \xverbatim and defines the verbatim end
595 (a macro delimited with \end{<curr.envir.>}).}
596 }
597 }

\endverbatim
598 \def\endverbatim{\@@par
599 \hyphenchar\font=\gmv@storedhyphenchar% hyphenchar assignments
600 are always global.
601 \ifdim\lastskip>\z@
602 \tempskipa\lastskip\vskip-\lastskip
603 \advance\tempskipa\parskip\advance\tempskipa-%
604 \outerparskip
605 \vskip\tempskipa
606 \fi
607 \addvspace\topsepadd
608 \endparenv}

609 \n@melet{\endverbatim*}{\endverbatim}

610 \begingroup\catcode`!=0%
611 \catcode`[=1\catcode`=2%
612 \catcode`\{=\active
613 \makeother\}%
614 \catcode`\\\=\active%
615 !gdef!\xverbatim[%
616 [!endlinechar!m@ne!everyeof[!@nx]%
617 !edef!\verb@currenvir[%%
618 !@xa!scantokens!@xa[!@currenvir]%
619]% of \verb@currenvir. This macro is defined as the meaning of
620 % \currenvir rescanned. It's done specially for the active star in my
621 verbatims. \currenvir is fully expanded but my active star is \protected.
622 !@xa]% and here a little trick with groups:
623 !@xa!def!@xa!\verb@currenvir
624 !@xa[!verb@currenvir]%
625 !edef!\verb@currenvir@edef[%%
626 !def!@nx!\verb@currenvir@end%
627 #####1!noexpand\end!@nx{%
628 !@xa!unexpanded!@xa[!verb@currenvir]%
629 }[%#
630 #####1!@nx!end[!@currenvir]]%
631 !verb@edef
632 !verb@end]%
633 !endgroup

\c@sxverbatim
644 \let\c@sxverbatim=\xverbatim

```

F. Mittelbach says the below is copied almost verbatim from L<sup>A</sup>T<sub>E</sub>X source, modulo \check@percent.

```
\@verbatim 651 \def\@verbatim{%
```

Originally here was just `\trivlist\item[]`, but it worked badly in my document(s), so let's take just highlights of if.

657    \parsep\parskip

From `\@trivlist`:

```
659 \if@noskipsec\leavevmode\fi
660 \@topsepadd\topsep
661 \ifvmode
662 \advance\@topsepadd\partopsep
663 \else
664 \unskip\par
665 \fi
666 \@topsep\@topsepadd
667 \advance\@topsep\parskip
668 \outerparskip\parskip
```

(End of `\trivlistlist` and `\@trivlist` highlights.)

```
670 \@@par\addvspace\@topsep
671 \if@minipage\else\vskip\parskip\fi
672 \advance\@totalleftmargin\verbatimleftskip
673 \raggedright
674 \leftskip\@totalleftmargin% so many assignments to preserve the list
 thinking for possible future changes. However, we may be sure no internal
 list shall use \@totalleftmargin as far as no inner environments are
 possible in \verbatim[*].
675 \@@par% most probably redundant.
676 \tempswafalse
677 \def\par{% but I don't want the terribly ugly empty lines when a blank line
 is met. Let's make them gmdoc-like i.e., let a vertical space be added as in
 between stanzas of poetry. Originally \if@tempswa\hbox{} \fi, in my
 version will be
678 \ifvmode\if@tempswa\addvspace\stanzaskip\@tempswafalse%
 \fi\fi
679 \@@par
680 \penalty\interlinepenalty\check@percent)%
681 \everypar{\@tempswatrue\hangindent\verbatimhangindent%
 \hangafter\@ne}% since several chars are breakable, there's
 a possibility of breaking some lines. We wish them to be hanging
 indented.
682 \obeylines
683 \ttverbatim
684 \verbatim@specials
685 }

\stanzaskip 699 \@ifundefined{stanzaskip}{\newlength\stanzaskip}{}
700 \stanzaskip=\medskipamount

\verbatimleftskip 703 \newskip\verbatimleftskip
705 \verbatimleftskip\leftmargini

\verbatimhangindent 707 \newskip\verbatimhangindent
709 \verbatimhangindent=3em

\check@percent 713 \providecommand*\check@percent {}
```

In the gmdoc package shall it be defined to check if the next line begins with a comment char.

Similarly, the next macro shall in gmdoc be defined to update a list useful to that package. For now let it just gobble its argument.

```
720 \providecommand*\AddtoPrivateOthers[1] {}
```

Both of the above are \provided to allow the user to load gmverb after gmdoc (which would be redundant since gmdoc loads this package on its own, but anyway should be harmless).

Let's define the 'short' verbatim command.

```
\verb* 729 \def\verb{%
\verb 730 \relax\ifmmode\hbox\else\leavevmode\null\fi
731 \bgroup
732 \ttverbatim
733 \verb@specials
734 \gm@verb@eol
735 \gm@ifstar
736 {\verb@lasthook\@sverb@chbsl}%
737 {\gmobeyspaces\frenchspacing\verb@lasthook\@sverb@chbsl}%
 in the LATEX version there's \vobeyspaces instead of \gmobeyspaces.

741 \emptyify\verb@lasthook
\@sverb@chbsl 743 \def\@sverb@chbsl#1{\@sverb#1\check@bslash}
@def@breakbslash 746 \def\@def@breakbslash{\breakbslash}% because \ is \defined as \break \
bslash not \let.
```

For the special case of a backslash opening a (short) verbatim, in which it shouldn't be breakable, we define the checking macro.

```
\DefineTypeChar 753 \DeclareCommand\DefineTypeChar [mmo]{%
% #1 m the char as a CS,
% #2 m short name of the char.
% [#3] o the cs of the char in 'other' catcode.

760 \namedef{\gmu@#2wd@name}{#2wdof
761 \xa\meaning\the\font\spaceat\detoken@xa\f@sizept}

763 \namedef{\gmu@measure#2}{%
764 \unless\ifcsname\csname\gmu@#2wd@name\endcsname\endcsname
765 \gmu@measurewd[#1]\edefs\gmu@tempa as the width of the char and
 \gmu@tempb as the width of the char among 20 copies of itself.
768 \xa\gn@melet\csname\gmu@#2wd@name\endcsname{\gmu@tempb}% here
 we let the CS with the name contained in \gmu@<char-name>wd@name to
 the expanded value of width of the char measured among copies of it.
772 \typeout{@@@@package@gmverb:
773 \xa\string
774 \csname\csname\gmu@#2wd@name\endcsname\endcsname
775 \spaceis
776 \csname\csname\gmu@#2wd@name\endcsname\endcsname}%
777 \fi
778 }% of \gmu@measure<char-name>.

780 \namedef{\type@#2}{%
```

```

781 \@nx\leavevmode
782 \@xa\@nx\csname\gmu@measure#2\endcsname
783 \hbox\to\@nx\csname
784 \@xa\@nx\csname\gmu@#2wd@name\endcsname\@nx\endcsname%
785 {\IfValueTF{#3}{\@nx#3}{\@xa\@nx\csname\#2\endcsname}%
786 \@nx\hss}%
787 }% of \type@char-name,
788 }% of \DefineTypeChar.

790 \DefineTypeChar{\bslash}% this defines \type@bslash and its aides \gmu@measurebsl
 and \gmu@bslashwd@name.

793 \DefineTypeChar{{\{}{\xilbrace}}% this defines \type@lbrace and
 its auxilia analogous to the above.

\check@bslash 796 \def\check@bslash{%
797 \@ifnextchar\def\breakbslash
798 {\type@bslash\yeshy\gobble}% note we allow hyphenation but actually
 this will have effect if \hyphenchar\font allows hyphenation (when it's
 not > o).
801 {}}
805 \let\verb@balance@group\empty

\verb@egroup 808 \def\verb@egroup{\global\let\verb@balance@group\empty
809 \hyphenchar\font=\gmv@storedhyphenchar
810 \egroup}

\gm@verb@eol 814 \let\gm@verb@eol\verb@eol@error

```

The latter is a  $\text{\LaTeX}_2\epsilon$  kernel macro that \activates line end and defines it to close the verb group and to issue an error message. We use a separate CS'cause we are not quite positive to the forbidden line ends idea. (Although the allowed line ends with a forgotten closing shortverb char caused funny disasters at my work a few times.) Another reason is that gmdoc wishes to redefine it for its own queer purpose.

However, let's leave my former 'permissive' definition under the \verb@eol name.

```

826 \begingroup
827 \obeylines\obeyspaces%
828 \gdef\verb@eolOK{\obeylines%
\check@percent 829 \def^M{\check@percent}%
830 }%
831 \endgroup

```

The \check@percent macro here is \provided to be \empty but in gmdoc employed shall it be.

Let us leave (give?) a user freedom of choice:

```
\verbeolOK 836 \def\verbeolOK{\let\gm@verb@eol\verb@eolOK}
```

And back to the main matter,

```

839 \def@sverb#1{%
840 \catcode`#1\active\lccode`\~=`#1%
841 \gdef\verb@balance@group{\verb@egroup
842 \@latex@error{Illegal use of \bslash verb command}%
 \ehc}%
843 \aftergroup\verb@balance@group

```

```

844 \lowercase{\let~\verb@egroup}{ here we make the delimiter to be the
845 macro closing the verbatim group.
846 }

```

```
\verbatim@nolig@list 848 \def\verbatim@nolig@list{\do` \do\<\do\> \do\, \do\'\ \do\-\}
```

```

849 \def\do@noligs#1{%
850 \catcode`\#1\active
851 \begingroup
852 \lccode`\~=`\#1\relax
853 \lowercase{\endgroup\def~{\leavevmode\kern\z@\char`\#1}}}

```

And finally, what I thought to be so smart and clever, now is just one of many possible uses of a general almost Rainer Schöpf's macro:

```

\dekclubs 859 \def\dekclubs{\gm@ifstar{\MakeShortVerb*}{}{\MakeShortVerb%
860 \{}}
\olddekclubs 860 \def\olddekclubs{\OldMakeShortVerb{}}

```

But even if a shortverb is unconditional, the spaces in the math mode are not printed. So,

```

\edverbs 868 \newcommand*\edverbs{%
869 \let\gmv@dismath\[%%
870 \let\gmv@edismath\]%
871 \def\[\{%
872 \ifnextac\gmv@disverb\gmv@dismath\}%
873 \relaxen\edverbs\}%
\gmv@disverb 875 \def\gmv@disverb{%
876 \gmv@dismath
877 \hbox\bgroup\def\]\{\egroup\gmv@edismath\}}

```

### doc- and shortverb-compatibility

One of minor errors while  $\text{\TeX}$ ing doc.dtx was caused by my understanding of a 'shortverb' char: at my settings, in the math mode an active 'shortverb' char expands to itself's 'other' version thanks to \string. doc/shortverb's concept is different, there a 'shortverb' char should work as usual in the math mode. So let it may be as they wish:

```

\old@MakeShortVerb 890 \def\old@MakeShortVerb#1{%
891 @xa\ifx\csname_cc\string#1\endcsname\relax
892 @shortvrbinfo{Made_}{#1}\@shortvrbdef
893 \add@special{#1}%
894 \AddtoPrivateOthers#1% a macro to be really defined in gmdoc.
895 @xa
896 \xdef\csname_cc\string#1\endcsname{\the\catcode`\#1}%
897 \begingroup
898 \catcode`\~\active\lccode`\~`#1%
899 \lowercase{%
900 \global@xa\let\csname_ac\string#1\endcsname~%
901 @xa\gdef@xa~\@xa{%
902 @shortvrbdef~}}%
903 \endgroup
904 \global\catcode`\#1\active
905 \else
906 @shortvrbinfo@empty{#1 already}{@empty\verb(*)}%

```

```

908 \fi}
\OldMakeShortVerb{%
911 \def\OldMakeShortVerb{\begingroup
912 \let\@MakeShortVerb=\old@MakeShortVerb
913 \g@ifstar{\eg@MakeShortVerbStar}{\eg@MakeShortVerb}%
916 \def\eg@MakeShortVerbStar#1{\MakeShortVerb*#1\endgroup}%
917 \def\eg@MakeShortVerb#1{\MakeShortVerb#1\endgroup}%

```

### Grey visible spaces

In August 2008 Will Robertson suggested grey spaces for gmdoc. I added a respective option to that package but I like the grey spaces so much that I want provide them for any verbatim environments, so I bring the definition here. The declaration, if put in the preamble, postpones redefinition of \visiblespace till \begin{document} to recognise possible redefinition of it when xltextra is loaded.

```

929 \let\gmd@preambleABD\AtBeginDocument
930 \AtBeginDocument{\let\gmd@preambleABD\firstofone}
932 \RequirePackage{xcolor}%
933 for \providecolor
\VisSpacesGrey{%
934 \def\VisSpacesGrey{%
935 \providecolor{visspacesgrey}{gray}{0.5}%
936 \gmd@preambleABD{%
937 \edef\visiblespace{%
938 \hbox{\@nx\textcolor{visspacesgrey}{%
939 {\@xa\unexpanded\@xa{\visiblespace}}}}%
940 }%
941 }%

```

### Verbatim specials—CSes in verbatims

```

\verbatimspecials{%
946 \pdef\verbatimspecials{%
 This declaration only defines a bearer of the 'ver-
 batim specials'.
 % #1 m char for verbatim escape char (for catcode 0), has to be unbraced,
 % #2 m char for verbatim group begin (for catcode 1), has to be unbraced,
 % #3 m char for verbatim group end (for catcode 2), has to be unbraced,
 % [#4] o char for verbatim math shift (for catcode 3),
 % [#5] o char for a shorthand for \metachar.
 % {[#6]} b (optional braced) additional stuff (commands) to be executed at the
 beginning of the verbatims.

```

```

947 \@bsphack
948 \begin{group}
949 \let\do\@makeother
950 \dospecials
951 \catcode`_=10
952 \verb@specials@iii}

```

```

\verb@specials@iii{%
953 as you see, we take only first three
954 arguments in a despecialized group. It's to avoid \futurelet of the
955 optionals' parser to touch (and thus spoil) subsequent token. Yes, we could
956 handle the case of a space or single line end but handling the case of a back-
957 slash would be somewhat difficult.

```

```

958 \endgroup
959 \def\verb@specials@list{#1#2#3}%
960 \@ifnextchar[%

```

```

973 { \begingroup\let \do \@makeother\dospecials
974 \catcode`_=9
975 \verbatim@specials@iv}%
976 { \addtomacro\verbatim@specials@list {\NoValue \NoValue}%
977 \verbatim@specials@vi}%
978 }% of \verbatim@specials@iii.

\verbatim@specials@iv
980 \pdef\verbatim@specials@iv[#1] {%
981 \endgroup
982 \addtomacro\verbatim@specials@list{#1}%
983 \@ifnextchar[%
984 { \begingroup\let \do \@makeother\dospecials
985 \catcode`_=9
986 \verbatim@specials@v}%
987 { \addtomacro\verbatim@specials@list {\NoValue}%
988 \verbatim@specials@vi}%
989 }% of \verbatim@specials@iv.

\verbatim@specials@v
992 \pdef\verbatim@specials@v[#1] {%
993 \endgroup
994 \addtomacro\verbatim@specials@list{#1}%
995 \verbatim@specials@vi
996 }% of \verbatim@specials@v.

\verbatim@specials@vi
998 \DeclareCommand\verbatim@specials@vi\long{b}
999 { \addtomacro\verbatim@specials@list{{#1}}%
1000 \esphack}

\verbatim@specials
1002 \def\verbatim@specials{%
 this is the macro that actually sets the chars given
 in \verbatim@specials@list as the escape char, group begin and group
 end.
1005 \ifdefined\verbatim@specials@list
1006 \xa\verbatim@specials@\verbatim@specials@list
1007 \fi
1008 }% of \verbatim@specials.

\verbatim@specials@
1010 \long\def\verbatim@specials@#1#2#3#4#5#6{%
1011 \catcode`#=0
1012 \protected\@namedef{#1}{#1}%
1013 \catcode`#=1
1014 \protected\@namedef{#2}{#2}%
1015 \catcode`#=2
1016 \protected\@namedef{#3}{#3}%
1017 \edef\gmu@tempa{\the\endlinechar}%
1018 \endlinechar\m@ne_%
 we have to suppress adding of a line end by \scant
 okens since it would turn into an active char ^^M and raise an error (which
 actually did happen).
1019 \scantokens{%
1020 #1\relax
1021 #1\relax
1022 #1\relax
1023 #1\relax
1024 #1\relax
1025 #1\relax
1026 #1\relax
1027 #1\relax
1028 }%
1029 \endlinechar\gmu@tempa\relax

```

```

1030 \IfValueT{#4} {%
1031 \catcode`#4=3
1032 \protected\@namedef{#4}{#4}%
1033 \IfValueT{#5} {%
1034 \begingroup
1035 \lccode`\~=\#5\lowercase{\endgroup\let~\metachar}%
1036 \protected\@namedef{#5}{#5}%
1037 \catcode`#5=\active
1038 }% of if value #5.
1039 \PutIfValue{#6}%
1040 }

\noverbatimspecials 1042 \pdef\noverbatimspecials{\let\verb@specials@list%
1043 \undefined}

\GMverbatimspecials 1045 \def\GMverbatimspecials{%
1046 \gm@testdefined\XeTeXversion\iftrue
1047 \verb@specials@list\relax[{\x}][]%
1048 {\def\|{\metachar{\vert}}%
1049 \makestarlow
1050 \relaxen\%' to provide a CS separator (space is not 10 in verbatims).
1051 }% of #6.
1052 \fi
1053 }% of \GMverbatimspecials.

```

### Partial \verb in arguments

Now command for partial verbatims in arguments of commands:

```

1060 \let\gmu@tempa\all@stars
1061 @xa\addtomacro@xa\gmu@tempa@xa{\all@unders}
1062 \foone{\catcode`#=active}
1063 {\def\gmv@hashhalfing{%
1064 \def#{\xiihash@ifnextchar#\gobble{}%
1065 \catcode`#active}%
1066 }
1067
1068 }

1069 \foone{@makeother\^\^R}%
1070 @xa\DeclareCommand@xa\scanverb@xa{%
1071 @xa_Q@xa{\gmu@tempa}>Pm}%
1072 % #1 Q{*_}
1073 % #2 m the stuff to be rescanned and typeset verbatim. Note that %
1074 will be executed during first scan so at best will disappear.

```

Spaces are ignored (because of detokenizers that add a space after a CS) but if you declare some \verb@specials, then you can use // where / denotes the escape char in verbatim.

```

1075 \begingroup
1076 \gmu@septify
1077 \endlinechar=\m@ne
1078 @xa\IfIntersect@xa{\all@stars}{#1}%
1079 {\def\|{\breakablevisspace}%
1080 {\let\|=space}%
1081 @xa\IfIntersect@xa{\all@unders}{#1}%
1082 {}% We make spaces ignored only if there was no underscore in #1 and if #2
1083 doesn't contain \|.

```

```

1092 \IfAmong \u\among{\#2}%
1093 {} {\addtomacro{\verb@lasthook}{\catcode`\u=9}\}}%
1094 \addtomacro{\verb@lasthook}{\gmv@hashhalfing}\}%
1095 \makeother{R}%
1096 \edef\gmu@tempa{%
1097 @nx\scantokens{%
1098 \bslash\verb%R%
1099 ^R\detokenize{\#2}^R% we delimit the \verb's argument with
 'other' ^R assuming this char to be used very seldom if at all.
1100 }% of \scantokens,
1101 } \gmu@tempa
1102 \endgroup
1103 }% of \scanverb,
1104 }% of \foone.

verbDiscretionaryHyphen
\gmv@hyphenchar
\gmv@hyphen
1109 \def\verbDiscretionaryHyphen#1#2{%
1110 \def\gmv@hyphenchar{\numexpr#1\relax}%
1111 \def\gmv@hyphen{\#2}%
1112 }
1113 \verbDiscretionaryHyphen{"A6}{}{}%
1114

\verbLongDashes
1115 \def\gmu@tempa{%
1116 \DeclareCommand{\verbLongDashes}{%
1117 >iS{--} to memorise which dash we set
1118 B{1.41}\ expansion of en-dash
1119 >iS{—} as above
1120 B{2}\ expansion of em-dash
1121 }%
1122 }
1123 \def\gmu@tempb{\catcode`-\active\catcode`—\active}
1124 \foone{\catcode`-\active\catcode`—\active}%
1125 {%
1126 \edef\gmu@tempa{@xau\gmu@tempa
1127 {%
1128 @nx\addtomacro{@nx\ttverbatim@hook{%
1129 @xau\gmu@tempb
1130 \def{@nx-{@nx\scalebox{\#1}[1]{\string-}}\}%
1131 \def{@nx-{@nx\scalebox{\#2}[1]{\string—}}\}%
1132 }%
1133 }%
1134 }%
1135 }%
1136 \gmu@tempa
1137 \endinput% for the Tradition.

```

## f. The gmoldcomm Package<sup>1</sup>

March 4, 2010

This is a package for handling the old comments in L<sup>A</sup>T<sub>E</sub>X 2<sub>E</sub> Source Files when L<sup>A</sup>T<sub>E</sub>X ing them with the gmdoc package.

Written by Natror (Grzegorz Murzynowski) 2007/11/10.

It's a part of the gmdoc bundle and as such a subject to the L<sup>A</sup>T<sub>E</sub>X Project Public License.

Scan CSs and put them in tt. If at beginning of line, precede them with %. Obey lines in the commentary.

```
23 \NeedsTeXFormat {LaTeX2e}
24 \ProvidesPackage {gmoldcomm}
25 [2007/11/10 v0.99 LATEX old comments handling
 (GM)]
oldcomments 28 \newenvironment {oldcomments} {%
29 \catcode`\\=\active
30 \let\do\@makeother
31 \do\$% Not only CSs but also special chars occur in the old comments.
32 \do|\do#\do{\do}\do^{\do}_\do\&%
33 \gmc@defbslash
34 \obeylines
35 \StoreMacro\finish@macroscan
36 \def\finish@macroscan{%
37 \@xa\gmc@ifinmeaning\macro@pname\of\gmc@notprinted%
38 {}{{\tt\ifvmode\%\fi\bslash\macro@pname}}%
39 \gmc@checkenv
40 }%
41 }{%
42 }{%
44 {\escapechar\m@ne
45 \xdef\gmc@notprinted{\string\begin,\string\end}}
\gmc@maccname 47 \def\gmc@maccname{macrocode}
\gmc@ocname 48 \def\gmc@ocname{oldcomments}
49 \foone{%
50 \catcode`\\=1\catcode`\\=2
51 \catcode`\\=12\catcode`\\=12}
52 [\def\gmc@checkenv{%
53 \@ifnextchar{%
54 [\gmc@checkenvin{}][]
55 }{%
56 [\gmc@checkenvin{}][]
57 }
58 \def\gmc@checkenvin[#1]{%
59 \def\gmc@resa[#1]{%
60 \ifx\gmc@resa\gmc@maccname
61 \def\next{%
```

---

<sup>1</sup> This file has version number v0.99 dated 2007/11/10.

```

62 \begingroup
63 \def\@currenvir[macrocode]%
64 \RestoreMacro\finish@macroscan
65 \catcode`\\=\z@
66 \catcode`\{=1\catcode`\}=2
67 \macrocode]%
68 \else
69 \ifx\gmoc@resa\gmoc@ocname
70 \def\next[\end[\oldcomments]]%
71 \else
72 \def\next[%
73 \{\#1\}%
74]%
75 \fi
76 \fi
77 \fi
78 \next]%
79]
80]

82 \foone{%
83 \catcode`\\=\z@
84 \catcode`\-=\active}
85 {/def/\gmoc@defbslash{%
86 /let/\scan@macro}}}

\gmoc@defbslash
\task \def\task#1#2 {}

\endinput

```

# Change History

gmdoc v0.96

General:

CheckSum 2395 , a-0

gmdoc v0.98d

\ChangesStart:

An entry to show the change history works: watch and admire. Some sixty \changes entries irrelevant for the users-other-than-myself are hidden due to the trick described on p. 90.  
a-6287

gmdoc v0.991

General:

CheckSum 6134 because of compatibilising the enumargs environment with \DeclareCommandof gutils v.0.991; abandoning gmeometric, a-0 put to CTAN on 2010/03/04, a-0

gmdoc v0.99a

General:

CheckSum 4479 , a-0

gmdoc v0.99b

General:

Thanks to the \edverbs declaration in the class, displayed shortverbs simplified; Emacs mode changed to doctex. Author's true name more exposed, a-8327

gmdoc v0.99c

General:

A bug fixed in \DocInput and all \expandafters changed to \@xa and \noexpands to \@nx, a-8327  
The TeX-related logos now are declared with \DeclareLogo provided in gutils, a-8327

\DocInput:

added ensuring the code delimiter to be the same at the end as at the beginning, a-2518

\gmd@bslashEOL:

a bug fix: redefinition of it left solely to \QeerrEOL, a-3565

gmdoc v0.99d

General:

\@namelet renamed to \n@melet to solve a conflict with the beamer class (in gutils at first), a-8327

\afterfi & pals made two-argument, a-8327

\FileInfo:

added, a-7253

gmdoc v0.99e

General:

a bug fixed in \DocInput and \IndexInput, a-8327

CheckSum 4574 , a-0

gmdoc v0.99g

General:

CheckSum 5229 , a-0

The bundle goes XeTeX. The TeX-related logos now are moved to gutils. ^^A becomes more comment-like thanks to re\catcode'ing. Automatic detection of definitions implemented, a-8327

\hyperref:

added bypass of encoding for loading url, a-2226

\inverb:

added, a-7439

\OldDocInput:

obsolete redefinition of the macro environment removed, a-8143

gmdoc v0.99h

General:

Fixed behaviour of sectioning commands (optional two heading skip check) of mwcls/gutils and respective macro added in gmdoc. I made a tds archive, a-8327

gmdoc v0.99i

General:

A "feature not bug" fix: thanks to \everyeof the \[No]EOF is now not necessary at the end of \DocInput file., a-8327

CheckSum 5247 , a-0

gmdoc v0.99j

General:

CheckSum 5266 , a-0

**quotation:**  
 Improved behaviour of redefined **quotation** to be the original if used by another environment., a-7404

**gmdoc vo.99k**  
**General:**  
 CheckSum 5261 , a-0

**hyperref:**  
 removed some lines testing if  $\text{\TeX}$  colliding with tikz and most probably obsolete, a-2244

**gmdoc vo.99l**  
**General:**  
 CheckSum 5225 , a-0

**\CodeSpacesGrey:**  
 added due to Will Robertson's suggestion, a-2752

**codespacesgrey:**  
 added due to Will Robertson's suggestion, a-2205

**\gmd@FIrescan:**  
 $\text{\scantokens}$  used instead of  $\text{\write}$  and  $\text{\@@input}$  which simplified the macro, a-7287

**macrocode:**  
 removed **\CodeSpacesBlank**, a-5369

**\SelfInclude:**  
 Made a shorthand for **\Docinclude**  $\text{\jobname}$  instead of repeating 99% of **\DocInclude**'s code, a-6976

**gmdoc vo.99m**  
**\@oldmacrocode:**  
 renamed from **\VerbMacrocodes**, a-5463

**$\wedge\wedge M$ :**  
 there was  $\text{\let}\wedge\wedge M$  but  $\text{\QueerEOL}$  is better: it also redefines  $\wedge\wedge M$ , a-2501

**General:**  
 CheckSum 5354 , a-0  
 CheckSum 5356 , a-0  
 Counting of all lines developed (the **countalllines** package option), now it uses **\inputlineno**, a-8327

**\changes:**  
 changed to write the line number instead of page number by default and with **codelineindex** option which seems to be more reasonable especially with the **countalllines** option, a-5148

**\DocInclude:**  
 resetting of codeline number with every **\filedivname** commented out because with the **countalllines** option it caused

that reset at **\maketitle** after some lines of file, a-6912

**\FileInfo:**  
 $\text{\egroup}$  of the inner macro moved to the end to allow **\gmd@ctallsetup**. From the material passed to **\gmd@FIrescan** ending  $\wedge\wedge M$  stripped not to cause double labels., a-7270

**\gmd@bslashEOL:**  
 also **\StraightEOL** with **countalllines** package option lets  $\wedge\wedge M$  to it, a-3565

**\thefilediv:**  
 let to **\relax** by default, a-7127

**theglossary:**  
 added **\IndexLinksBlack**, a-6358

**gmdoc vo.99n**  
**General:**  
 CheckSum 5409 , a-0  
 CheckSum 5547 , a-0  
 In-line comments' alignment developed, a-8327

**\CS:**  
 added, a-7558

**\CSes:**  
 added, a-7566

**\CSS:**  
 added, a-7564

**\finish@macroscan:**  
 the case of  $\text{\_}$  taken care of, a-3895

**\gmd@eat1space:**  
 $\text{\afterfifi}$  added—a bug fix, a-3019

**\gmd@nlperc:**  
 added **\hboxes** in **\discretionary** to score **\hyphenpenalty** not **\exhyphenpenalty**, a-7451

**\gmd@percenthack:**  
 $\text{\space}$  replaced with a tilde to forbid a line break before an in-line comment, a-3122

**\HideDef:**  
 added the starred version that calls **\UnDef**, a-4420

**\HideDefining:**  
 Added the starred version that hides the defining command only once, a-4585

**\ilrr:**  
 added, a-3236

**\mcdiagOff:**  
 developed for the case of in-line comment, a-7626

**\nostanza:**  
 added adding negative skip if in vmode and **\par**, a-2439

**\UnDef:**

a bug fixed: `\gmd@charbychar`  
 appended to `\next`—without it  
 a subsequent in-line comment was  
 typeset verbatim, a-4408

`\verbcodecorr:`  
 added, a-3257

`gmdoc` v0.990

`\@codetonarrskip:`  
 a bug fix: added `\@nostanzagtrue`,  
 a-3382

`\mcdiagOff:`  
 added the optional argument which is  
 the number of hashes (1 by default or  
 2 or 4), a-7626

`gmdoc` v0.99p

General:  
`CheckSum` 5607 , a-0

`\DeclareCommand:`  
 added, a-4441

`\mcdiagOff:`  
 added optional arguments' handling,  
 a-7626

`gmdoc` v0.99q

General:  
`CheckSum` 5603 , a-0

`gmdoc` v0.99r

General:  
`CheckSum` 5607 , a-0  
 put to CTAN on 2008/11/22, a-0

`\toCTAN:`  
 added, a-6431

`gmdoc` v0.99s

General:  
`\@bsphack`—`\@espahack` added to  
`\TextMarginize`, `\Describe`,  
`\DescribeMacro` and  
`\DescribeEnv`, a-8327  
`\gmd@ifinmeaning` moved to `gmutils`  
 and renamed to `\@ifinmeaning`,  
 a-8327  
`CheckSum` 5974 because of `enumargs`  
 handling the argument types of  
`\DeclareCommand`; handling  
`\verbatimspecials`, including  
 writing them to index; introduction of  
`\narrativett` including  
`\ampulexdef` of `gmverb` internals, a-0

`\check@checksum:`  
`!*!*!`... sequence changed to ! !  
 !... for better distinction, a-6548

`\chgs:`  
 added, a-6453

`\DeclareOption:`  
 declared as defining if without star  
 because `\DeclareOption*` doesn't  
 define a named option and so it  
 doesn't have a text argument, a-4454

`\egText@MarginizeEnv:`  
 a bug fixed: braces added around #1,  
 a-5279

`\gmd@ABIOnce:`  
 deferred till the end of package to  
 allow adding titles  
`\AtBeginOnce`, a-2696

`\gmd@nlperc:`  
`\newcommand*` replaced with `\pdef`  
 and optional argument's declaration  
 removed since nothing is done to #1  
 in the body of now-macro. Wrapped  
 in a group for setting  
`\hyphenpenalty`, a-7451

`\gmd@writeFI:`  
 added assignment of `\newlinechar`,  
 a-7280

`\gmd@wykrzykniki:`  
 added, a-6561

`\InclMaketitle:`  
 a bug fixed: `\if\relax\@date`  
 changed to `\ifx`, a-7137

`\mcdiagOff:`  
 added `\StraightEOL` to let the in-line  
 comment continue after this  
 environment, a-7626

`\narrationmark:`  
 added and  
 introduced—`\code@delim` forked to  
 what delimits the code  
`(\code@delim)` and what is typeset  
 at the boundary of code:  
`\narrationmark`, a-2316

`\narrativett:`  
 introduced in `gmutils` and employed in  
 the narrative verbatims, including  
`\ampulexdef` of the `gmverb` macros,  
 a-5244

`\TextMarginize:`  
 added `\@sanitize` in the starred  
 version, a-5270

`\TextUsage:`  
 added `\@sanitize` in the starred  
 version, a-5233

`\toCTAN:`  
 made a shorthand for `\chgs` not  
`\changes`, a-6431

`gmdoc` v0.99t

General:  
 Since `geometry` v.5.2 `gmeometric` is  
 obsolete so was removed, a-8327

`gmdoc` v1.0

`\gmd@chgs:`  
 made `\long` (consider it a bug fix), a-6456

`\gmd@chgs@parse:`  
 made `\long` (consider it a bug fix), a-6467

`gmdocc` v0.74

\edverbs:  
used to simplify displaying shortverbs,  
b-529

gmdocc v0.75  
General:  
CheckSum 130 , b-0

gmdocc v0.76  
\+:  
The gmeometric option made  
obsolete and the gmeometric package  
is loaded always, for  
X<sub>E</sub>T<sub>E</sub>X-compatibility. And the class  
options go xkeyval., b-569

General:  
CheckSum 257 , b-0

gmdocc v0.77  
\+:  
Bug fix of sectioning commands in  
mwcls and the default font encoding  
for T<sub>E</sub>Xing old way changed from QX  
to T<sub>1</sub> because of the 'corrupted NTFS  
tables' error, b-569

General:  
CheckSum 262 , b-0

gmdocc v0.78  
\+:  
Added the pagella option not to use  
Adobe Minion Pro that is not freely  
licensed, b-569

General:  
CheckSum 267 , b-0

gmdocc v0.79  
General:  
CheckSum 271 , b-0

gmdocc v0.80  
General:  
CheckSum 275 , b-0  
CheckSum 276 , b-0

gmcc@fontspec:  
added, b-353

gmdocc v0.81  
General:  
put to CTAN on 2008/11/22, b-0

gmdocc v0.82  
General:  
CheckSum 303 , b-0  
CheckSum 316 because of  
\verbatimspecials, hyphenation  
in verbatims etc., b-0  
CheckSum 320 , b-0

\ac:  
added, b-548

countalllines:  
gmdoc option here executed by default,  
b-363

gmcc@cronos:  
added, for Iwona sans font, b-318

gmcc@cursor:  
added, for T<sub>E</sub>X Gyre Cursor mono font,  
which I embolden a little and shrink  
horizontally a little, b-330  
subtly distinguished weights of the  
T<sub>E</sub>X Gyre Cyursor typewriter font in  
the code and in verbatims in the  
commentary, b-330

\gmcc@dff:  
I commented out setting of Latin  
Modern fonts for sans serif and  
monospaced: X<sub>E</sub>T<sub>E</sub>X/fontspec does  
that by default., b-293

gmcc@lsu:  
added, for Lucida Sans Unicode sans  
font, b-326

gmcc@myriad:  
added, for Myriad Web Pro sans font,  
b-323

gmcc@trebuchet:  
added, for Trebuchet MS sans font, b-320

\LineNumFont:  
added, b-306

gmdocc v0.83  
General:  
CheckSum 332 because of abandoning  
gmeometric since geometry v.5.2  
provides \newgeometry, b-0

gmutils v0.74  
\begnamedgroup@ifcs:  
The catcodes of \begin and \end  
argument(s) don't have to agree  
strictly anymore: an environment is  
properly closed if the \begin's and  
\end's arguments result in the same  
\csname, c-2683

General:  
Added macros to make sectioning  
commands of mwcls and standard  
classes compatible. Now my  
sectionings allow two optionals in  
both worlds and with mwcls if there's  
only one optional, it's the title to toc  
and running head not just to the  
latter, c-7063

gmutils v0.75  
\ifnextcat:  
\let for #1 changed to \def to allow  
things like \noexpand ~ , c-794

\ifnextif:  
\let for #1 changed to \def to allow  
things like \noexpand ~ , c-836

\ifnif:  
added, c-876

gmutils v0.76  
General:

A ‘fixing’ of `\dots` was rolled back since it came out they were O.K. and that was the QX encoding that prints them very tight, c-7063

`\freeze@actives:`  
added, c-5536

gmutils v0.77  
General:  
`\afterfi` & pals made two-argument as the Marcin Woliński’s analogoi are. At this occasion some redundant macros of that family are deleted, c-7063

gmutils v0.78  
General:  
`\@namelet` renamed to `\n@melet` to solve a conflict with the beamer class. The package contents regrouped, c-7063

gmutils v0.79  
`\not@onlypreamble:`  
All the actions are done in a group and therefore `\xdef` used instead of `\edef` because this command has to use `\do` (which is contained in the `\@preamblecmds` list) and `\not@onlypreamble` itself should be able to be let to `\do`, c-3312

gmutils v0.80  
General:  
CheckSum 1689, c-0

`\hfillneg:`  
added, c-5433

gmutils v0.81  
`\dekfracslash:`  
moved here from `pmlectionis.cls`, c-5728

`\ifSecondClass:`  
moved here from `pmlectionis.cls`, c-5697

gmutils v0.82  
`\ikern:`  
added, c-5736

gmutils v0.83  
`\texttilde:`  
postponed to `\begin{document}` to avoid overwriting by a text command and made sensible to a subsequent `/`, c-5389

gmutils v0.84  
General:  
CheckSum 2684, c-0

gmutils v0.85  
General:  
CheckSum 2795, c-0  
fixed behaviour of too clever headings with gmdoc by adding an `\ifdim` test, c-7063

gmutils v0.86  
`\texttilde:`

renamed from `\~` since the latter is one of L<sup>A</sup>T<sub>E</sub>X’s accents, c-5389

gmutils v0.87  
General:  
CheckSum 4027, c-0  
the package goes ε-T<sub>E</sub>X even more, making use of `\ifdef` and the code using UTF-8 chars is wrapped in a X<sub>H</sub>T<sub>E</sub>X-condition, c-7063

gmutils v0.88  
General:  
CheckSum 4040, c-0

`\RestoreEnvironment:`  
added, c-1181

`\storedcsname:`  
added, c-1172

`\StoreEnvironment:`  
added, c-1177

gmutils v0.89  
General:  
removed obsolete adjustment of `pgf` for X<sub>H</sub>T<sub>E</sub>X, c-7063

gmutils v0.90  
General:  
CheckSum 4035, c-0

`\XeTeXthree:`  
adjusted to the redefinition of `\verb` in `xlxtra 2008/07/29`, c-4158

gmutils v0.91  
General:  
CheckSum 4055, c-0  
removed `\jobnamewoe` since `\jobname` is always without extension. `\xiispace` forked to `\visiblespace \let` to `\xxt@visiblespace` of `xtxtra` if available. The documentation driver integrated with the `.sty` file, c-7063

gmutils v0.92  
`\@checkend:`  
shortened thanks to `\@ifenvir`, c-2771

`\@gif:`  
added redefinition so that now switches defined with it are `\protected` so they won’t expand to a further expanding or unbalanced `\iftrue/false` in an `\edef`, c-489

`\@ifenvir:`  
added, c-2707

`\@ifprevenvir:`  
added, c-2748

General:  
CheckSum 4133, c-0

gmutils v0.93  
`\@nameedef:`  
added, c-282

General:

A couple of  
`\DeclareRobustCommand*`  
 changed to `\pdef`, c-7063  
`CheckSum 4140 , c-0`  
`CheckSum 4501 , c-0`  
 The numerical macros commented out  
 as obsolete and never really used, c-7063  
`\ampulexdef:`  
 added, c-2523  
`\em:`  
 added, c-5110, c-5119  
`\gmu@RPfor:`  
 renamed from `\gmu@RPIf` and #3  
 changed from a csname to CS, c-5600  
`\litshape:`  
 copied here from E. Szarzyński's *The Letters*, c-5068  
`\lsl:`  
 copied here from E. Szarzyński's *The Letters*, c-5093  
`\nocite:`  
 a bug fixed: with natbib an 'extra )'  
 error. Now it fixes only the standard  
 version of `\nocite`, c-3349  
`\pdef:`  
 added, c-240  
`\pprovide:`  
 added, c-269  
`\provide:`  
 added, c-254  
`\textlit:`  
 added, c-5085  
`\thoussep:`  
 added, c-6513  
 gutils v0.94  
`\@xau:`  
 added, c-225  
 General:  
`\bgroup` and `\egroup` in the macro  
 storing commands and in `\foone`  
 changed to `\begingroup` and  
`\endgroup` since the former produce  
 an empty `\mathord` in math mode  
 while the latter don't, c-7063  
`CheckSum 4880 , c-0`  
 removed `\unex@namedef` and  
`\unex@nameuse`, probably never  
 really used since they were  
 incomplete: `\edef@other`  
 undefined, c-7063  
 The code from ancient `xparse` (1999) of  
`\TeXLive` 2007 rewritten here, c-7063  
`\afterfifi:`  
`\if` removed from parameters' string,  
 c-357  
`\ampulexdef:`  
 made `xparse`-ish and `\ampulexset`  
 removed, c-2523  
`\dekfracc:`  
 made to work also in math mode, even  
 with math-active digits, c-4265  
`\gm@ifundefined:`  
 added. All `\@ifundefined`s used by  
 me changed to this, c-623  
 made robust to unbalanced `\ifs` and  
`\fis` the same way as `LATEX`'s  
`\@ifundefined` (after a heavy  
 debug :-), c-623  
`\gmathfurther:`  
 removed definition of `\langleletter`s and  
`\langle digit`s, c-4844  
`\ldate:`  
`\leftline` replaced with  
`\par... \par` to work well with  
`floatfl`, c-6303  
`\prependtomacro:`  
 order of arguments reversed, c-589  
`\resizegraphics:`  
`\includegraphics` works well in  
`XETEX` so I remove the complicated  
 version with `\XeTeXpicfile`, c-4294  
`\textbullet:`  
 the `XETEX` version enriched with  
`\iffontchar` due to lack of bullets  
 with the default settings reported by  
 Morten Høgholm and Edd Barrett, c-6193  
 gutils v0.95  
 General:  
`CheckSum 4908 , c-0`  
`\gm@testdefined:`  
 added, c-641  
`\gm@testundefined:`  
 added, c-656  
 gutils v0.96  
 General:  
`CheckSum 5363 , c-0`  
 put to CTAN on 2008/11/21, c-0, c-7063  
`\glyphname:`  
 moved here from my private  
 document class, c-6198  
`\gmathfurther:`  
 Greek letters completed. Wrapped  
 with `\addtotoks` to allow using in  
 any order with `\garamath` and  
 others, c-4844  
 the `\everymath`'s left brace moved  
 here: earlier all the stuff was put into  
`\everymath`, c-4913  
`\gmathscripts:`  
 added, c-4959  
`\LuaTeX:`  
 added, c-4109  
`\pk:`

\textup removed to allow slanting the name in titles (that are usually typeset in Italic font), c-3053

gmutils v0.97

- \\*:
  - removed (it was a text tilde, available as \texttilde), c-5381
- \@allbutfirstof:
  - added, c-322
- General:
  - CheckSum 5375, c-o
  - put to CTAN on 2008/11/22, c-o
- \pdfLaTeX:
  - added, c-4086

gmutils v0.98

- \@allbutfirstof:
  - renamed from \@secondofmany, c-322
- \@ifdetokens:
  - rewritten not to make entries in the hash table, thanks to \detokenize and made robust to open \ifs in the arguments thanks to substitution of explicit parameters with \@firstoftwo and \@secondoftwo, c-2722
- \@ifinmeaning:
  - moved here from gmdoc and rewritten, including split to \IfAmong because the latter is needed in \DeclareCommand, c-442
- \@ifnextsingle:
  - added test for \egroup, c-956
  - extracted from \@ifnextac, c-951
- General:
  - CheckSum 5429, c-o
  - CheckSum 5577 because of \DeclareCommand, c-o
  - CheckSum 5627 because of \fakeonum, c-o
  - CheckSum 6035 because of \DeclareCommand, \arg, c-o
  - CheckSum 6129 because of \DeclareEnvironment, c-o
  - CheckSum 6147 because of \arg in verbatims, c-o
  - CheckSum 6535 because of \UrlFix, c-o
  - CheckSum 6656 because of type settings for gmdoc (\narrativett) and for \verbatimspecials, c-o
- \addto@macro:
  - \toks@ replaced with some \expandafters because use of \toks@ here caused a disaster in \DeclareCommand, c-576
- \ampulexdef:
  - a bug fixed: added \unexpanded, c-2539

first argument (the prefix) made of the Q type, c-2523

\arg:
 

- made iterating thanks to \DeclareCommand\arg@dc, c-3299

\ATfootnotes:
 

- moved here from my own private macro package to allow the beauty of these footnotes for the general audience, c-6611

\balsmiley:
 

- moved here from my personal macro package since I needed it in the documentation of gmutils, c-6674

\cs:
 

- added \verbatim@specials, c-3067
- made \- switch to \normalfont because IMHO this underlines the fact that a CS belongs to the narrative.
- \hyphenchar set to 45 as in usual texts, c-3067

\DeclareCommand:
 

- added the \@bsphack—\@esphack option, c-2262
- added the Q {\langle tokens \rangle} argument type (a word over the alphabet \langle tokens \rangle), c-2262
- added the S/T spec parsing, the s spec parsing rewritten to be a shorthand for S {\*}, unused code removed, c-1605

\enoughpage:
 

- \par removed since to let it be used for \pagebreak inside a paragraph, c-5495
- made ε-TeX-ish, c-5495
- made ifthenelse-like with 'then' and 'else' optional default \nothing and \newpage resp., c-5495

\fakeonum:
 

- added, c-5148

\gmu@lowstar:
 

- renamed from \@ifstar since something redefines \@ifstar, c-935

\IfAmong:
 

- split from \IfAmong, c-399

\IfNoValueTF:
 

- the xparse tests for the presence of value redefined and much simplified (43 CS'es less). Moreover, they are now fully expandable:
- \IfNoValueTF, \IfNoValueT,
- \IfNoValueF, \IfValueTF,
- \IfValueT, \IfValueF, c-2162

\napapierkicore:
 

- added optional star controlling globalness, c-6003

\rrthis:  
added, c-6661

\scantnoline:  
added, c-6682

gmutils vo.99  
\@ifedetokens:  
moved to a separate macro from  
\@ifenvir and made symmetric to  
both arguments, c-2722

gmutils vo.991  
\@ifnextif:  
inner macro fixed to handle active  
chars more properly (more as \if  
would do it), c-836

\@xau:  
made 1-parameter, c-225

General:  
CheckSum 8257 because of some  
shorthands and major development  
of \DeclareCommand, including  
'Knuthian' and general optional  
arguments and  
'\afterassignment'  
pseudo-argument, c-0  
put to CTAN on 2010/03/04, c-0

gmverb vo.79  
\edverbs:  
added, e-860

gmverb vo.80  
\edverbs:  
debugged, i.e. \hbox added back and  
redefinition of \[, e-860

\xiiclus:  
\ttverbatim@hook added, e-436

gmverb vo.81  
General:  
\afterfi made two-argument (first  
undelimited, the stuff to be put after  
\fi, and the other, delimited with  
\fi, to be discarded, e-1144

gmverb vo.82  
General:  
CheckSum 663 , e-0

gmverb vo.83  
General:  
added a hook in the active left brace  
definition intended for gmdoc  
automatic detection of definitions (in  
line 371), e-1144  
CheckSum 666 , e-0

gmverb vo.84  
General:  
CheckSum 658 , e-0

gmverb vo.85  
General:

added restoring of \hyphenpenalty  
and \exhyphenpenalty and  
setting \hyphenchar=-1, e-1144  
CheckSum 673 , e-0

gmverb vo.87  
General:  
CheckSum 661 , e-0  
visible space tidied and taken from  
xltextra if available. gmutils required.  
The \xi... CSes moved to gmutils.  
The documentation driver moved  
into the .sty file, e-1144

gmverb vo.88  
General:  
CheckSum 682 , e-0

\VisSpacesGrey:  
added, or rather moved here from  
gmdoc, e-934

gmverb vo.89  
General:  
\dekclubs, \dekclubs\* and  
\olddekclubs made more  
consistent, shorthands for  
\MakeShortVerb\|,  
\MakeShortVerb\*\| and  
\OldMakeShortVerb\|  
respectively., e-1144  
CheckSum 686 , e-0

gmverb vo.90  
General:  
CheckSum 684 , e-0  
some \bgroup changed to  
\begin{group}, e-1144

gmverb vo.91  
General:  
CheckSum 686 , e-0  
put to CTAN on 2008/11/21, e-0

\verbatimleftskip:  
added, e-703

gmverb vo.92  
General:  
CheckSum 979 because of  
\verbatimspecials, hyphenation  
in verbatims, low star in verbatims,  
kerning of backslash in shrunk fonts,  
e-0

\breakbslash:  
renamed from \fixbslash, e-398

\breaklbrace:  
renamed from \fixlbrace, e-405

\ttverbatim:  
added \makeatletter to sound with  
the 'verbatim specials', namely to  
allow control sequences containing @,  
e-450

gmverb vo.93  
General:

CheckSum 1035 because of a bug fix in  
  \scanverb (halving the hashes), e-0  
put to CTAN on 2010/03/04, e-0  
\gmv@hashhalfing:  
  cut out as separate macro, e-1064  
\verbDiscretionaryHyphen:

added to synchronise hyphen chars in  
gmdoc's documentation, e-1109

\xiihash:

mandatory argument made long (a  
bug fix), e-1072

# Index

Numbers written in italic refer to the code lines where the corresponding entry is described; numbers underlined refer to the code line of the definition; numbers in roman refer to the code lines where the entry is used. The numbers with no prefix are page numbers. All the numbers are hyperlinks.

#1, c-5900  
\*, c-7024  
\\*, a-3682, a-6031, c-691, c-693, c-925, c-985, c-986, c-987, c-989, c-993, c-994, c-995, c-999, c-5591, c-5592, c-6825, c-6826, c-7023, c-7026, c-7048  
\+, 22, a-6031, a-7550, b-551, c-5593, c-6732  
\-, a-6031, b-551, c-2976, c-3008, c-3010, c-3082, c-6181, c-6757, c-6831, c-6833, e-848  
\<...>, c-2945  
\<..>, 115  
\@codeline@wrindex, a-5178  
\@nil, a-4351, a-4832, a-4837, a-4935, a-4972, a-5063, a-5542, a-5560, a-6192, a-6249, a-6252, a-6271, a-6442, a-6462, a-6464, a-6467, a-6720, a-7284, c-319, c-322, c-2168, c-2614, c-2619, c-3023, c-3025, c-3027, c-3136, c-4383, c-4391, c-5254, c-5255, c-5288, c-5289, c-5291, c-5295, c-5303, c-5305, c-5308, c-5318, c-5320, c-5322, c-5329, c-5331, c-5333, c-5339, c-5343, c-5756, c-5766, c-5918, c-5923, c-5926, c-5927, c-5931  
\@par, a-2581, a-3192, a-3210, a-3317, a-5681  
\@settexcodehangi, a-2334, a-2877, a-2939  
\@EOF, a-8309, a-8312  
\@M, a-6255, b-551, c-3067, c-3669, c-5736, c-5741, c-6805, c-6813, c-6822, c-6831, c-6865, c-6891, c-6908  
\@MakeShortVerb, a-8146, e-486, e-487, e-490, e-912  
\@NoEOF, a-8307, a-8311  
\@alph, a-6874, a-6875  
\@aarg, c-3210, c-3211, c-3212, c-3214, c-3217  
\@aarga, c-3210, c-3212  
\@aftercodegfalse, a-2910, a-3199, a-3382  
\@aftercodegtrue, a-2444, a-2917, a-2946, a-3168, a-6043, a-6077  
\@afterheading, c-3662  
\@afternarrgfalse, a-2444, a-2917, a-3168, a-6043, a-6077  
\@afternarrgtrue, a-2557  
\@allbutfirstof, a-4837, c-322  
\@badend, c-2771  
\@beginputhook, a-2506, a-2619, a-2620, a-8214  
\@begnamedgroup, c-2652, c-2676, c-2681  
\@begnamed| group@ifcs, c-2677, c-2680  
\@car, c-4019  
\@cclv, c-5574  
\@cclvi, c-5559  
\@charlb, a-6860  
\@charrb, a-6862  
\@checkend, c-2771  
\@clexception, c-5697  
\@clubpenalty, a-2488, c-3674  
\codeskipput, 45  
\codeskipputgfalse, a-2557, a-2888, a-3169, a-6043, a-6078, a-7398  
\codeskipputgtrue, a-2427, a-2434, a-2443, a-2890, a-3318, a-5357, a-5368, a-5502, a-5509  
\codetonarrskip, a-2508, a-2770, a-2791, a-3144, a-3165, a-3209, a-3228, a-3363, a-3409  
\countalllinestrue, a-2136, a-2140  
\ctrerr, a-6881  
\currenvir, a-5419, a-5426, a-5449, a-7411, a-7415, c-2659, c-2661, c-2719, e-625, e-639, f-63  
\currenvline, c-2666  
\currsize, c-2835, c-2836, c-2837, c-2838, c-2839, c-2840, c-2841, c-2842, c-2843, c-2844  
\dc, c-1629, c-1657, c-1708, c-1732, c-1733, c-1768, c-2316, c-2386  
\dc@, c-1631, c-1643, c-2327

\@dc@Gname, c-2118,  
c-2119, c-2120,  
c-2125, c-2136,  
c-2137, c-2138, c-2142  
\@dc@Gparams, c-2121,  
c-2139, c-2147  
\@dc@K@defaultrep1,  
c-2359, c-2375  
\@dc@add@ms, c-1666,  
c-1684, c-1720  
\@dc@addargum, c-1750,  
c-1753, c-1756,  
c-1757, c-1778,  
c-1781, c-1786,  
c-1789, c-1800,  
c-1812, c-1821,  
c-1828, c-1840,  
c-1851, c-1862,  
c-1863, c-1876,  
c-1929, c-1940,  
c-1946, c-1947,  
c-1953, c-1954,  
c-1959, c-1962,  
c-1970, c-1976,  
c-1982, c-1986,  
c-2024, c-2092,  
c-2097, c-2121,  
c-2126, c-2139,  
c-2143, c-2206, c-2374  
\@dc@alllong@false,  
c-2287  
\@dc@alllong@true, c-2292  
\@dc@argnum, c-1599,  
c-1617, c-1693,  
c-1695, c-2311  
\@dc@arguments, c-1596,  
c-1654, c-1751,  
c-1771, c-1778,  
c-1781, c-1783,  
c-1786, c-1789,  
c-1791, c-1800,  
c-1803, c-1812,  
c-1815, c-1821,  
c-1824, c-1827,  
c-1828, c-1831,  
c-1840, c-1842,  
c-1850, c-1851,  
c-1862, c-1863,  
c-1872, c-1876,  
c-1899, c-1931,  
c-1934, c-1940,  
c-1947, c-1954,  
c-1959, c-1962,  
c-1967, c-1970,  
c-1973, c-1976,  
c-1979, c-1982,  
c-1985, c-1986,  
c-2015, c-2026,  
c-2087, c-2093,  
c-2096, c-2097,  
c-2105, c-2116,  
c-2121, c-2126,  
c-2134, c-2139,  
c-2143, c-2204,  
c-2207, c-2378, c-2413  
\@dc@bsphack@, c-2296,  
c-2299, c-2302, c-2320  
\@dc@catchernum,  
c-1598, c-1616, c-1691,  
c-2310, c-2368, c-2372  
\@dc@commasep, c-1946,  
c-1949, c-1949, c-1953  
\@dc@define@K, c-1705,  
c-1712, c-2363  
\@dc@env@argstoend,  
c-2304, c-2344, c-2409  
\@dc@esphack@, c-2297,  
c-2299, c-2302, c-2350  
\@dc@global@, c-2305,  
c-2307, c-2318,  
c-2339, c-2380,  
c-2438, c-2450, c-2453  
\@dc@ignorearg, c-1685,  
c-1755  
\@dc@ignorefalse,  
c-1628, c-1697  
\@dc@ignoretrue,  
c-1672, c-1676, c-1767  
\@dc@long@, c-1627,  
c-1671, c-1678,  
c-1687, c-1713,  
c-1765, c-2381, c-2385  
\@dc@long@yes, c-1594,  
c-1671  
\@dc@outer@, c-2305,  
c-2306, c-2318, c-2329  
\@dc@parse@next,  
c-1773, c-1781,  
c-1798, c-1817,  
c-1838, c-1901,  
c-1931, c-1936,  
c-1947, c-1954,  
c-1959, c-1962,  
c-1969, c-1975,  
c-1981, c-2015,  
c-2026, c-2106,  
c-2205, c-2208, c-2378  
\@dc@quiet@false, c-2288  
\@dc@quiet@true, c-2293  
\@dc@seQuence, c-1902,  
c-1921, c-1928,  
c-1928, c-1929  
\@dc@seQuence@finish,  
c-1912, c-1923, c-1926  
\@debugtrue, b-227  
\@def@breakbslash,  
e-746, e-797  
\@defentryze, a-3868,  
a-4378, a-4815,  
a-4822, a-4827, a-5202  
\@dischyp, c-3011  
\@docinclude, a-6751,  
a-6757  
\@dsdirgfalse, a-2899,  
a-2920, a-2988,  
a-3054, a-3135,  
a-3150, a-3156, a-5487  
\@dsdirgtrue, a-2566,  
a-2883  
\@emptyify, a-2695,  
a-2848, a-4717,  
a-4860, a-4988,  
a-5075, a-5082,  
a-5083, a-5890,  
a-6229, a-6870,  
a-7003, a-7158,  
a-7443, a-7445,  
a-7502, a-7518,  
a-7616, a-7622,  
a-8093, a-8094,  
a-8098, c-599, c-600,  
c-604, e-468  
\@endinputhook, a-2534,  
a-2615, a-2616  
\@enumctr, a-7652,  
a-7657, a-7748,  
a-7753, c-3928,  
c-3929, c-3935  
\@enumdepth, c-3924,  
c-3927, c-3928  
\@filepart, c-5821, c-5823  
\@filesfalse, a-7374  
\@fileswithoptions,  
c-5697  
\@firstofmany, a-4832,  
a-4935, a-4972,  
a-5063, a-6192,  
a-6720, a-7284,  
c-319, c-2168, c-3136  
\@firstofone, c-1650  
\@firstofthree, a-6445,  
c-309  
\@gif, c-475, c-476, c-483  
\@glossaryfile, a-5144  
\@gmccnochangeptrue,  
b-239  
\@gmu@mmhboxtrue,  
c-4267, c-6576  
\@if, c-498, c-499, c-502

**File Key:** a=gmdoc.sty, b=gmocc.cls, c=gmutils.sty, d=gmiflink.sty, e=gmverb.sty,  
f=gmoldcomm.sty

```

\@ifeOLactive, a-2644,

 a-2658, a-3440,

 a-3474, a-3616

\@ifQueerEOL, a-2631,

 a-2652, a-2665,

 a-2679, a-5953,

 a-6426, a-6589, c-3288

\@ifXeTeX, b-415, c-4150,

 c-4156, c-4162,

 c-4333, c-5725,

 c-6192, c-6707, c-6723

\@ifaauthor, a-7147,

 a-7165, a-7208

\@ifedetokens, c-2719,

 c-2722, c-2746, c-2760

\@ifempty, c-386, c-397,

 c-409, c-1721, c-2541,

 c-2984, c-2988,

 c-2994, c-3001,

 c-3604, c-3624

\@ifendswithpdf,

 c-4300, c-4305

\@ifenvir, c-2707,

 c-2771, c-6507

\@ifinmeaning, a-3793,

 a-4001, c-442, c-5236

\@ifjobname, c-2746

\@ifl@aded, c-3335

\@ifncat, c-801, c-804, c-819

\@ifnextMac, c-1009,

 c-6102, c-6121

\@ifnextac, a-7762,

 c-887, e-872

\@ifnextcat, a-3742,

 a-3770, a-7560,

 a-7564, a-7566,

 c-794, c-833, c-955,

 c-956, c-1816, c-1826,

 c-1837, c-1849, c-3312

\@ifnextgroup, c-960,

 c-968, c-7011

\@ifnextif, c-836, c-938,

 c-3260, c-3262

\@ifnextsingle, c-888,

 c-951, c-962, c-1873,

 c-1910

\@ifnextspace, c-992,

 c-6100, c-6120

\@ifnif, c-846, c-849, c-876

\@ifnonempty, a-6718, c-397

\@ifnotmw, b-512, c-3483,

 c-3508, c-3657,

 c-3762, c-3851, c-3907

\@ifprevenvir, c-2748

\@ilgroupfalse, a-2948,

 a-3541

\@ilgrouptrue, a-3189,

 a-3238, a-3253

\@include, c-5763, c-5765

\@indexall!

 macrotrue,

 a-2163

\@itemdepth, c-3942,

 c-3945, c-3946

\@itemitem, c-3946, c-3947

\@latexerr, a-6750

\@linesnotnumtrue, a-2119

\@ltxDocIncludetrue,

 a-6994

\@makefntext, a-7052,

 c-6638, c-6647

\@marginparsused!

 false,

 a-2190

\@marginpar!

 susedtrue, a-2180,

 a-2183, a-2185, a-2188

\@minus, c-3773, c-3779,

 c-3785, c-3787,

 c-3792, c-3794,

 c-3801, c-3806

\@nameedef, a-4608,

 c-282, c-2441, c-6929,

 e-780

\@newlinefalse,

 a-2789, a-2921,

 a-3085, a-3103, a-3113

\@newlinetrue, a-2564,

 a-2882

\@nobreakfalse, c-3668,

 c-5481

\@nobreaktrue, c-3663

\@noindextrue, a-2149

\@normalcr, c-6350

\@nostanzagfalse, a-3318

\@nostanzagtrue,

 a-2443, a-3382

\@nx, c-223

\@oarg, c-3198, c-3200,

 c-3201, c-3250

\@oargsq, c-3198, c-3201

\@oldmacrocode, a-5420,

 a-5445

\@old!

 macrocode@launch,

 a-5397, a-5399, a-5402

\@onlypreamble, a-6997,

 a-7949, a-7962,

 a-7966, c-3375,

 c-4837, c-5483, c-6606

\@pageinclindex!

 false,

 a-4668

\@pageinclindextrue,

 a-5334

\@pageindexfalse, a-7961

\@pageindextrue,

 a-2154, a-4737, a-7964

\@parg, c-3205, c-3207,

 c-3208, c-3251

\@pargp, c-3205, c-3208

\@parindent, c-3931,

 c-3932, c-3934,

 c-3949, c-3950,

 c-3952, c-6383,

 c-6417, c-6425,

 c-6426, c-6427,

 c-6429, c-6648

\@pkgextension, c-3336

\@preamblecmds, c-3330,

 c-3332, c-3345, c-3349

\@preenvir, c-2659, c-2760

\@prevgrouplevel, c-2653

\@printalllinenos!

 false,

 a-2137

\@printalllinenos!

 true,

 a-2141

\@relaxen, a-2412,

 a-3270, a-3297,

 a-5830, a-6228,

 a-6338, a-6819,

 a-6886, a-7125,

 a-7126, a-7127,

 a-8011, a-8308, c-608,

 c-609, c-613

\@secondoffive, c-312

\@secondofthree,

 a-6446, a-6447, c-310

\@shortvrbdef, e-486,

 e-487, e-492, e-505,

 e-892, e-903

\@shortvrbinfo, e-492,

 e-511, e-517, e-519,

 e-538, e-892, e-907

\@starttoc, c-5476

\@sverb@chbsl, a-7489,

 e-736, e-737, e-743

\@tempdima, a-7669,

 a-7670, c-5072,

 c-5079, c-5096,

 c-5099, c-5949,

 c-5953, c-5954,

 c-5956, c-5957, c-5961

\@tempdimb, c-5199,

 c-5204, c-5950,

 c-5952, c-5953

\@temptokena, c-1622,

 c-1637, c-1694,

```

File Key: a=gmdoc.sty, b=gmdocc.cls, c=gmutils.sty, d=gmiflink.sty, e=gmverb.sty,  
f=gmoldcomm.sty

c-1695, c-2313,  
 c-2342, c-2456  
`\@temptokenb`, c-1582,  
 c-1583, c-1623,  
 c-1681, c-1689,  
 c-1721, c-1725, c-2314  
`\@textsuperscript`,  
 a-7051, a-7054,  
 c-4326, c-4327  
`\@thirdofthree`, c-311  
`\@toodeep`, c-3925, c-3943  
`\@topnewpage`, c-3564  
`\@topsep`, e-666, e-667, e-670  
`\@topsepadd`, e-607,  
 e-660, e-662, e-666  
`\@trimandstore`, a-2567,  
 a-2769, a-3396,  
 a-3396, a-3404, a-3407  
`\@trimandstore@hash`,  
 a-3397, a-3398  
`\@trimandstore@ne`,  
 a-3404, a-3407  
`\@typeset@protect`, c-1649  
`\@undefined`, c-181, e-1042  
`\@resetlinecount!`  
 true,  
 a-2126  
`\@usgentryze`, a-4856,  
 a-4874, a-4881,  
 a-4962, a-4966,  
 a-5212, a-5262,  
 a-7860, a-7868  
`\@variousauthors!`  
 false,  
 a-7206  
`\@vari!`  
 ousauthorstrue,  
 a-7204  
`\@verbaarga`, c-3214, c-3265  
`\@verbaargact`, c-3217,  
 c-3264  
`\@verbmargin`, c-3223, c-3260  
`\@wckptelt`, c-5891  
`\@whilenum`, c-2006,  
 c-2037, c-5559, c-5574  
`\@xa`, c-222, c-225  
`\@xau`, c-225, c-2375,  
 c-6540, e-1131, e-1134  
`\@xifncat`, c-806, c-819  
`\@xifnif`, c-851, c-876  
`\@xiispaces`, a-4351,  
 c-2614, c-2616, c-2619  
`\@zf@euenctrue`, b-449  
`^^A`, 7, a-3465  
`^^B`, 7, a-3431  
`^^M`, 7, a-3558  
`^^M`, a-2501, a-2881  
`\aalph`, a-6874, a-6921  
`\aarg`, c-3210, c-3253  
`\abovedisplayskip`, a-2383  
`\ac`, b-548  
`\acro`, a-6447, a-7559,  
 b-548, c-5628, c-5671,  
 c-5672, c-5676  
`\acrocore`, c-5655, c-5665  
`\acropresetting`,  
 c-5630, c-5636  
`\activeM`, c-774  
`\actualchar`, 21, a-3653,  
 a-3852, a-4676,  
 a-6154, a-6211,  
 a-6216, a-6696, a-8037  
`\adashes`, c-6156, c-6156,  
 c-6158, c-6167  
`\add@special`, e-493,  
 e-544, e-893  
`\adddefaultfontfea`  
 tures,  
 c-4179  
`\addfontfeature`,  
 b-307, b-340, b-344,  
 c-4192, c-4223,  
 c-4225, c-4256,  
 c-4338, c-4992,  
 c-5006, c-5079,  
 c-5099, c-5158,  
 c-5277, c-5674,  
 c-5719, c-5968, c-6839  
`\addto@estoindex`,  
 a-4821, a-4880,  
 a-4897, a-5201,  
 a-5211, a-5222  
`\addto@estomarginpar`,  
 a-5037, a-5199,  
 a-5200, a-5209,  
 a-5210, a-5215  
`\addto@macro`, c-576, c-586  
`\addtoheading`, c-3638  
`\addtomacro`, a-5308,  
 a-5312, a-5397,  
 a-5398, a-5596,  
 a-7721, a-7723,  
 a-7725, a-7728,  
 a-7730, a-7733,  
 a-7739, a-7745,  
 a-7752, c-586, c-682,  
 c-684, c-686, c-692,  
 c-694, c-1921, c-2429,  
 c-2450, c-4180,  
 c-4972, c-5159,  
 c-6162, c-6163,  
 c-6164, c-6864,  
 c-6890, e-976, e-982,  
 e-987, e-994, e-999,  
 e-1061, e-1093,  
 e-1094, e-1133  
`\AddtoPrivateOthers`,  
 20, a-3598, e-494,  
 e-720, e-894  
`\addtotoks`, c-595,  
 c-1685, c-1686,  
 c-1751, c-4913,  
 c-4960, c-4964  
`\AE`, c-5936  
`\ae`, b-461  
`\afterassignment`,  
 a-7680, a-7716,  
 a-7759, c-1672,  
 c-1706, c-2201,  
 c-2208, c-2214, c-2215  
`\afterfi`, a-2039, a-2989,  
 a-2992, a-3087,  
 a-3089, a-3404,  
 a-3595, a-3710,  
 a-3742, a-3756,  
 a-3781, a-3784,  
 a-4288, a-4292,  
 a-4296, a-4933,  
 a-5491, a-5561,  
 a-5564, a-6602,  
 a-7412, a-7413,  
 a-7416, a-7417,  
 a-7603, b-290, b-309,  
 b-412, b-485, c-259,  
 c-354, c-637, c-996,  
 c-997, c-1041, c-1119,  
 c-2168, c-2169,  
 c-2608, c-2619,  
 c-2681, c-2682,  
 c-2969, c-2970,  
 c-3046, c-3285,  
 c-4153, c-4410,  
 c-5644, c-5660,  
 c-5805, c-5821,  
 c-5923, c-6099,  
 c-6119, c-6555,  
 c-6629, c-7014, e-505  
`\afterfifi`, a-3019,  
 a-3021, a-4839,  
 a-4841, a-4930,  
 a-4948, a-5488,  
 a-5489, a-5651,  
 a-5662, c-256, c-257,  
 c-357, c-632, c-634,  
 c-1097, c-1102,  
 c-4152, c-4404,  
 c-4405, c-6178  
`\afterfififi`, c-365  
`\afteriffifi`, a-3015, c-359  
`\afteriffififi`, c-364  
`\afteriffiffifi`, c-363

**File Key:** a=gmdoc.sty, b=gmocc.cls, c=gmutils.sty, d=gmiflink.sty, e=gmverb.sty,  
 f=gmoldcomm.sty

\AfterMacrocode, 25,  
     a-7606  
 \grave, b-437, b-451  
 \hyphen, c-6181  
 \AKA, c-5672  
 \all@other, c-2548, c-4138  
 \all@stars, a-5307,  
     c-690, c-692, c-694,  
     c-1888, c-6954,  
     e-1060, e-1086  
 \all@unders, a-5308,  
     c-680, c-682, c-684,  
     c-686, e-1061, e-1089  
 alltt, 238  
 \alpha, c-4782  
 \AlsoImplementation,  
     22, a-7981, a-7995  
 \AltMacroFont, a-8098  
 \among, a-7686, a-7693,  
     a-7699, a-7703,  
     a-7707, a-7711,  
     a-7716, a-7759, c-399,  
     c-424, c-458, c-1699,  
     c-1701, c-1861,  
     c-1920, c-1945,  
     c-1952, c-2195,  
     c-2300, c-2306,  
     c-2307, e-1092  
 \ampulexdef, a-3620,  
     a-3623, a-5896,  
     a-7548, c-2510,  
     c-2561, c-2577,  
     c-3372, c-4941,  
     c-6634, c-6638  
 \ampulexhash, a-7549,  
     c-2581  
 \AmSTeX, 23, c-4056  
 \and, a-7105, a-7143  
 \approx, c-4726  
 \arg, 7, c-3243, c-3282,  
     c-3283, 170  
 \arg@dc, c-3214, c-3217,  
     c-3230, c-3235,  
     c-3255, c-3296  
 \ArgumentCatcher@A,  
     c-2153  
 \ArgumentCatcher@a,  
     c-2156  
 \ArgumentCatcher@B,  
     c-1831  
 \ArgumentCatcher@b,  
     c-1815  
 \ArgumentCatcher@C,  
     c-1967  
 \ArgumentCatcher@c,  
     c-1934

\ArgumentCatcher@c@,  
     c-1936, c-1944,  
     c-1958, c-1969  
 \ArgumentCatcher@G,  
     c-2112, c-2153, c-2156  
 \ArgumentCatcher@K,  
     c-2100, c-2108  
 \ArgumentCatcher@m@,  
     c-1817, c-1838  
 \Argument|  
     Catcher@mmmmmmmmmm,  
     c-2087  
 \ArgumentCatcher@O,  
     c-1791  
 \ArgumentCatcher@o,  
     c-1771  
 \ArgumentCatcher@o@,  
     c-1773, c-1780, c-1798  
 \ArgumentCatcher@PA,  
     c-2154  
 \ArgumentCatcher@Pa,  
     c-2157  
 \ArgumentCatcher@PB,  
     c-1842  
 \ArgumentCatcher@Pb,  
     c-1824  
 \ArgumentCatcher@PC,  
     c-1979  
 \ArgumentCatcher@Pc,  
     c-1973  
 \Argument|  
     Catcher@Pc@,  
     c-1951, c-1961,  
     c-1975, c-1981  
 \ArgumentCatcher@PG,  
     c-2130, c-2154, c-2157  
 \ArgumentCatcher@PK,  
     c-2108  
 \ArgumentCatcher@Pm,  
     c-1827, c-1850,  
     c-1985, c-2096  
 \ArgumentCatcher@PO,  
     c-1803  
 \ArgumentCatcher@Po,  
     c-1783  
 \Argument|  
     Catcher@Po@,  
     c-1785, c-1788, c-1810  
 \ArgumentCatcher@PQ,  
     c-1906  
 \ArgumentCatcher@PS,  
     c-1882  
 \ArgumentCatcher@Ps,  
     c-1891  
 \ArgumentCatcher@Q,  
     c-1893, c-1906

\ArgumentCatcher@Q@,  
     c-1903, c-1908, c-1922  
 \Argument|  
     Catcher@Q@@,  
     c-1911, c-1915  
 \ArgumentCatcher@s,  
     c-1887, c-1891  
 \ArgumentCatcher@s@,  
     c-1854, c-1875  
 \ArgumentCatcher@T,  
     c-1883  
 article, b-213  
 \AtBeginDocument,  
     a-2202, a-2210,  
     a-2261, a-2405,  
     a-3851, a-4738,  
     a-5156, a-6412,  
     a-7948, b-418, b-460,  
     c-762, c-3108, c-3281,  
     c-3343, c-3379,  
     c-3454, c-3969,  
     c-4173, c-4189,  
     c-4827, c-5123,  
     c-5125, c-5387,  
     c-5611, c-6054,  
     c-6095, c-6156,  
     c-6158, c-6424,  
     c-6600, c-6708,  
     c-6712, c-6928, e-929,  
     e-930  
 \AtBeginInput, 10, a-2619,  
     a-2629, a-2696,  
     a-3440, a-3474,  
     a-3592, a-3613,  
     a-6237, a-7029,  
     a-7047, a-7371, b-560  
 \AtBeginInputOnce, 10,  
     a-2700, a-8145  
 \AtDIProllogue, 21,  
     a-5891, a-5895  
 \AtEndDocument, a-6556  
 \AtEndInput, 10, a-2615,  
     a-6519, a-7914, a-7939  
 \AtEndOfClass, b-289,  
     b-319, b-322, b-325,  
     b-328, b-336  
 \AtEndOfPackage,  
     a-2201, a-2208,  
     a-2696, c-6725  
 \ATf@font, c-6651, c-6657  
 \ATfootnotes, b-559,  
     b-560, c-6611  
 \author, a-1994, a-7099

File Key: a=gmdoc.sty, b=gmdocc.cls, c=gmutils.sty, d=gmiflink.sty, e=gmverb.sty,  
 f=gmoldcomm.sty

\AVerySpecialMacro, a-8184

\backquote, c-703

\balsmiley, c-6674

\begin, c-2676

\begin\*, c-2676

\belowdisplayshortskip, a-2389, a-2391, a-2392

\belowdisplayskip, a-2388

\beta, c-4783

\beth, b-447

\bgcolor, c-4996

\BibTeX, 23, c-4059

\bidate, c-6271, c-6281

\bigcircle, c-4762

\Biggl, c-4909

\biggl, c-4907

\Biggr, c-4910

\biggr, c-4908

\Bigl, c-4905

\bigl, c-4903

\Bigr, c-4906

\bigr, c-4904

\bigskipamount, c-5430, c-6284

\bihyphen, c-2980

\binoppenalty, c-6798

\boldmath, c-4019

\box, c-4040, c-4902

\breakablevisspace, a-2733, a-2983, a-7477, e-423, e-429, e-1087

\breakbslash, a-7479, e-387, e-398, e-413, e-746

\breakbrace, a-7481, e-362, e-371, e-405

\bslash, a-2863, a-3852, a-3916, a-4251, a-4463, a-4466, a-4467, a-4470, a-4472, a-4483, a-4505, a-4506, a-4507, a-4508, a-4515, a-4677, a-4724, a-4942, a-4972, a-4987, a-5063, a-5074, a-6146, a-6181, a-6182, a-6192, a-6211, a-6213, a-7619, a-7807, a-7923, c-738, c-1049, c-1131, c-1142, c-1144, c-1149, c-1173, c-2039, c-2041, c-2411, c-2442, c-2443, c-2455, c-3631, c-3632, c-3633, c-6085, c-6086, c-6116, c-6117, e-383, e-842, e-1098, f-39

\bullet, c-6196, c-6203

\c@#1, c-5900

\c@ChangesStartDate, a-6245, a-6250, a-6271, a-6273, a-6274, a-6276

\c@CheckSum, a-6489, a-6528, a-6533, a-6545, a-6574, a-6579

\c@codelinenum, a-3273, a-3277, a-5132, a-5146, a-7620

\c@DocInputsCount, a-3276

\c@footnote, a-7103, a-7154

\c@GlossaryColumns, a-6352, a-6352, a-6360

\c@gmd@mc, a-7597, a-7602, a-7603, a-7619

\c@GMhlabel, d-150

\c@IndexColumns, a-5915, a-5915, a-5917, a-5947

\c@NoNumSecs, c-3456

\c@secnumdepth, c-3513

\c@StandardModuleDepth, a-8083

\acute, b-438, b-452

\cat, c-3311

\cataactive, 23, a-7387

\catletter, 23, a-7389

\catother, 23, a-7384

\CDAnd, 24, a-7575

\CDPerc, 24, a-7577

\centerthis, c-6666

\changes, a-6135, a-6145, a-6150

\changes@, a-6137, a-6161, a-6444, a-6461, a-6463, a-6595

\ChangesGeneral, a-6231, a-6237

\ChangesStart, 18, a-6268

ChangesStartDate, 18

\chaptermark, c-3855

\Character@Table, a-7799, a-7805

\CharacterTable, a-7797

\chardef, c-699

\check@bslash, e-743, e-796

\check@checksum, a-6519, a-6522

\check@percent, a-3592, a-8218, e-690, e-713, e-829

\check@sum, a-6485, a-6487, a-6523, a-6533, a-6544, a-6559

\CheckModules, a-8094

CheckSum, a-6489

\CheckSum, 19, a-6487, a-6599

\chgs, a-6453

\chi, c-4806

ChneOelze, a-4496

\chschange, a-6571, a-6575, a-6582

\chschange@, a-6592, a-6594

\chunkskip, 19, 24, a-2422

\cipolagwa, c-6389, c-6411

\cipolagwa, c-6402

class, b-196

\ClassError, c-3630

\cleardoublepage, c-3470, c-6995

\clearemptydou blepage, c-6994

\clubpenalty, a-2488, a-2611, c-3116, c-3669, c-3674

\cmd, a-7528, c-3182

\cmd@to@cs, c-3182, c-3186

\Code@CommonIndex, a-4890, a-4893

\Code@CommonIndexStar, a-4889, a-4896

\Code@DefEnvir, a-5099, a-5194

\Code@DefIndex, a-4807, a-4812, a-5110, a-5571

\Code@DefIndexStar, a-4806, a-4819, a-5575

\Code@DefMacro, a-5099, a-5109

\Code@Delim, a-2296, a-2304

\code@delim, a-2300, a-2496, a-2518, a-2523, a-2590, a-2601, a-3017, a-3041, a-3595, a-5406, a-7338, a-7342, a-7343

\Code@Delim@St, a-2296, a-2298, a-2304

\code@escape@char, a-3053, a-3671  
\Code@MarginizeEnvir, a-5034, a-5037  
\Code@MarginizeMacro, a-3867, a-5021, a-5024, a-5112, a-5113, a-5122, a-5123  
\Code@UsgEnvir, a-5106, a-5205  
\Code@UsgIndex, a-4868, a-4871, a-5119, a-5251  
\Code@UsgIndexStar, a-4867, a-4877  
\Code@UsgMacro, a-5106, a-5118  
\CodeCommonIndex, a-4886, a-8031  
\CodeCommonIndex\*, 16  
\CodeDelim, 20, a-2296, a-2313, a-2518, a-5407, a-7339, a-7575, a-7577  
\CodeDelim\*, 20  
\CodeEscapeChar, 20, a-3668, a-3678, a-5359, a-5370, a-7900  
\CodeIndent, 19, a-2344, a-2347, a-2903, a-3222, a-3588, a-7894, b-418, 114  
\codeline@glossary, a-5136, a-5163  
\codeline@wrindex, a-5129, a-5162, a-5171, a-5176  
\CodeLineIndex, a-7961, a-7962  
\codelinenum, 20, a-3273, a-3277  
\CodeLineNumbered, a-7948, a-7949, 115  
\CodeMarginize, 15, a-5010  
\CodeSpacesBlank, 11, a-2202, a-2740, a-8213  
\codespacesblank, 11, a-2200  
\CodeSpacesGrey, 11, a-2210, a-2752  
\codespacesgrey, 11, a-2205  
\CodeSpacesSmall, a-2745  
\CodeSpacesVisible, a-2731, a-2755, a-2762  
\CodeTopsep, 19, a-2362, a-2381, a-2425, a-2433, a-2442, a-3261, a-3317, a-5357, a-5359, a-5368, a-5370, a-5501, a-7896  
\CodeUsage, 14, a-5101  
\CodeUsgIndex, 16, a-4863  
\color, b-342, c-2931, c-5586, c-5587  
\columnsep, a-5929  
\CommonEntryCmd, 21, a-4659, a-4786  
\continue@macroscan, a-3764, a-3784  
\continuum, c-5622  
\copy, c-4001, c-4032, c-4046, c-4995, c-5008  
\copyrnote, 23, a-7395  
\cos, c-4821  
\count, a-6256, a-6260, a-6261, c-4007, c-4008, c-4009, c-4010, c-4011, c-4012, c-4013, c-4014, c-5557, c-5559, c-5560, c-5561, c-5564, c-5573, c-5574, c-5575, c-5576  
\countalllines, 10, a-2134  
\countalllines\*, 10, a-2139  
\cronos, b-318, 121  
\CS, 24, a-7558, a-7564, a-7566  
\cs, 22, a-7520, a-7548, a-7548, c-3067, c-3101, c-3106, c-3109, c-3182  
\cs@inner, c-3091, c-3093  
\CSes, a-7566  
\CSS, a-7564  
\ctan, c-4825  
\ctg, c-4823  
\cup, c-5005  
\currentfile, a-6716, a-6717, a-6725, a-6726, a-6728, a-6731, a-6732, a-6734, a-6736, a-6738, a-6740, a-6747, a-6748, a-6789, a-6796, a-6823, a-6941, a-6942, a-6944, a-7003, a-8215  
\currentgrouplevel, c-2653  
\cursor, b-330, 121  
\czas, c-6186  
\czer, c-5588, c-5592, c-5593  
\czerwo, c-5587, c-5588  
\daleth, b-447  
\date, a-1995, a-7100  
\date@left, c-6294, c-6305  
\date@line, c-6281, c-6291, c-6305  
\dateskipamount, c-6279, c-6289  
\day, a-6573, a-6577  
\dc, a-7672  
\dc@bsname, c-2270, c-2275, c-2280, c-2324, c-2328, c-2368, c-2372  
\dc@gobblespace@dummy, c-2224, c-2229, c-2229, c-2231  
\dc@grab@afterass, c-1707, c-1743  
\dc@parse@next, c-2124, c-2125  
\DCcoordinate, c-1943, 144  
\DCMessagesfalse, 145  
\DCMessagestrue, 145  
\DCnocoordinate, c-1957, c-1965, 144  
\deadcycles, c-5797, c-5847  
\debug, b-227, 121  
\debug@special, a-3068  
\Declare@Dfng, a-3973, a-3974, a-3979  
\Declare@Dfng@inner, a-3981, a-3984, a-3991  
\DeclareBoolOption, a-4506, a-4517  
\DeclareCommand, a-2669, a-4441, a-5109, a-5118, a-5194, a-5205, a-5243, a-5255, a-5309, a-6456, a-6594, a-6678, a-7672, a-8209, c-1605, c-2237, c-2237, c-2276, c-2280, c-2363, c-2390, c-2432, c-2510, c-2787, c-2980, c-2982, c-3067, c-3235, c-3311, c-3763, c-4158, c-4334, c-4430, c-4436, c-4514, c-4516, c-4552, c-4604, c-4656, c-4951, c-4987, c-5139, c-5280, c-5349,

**File Key:** a=gmdoc.sty, b=gmocc.cls, c=gmutils.sty, d=gmiflink.sty, e=gmverb.sty, f=gmoldcomm.sty

c-5488, c-5861,  
 c-5877, c-6223,  
 c-6246, c-6355,  
 c-6384, c-6401,  
 c-6410, c-6439,  
 c-6462, c-6494,  
 c-6573, c-6611,  
 c-6927, c-6953,  
 c-6998, c-7030,  
 c-7048, e-753, e-998,  
 e-1072, e-1119  
 \DeclareComplementaryOption,  
 a-4507, a-4518  
 \DeclareDefining, 13,  
 a-3970, a-4397,  
 a-4398, a-4399,  
 a-4400, a-4431,  
 a-4432, a-4433,  
 a-4434, a-4435,  
 a-4436, a-4437,  
 a-4438, a-4439,  
 a-4440, a-4441,  
 a-4445, a-4446,  
 a-4447, a-4448,  
 a-4449, a-4450,  
 a-4452, a-4453,  
 a-4454, a-4463,  
 a-4466, a-4467,  
 a-4470, a-4472,  
 a-4505, a-4506,  
 a-4507, a-4508  
 \DeclareDocumentCommand,  
 a-4440  
 \DeclareDOXHead, 13,  
 a-4481  
 \DeclareEnvironment,  
 a-7626, c-2390,  
 c-6003, 146  
 \DeclareFontFamily,  
 c-4461, c-4474,  
 c-4486, c-4498  
 \DeclareFontShape,  
 c-4463, c-4476,  
 c-4488, c-4500  
 \DeclareKVOFam, 13, a-4512  
 \DeclareLogo, c-3971,  
 c-3991, c-4018,  
 c-4029, c-4059,  
 c-4065, c-4068,  
 c-4070, c-4073,  
 c-4076, c-4083,  
 c-4085, c-4086,  
 c-4090, c-4097, c-4109  
 \DeclareMathVersion,  
 c-4443

\DeclareOption, a-2119,  
 a-2126, a-2134,  
 a-2139, a-2149,  
 a-2154, a-2163,  
 a-2188, a-2190,  
 a-2200, a-2205,  
 a-4454, c-215  
 \DeclareOptionX,  
 a-4472, a-4484,  
 a-4491, a-4496,  
 b-183, b-367  
 \DeclareRobustCommand,  
 mand, a-4448,  
 c-2867, c-2868,  
 c-2869, c-2870, d-161,  
 d-178, d-187  
 \DeclareStringOption,  
 a-4505,  
 a-4516  
 \DeclareSymbolFont,  
 c-4455, c-4458,  
 c-4472, c-4484, c-4496  
 \DeclareTextCommand,  
 a-4449, c-3984  
 \DeclareTextCommandDefault,  
 a-4450,  
 c-3986  
 \DeclareUrlCommand,  
 c-6921  
 \DeclareVoidOption,  
 a-4508, a-4519  
 \defaultfontfeatures,  
 b-303  
 \DefaultIndexExclusions, 17, a-5680,  
 a-5816, a-5844  
 \DefEntry, 21, a-4783, a-8073  
 \DefIndex, 16, a-4802, a-8023  
 \Define, 14, a-5090  
 \define@boolkey,  
 a-4033, a-4467  
 \define@choicekey,  
 a-4090, a-4470  
 \define@key, a-4042,  
 a-4057, a-4078, a-4466  
 \definecolor, a-2226  
 \DefineTypeChar, e-753,  
 e-790, e-793  
 \defLowStarFake,  
 c-7027, c-7030  
 \defobeylines, c-5417  
 \dekbigskip, c-5430  
 \dekclubs, 12, b-529,  
 e-859, 237  
 \dekclubs\*, 237  
 \dekfracc, c-4265

\dekfracc@args, c-4244,  
 c-4254, c-4268, c-5723  
 \dekfracsimple,  
 c-5722, c-5728  
 \dekfracslash, c-5712,  
 c-5725, c-5726  
 \dekmedbigskip, c-5428  
 \dekmedskip, c-5429  
 \deksmallskip, c-5426  
 \DeleteShortVerb, 12,  
 e-515, 237  
 \Delta, c-4767  
 \delta, c-4785  
 \Describe, 15, a-7847  
 \Describe@Env, a-7835,  
 a-7842, a-7852, a-7865  
 \Describe@Macro,  
 a-7835, a-7852, a-7857  
 \DescribeEnv, a-7840, 113  
 \DescribeMacro, a-7832, 113  
 \detoken@xa, c-2536,  
 c-2537, c-2551,  
 c-2552, c-4128, e-761  
 \detokenize, a-3913,  
 a-3927, a-4349,  
 a-4652, a-4655,  
 a-4657, a-4700,  
 a-4701, a-4713,  
 a-4714, a-4828,  
 a-4850, a-4857,  
 a-4937, a-6747,  
 a-7708, a-7709,  
 c-1687, c-2193,  
 c-2201, c-2214,  
 c-2215, c-2223,  
 c-2226, c-2227,  
 c-2737, c-2741,  
 c-4128, c-4245,  
 c-4247, c-4309,  
 c-4312, c-4320,  
 c-4321, c-5236,  
 c-5249, c-5252,  
 c-5307, c-5308,  
 c-5317, c-5318,  
 c-5328, c-5329, e-1099  
 \dilikern, c-5066  
 \dimexpr, c-4857, c-4862,  
 c-4870, c-4901,  
 c-4996, c-5194,  
 c-5204, c-5215,  
 c-5262, c-5272,  
 c-5277, c-5367,  
 c-5368, c-5503,  
 c-5506, c-6404,  
 c-6416, c-6448,  
 c-6956, c-6960

File Key: a=gmdoc.sty, b=gmocc.cls, c=gmutils.sty, d=gmiflink.sty, e=gmverb.sty,  
 f=gmoldcomm.sty

\DisableCrossrefs,  
a-7970, a-7973  
\discre, a-7550, c-2961,  
c-2964, c-2970,  
c-2991, c-3017, c-3034  
\discret, c-2964  
\disobeylines, c-6478,  
c-7008  
\divide, c-4009, c-4012,  
c-4013, c-5952,  
c-5953, c-5958, c-6538  
\division, 23, a-6955, a-7585  
\Do@Index, a-5823, a-5830  
\do@noligs, e-446, e-850  
\do@properindex,  
a-4913, a-4990, a-5332  
\do@url@hyp, c-6828  
\dobreakblankspace, e-431  
\dobreakbslash, e-413,  
e-448  
\dobreakbrace, e-369,  
e-448  
\dobreakspace, e-449, e-475  
\dobreakvisi!  
blespace, e-429,  
e-475  
\Doc@Input, a-8151  
\DocInclude, 8, 10, 25,  
a-2017, a-2024,  
a-2025, a-2026,  
a-2028, a-6678,  
a-6722, a-6727,  
a-6742, a-6750,  
a-6976, a-8226  
\DocInput, 8, a-2476,  
a-7001, a-7028, a-7343  
DocInputsCount, a-3276  
\docstrips@percent,  
a-5412  
\document, c-3375  
\documentclass, a-1984  
\DoIndex, 17, a-2012,  
a-5823, a-5842  
\DoNot@Index, a-5628,  
a-5636  
\DoNotIndex, 16, a-1989,  
a-5628, a-5840,  
a-5842, a-5846, b-545  
\dont@index, a-5640,  
a-5643, a-5651,  
a-5662, a-5830  
\DontCheckModules, a-8093  
\oprivateothers,  
a-3599, a-3600,  
a-3690, a-3694  
\dots, c-6413  
\downarrow, c-4750  
\dp, c-4898, c-4901, c-4996  
\ds, 23, a-7555  
\dywiz, c-6178  
\acute, b-439, b-454  
\edverbs, b-533, e-868,  
e-873, 237  
\eequals, c-5534  
\eg@MakeShortVerb,  
e-913, e-917  
\eg@MakeShortVerbStar,  
e-913, e-916  
\egCode@MarginizeEnvir,  
a-5013, a-5033  
\egCode@MarginizeMacro,  
a-5014, a-5020  
\egfirstofone, c-372, c-374  
\egRestore@Macro,  
c-1111, c-1113  
\egRestore@MacroSt,  
c-1111, c-1114  
\egStore@Macro, c-1031,  
c-1036  
\egStore@MacroSt,  
c-1031, c-1037  
\egText@MarginizeCS,  
a-5275, a-5283  
\egText@MarginizeEnv,  
a-5275, a-5278  
\em, c-5110, c-5120  
\emdash, c-6128  
\emptyify, a-2609, a-2624,  
a-2833, a-2852,  
a-4155, a-4562,  
a-5403, a-7661,  
a-7662, a-7755,  
c-600, c-1627, c-1713,  
c-1902, c-2299,  
c-2302, c-2304,  
c-2385, c-4194,  
c-4416, c-4512,  
c-4967, c-4971,  
c-5275, c-5602,  
c-5636, c-6521,  
c-6659, c-6923, e-741  
\EnableCrossrefs,  
a-6865, a-7972  
\encapchar, 21, a-3655,  
a-3852, a-4681,  
a-5145, a-6155  
\encodingdefault,  
c-4456, c-4459,  
c-4461, c-4463,  
c-4466, c-4473,  
c-4474, c-4476,  
c-4485, c-4486,  
c-4488, c-4497,  
c-4498, c-4500, c-4503  
\endenumargs, a-7774  
\endenumerate, a-7767  
\endenvironment, a-5613  
\endfilepart, c-5853,  
c-5855, c-5861,  
c-5873, c-5877  
\endgraf, c-6479, c-6485,  
c-6664, c-6671,  
c-6976, c-7013  
\endlinechar, a-7280,  
a-7292, a-7302,  
c-305, c-5161, c-6682,  
c-6752, c-6753,  
c-6759, e-620, e-1018,  
e-1019, e-1029, e-1085  
\endlist, c-3938, c-3955  
\endmacro, a-5524  
\endmacrocode, a-5390  
\endoldmc, a-5390  
\endskiplines, 26  
\endtheglossary, a-6867  
\endverbatim, e-599  
\englishdate, c-6245  
\enoughpage, c-5488  
\enspace, a-2316, b-515,  
c-6654  
\ensuremath, b-543,  
c-2902, c-2918,  
c-4078, c-4328,  
c-4647, c-5623,  
c-6196, c-6203  
\EntryPrefix, 20,  
a-4672, a-4674,  
a-4717, a-5139,  
a-5141, a-5943, a-6691  
\enumargs, a-7774  
\enumargs, 24  
\enumargs\*, 24, a-7771  
\enumerate, a-7644  
\enumerate\*, c-3923  
\env, 22, a-7653, c-3101  
\envhack, c-2244, c-2300,  
c-2394, c-2426  
\envirlet, c-1244  
\environment, a-5612  
\environment, 16, a-5612  
\envirs@toindex,  
a-4860, a-5050,  
a-5054, a-5055,  
a-5083, a-5226  
\envirs@tomarginpar,  
a-5044, a-5047,  
a-5048, a-5082, a-5220  
\EOF, a-8312

File Key: a=gmdoc.sty, b=gmocc.cls, c=gmutils.sty, d=gmiflink.sty, e=gmverb.sty,  
f=gmoldcomm.sty

\EOFMark, 20, a-2516,  
a-2718, a-7342,  
a-8308, b-543, 121  
\EOLwasQueer, a-6590,  
a-6601  
\epsilon, c-4786  
>equals, c-5532  
\errorcontextlines,  
a-1981, b-482  
\eta, c-4789  
\TeX, 23, c-4076, c-4081,  
c-4083, c-4177  
\evensidemargin, a-6646  
\everydisplay, c-4939,  
c-4961, c-4965  
\everyeof, 20, a-2530, e-620  
\everymath, c-4913,  
c-4939, c-4942,  
c-4960, c-4961,  
c-4964, c-4965  
\everypar, a-2508,  
a-2508, a-2567,  
a-2769, a-2770,  
a-2790, a-2877,  
a-3144, a-3165,  
a-3208, a-3227,  
a-3404, a-3412,  
a-5597, a-7396,  
c-3665, c-3675,  
c-6034, e-691  
\ExecuteOptionsX, b-184  
\exhyphenpenalty,  
b-537, e-583, e-587  
\exists, c-4872  
\f@encoding, c-5571  
\f@family, c-4421  
\f@series, c-4019  
\f@size, c-6960, e-761  
\fakeonum@ii, c-5157,  
c-5159  
\fakeonum, c-5139  
\fakern, c-4934  
\fakesc@extrascale,  
c-5956, c-5971  
\fakescaps, c-5931  
\fakescapscore, c-5909,  
c-5933  
\fakescextrascale, c-5971  
\file, 23, c-3039, c-6921  
\filedate, 24, a-6946,  
a-7229, a-7320  
\filediv, a-6890, a-6907,  
a-6954, a-6972,  
a-7125, a-7188  
\filedivname, a-6891,  
a-6901, a-6904,  
a-6908, a-6921,  
a-6923, a-6952,  
a-6971, a-7126  
\FileInfo, 24, a-7244  
\fileinfo, 24, a-7231  
\filekey, a-6823, a-6926,  
a-6929  
\filename, a-6945, a-7227  
\filenote, 24, a-7320, a-7322  
\filepart, c-5818, c-5820,  
c-5876  
\filesep, a-6690, a-6691,  
a-6870, a-6925  
\fileversion, 24,  
a-6572, a-6576,  
a-6947, a-7230, a-7320  
\Finale, 22, a-7982, a-8011  
\finish@macroscan,  
a-3742, a-3756,  
a-3770, a-3781,  
a-3860, f-36, f-37, f-64  
\Finv, b-447  
\fontchardp, c-7035  
\fontcharht, c-5191, c-5267  
\fontcharwd, c-5198, c-6382  
\fontencoding, e-471  
\fontseries, b-495  
\fontspec, b-497, c-766,  
c-5309, c-5310,  
c-5319, c-5320,  
c-5330, c-5331,  
c-5358, c-5362  
\fontspec, b-353  
\fooatletter, c-376  
\foone, a-2651, a-2868,  
a-3430, a-3464,  
a-3502, a-3557,  
a-3708, a-4173,  
a-4263, a-4306,  
a-5422, a-5436,  
a-6026, a-6070,  
a-7256, a-7286,  
a-7333, a-7382,  
a-7801, a-8306,  
c-372, c-376, c-670,  
c-674, c-677, c-681,  
c-683, c-685, c-691,  
c-693, c-702, c-706,  
c-722, c-742, c-746,  
c-749, c-752, c-772,  
c-774, c-925, c-970,  
c-1008, c-3105,  
c-3216, c-3220,  
c-3280, c-5408,  
c-5416, c-6155,  
c-6180, c-7007, e-367,  
e-381, e-410, e-427,  
e-435, e-1063, e-1071,  
e-1129, f-51, f-82  
\forall, c-4866  
\FormatHangHeading,  
c-3772, c-3780,  
c-3788, c-3795  
\FormatRunInHeading,  
c-3802, c-3807  
\freeze@actives, c-5556  
\fullcurrentfile,  
a-6717, a-6748, a-6798  
\fullpar, c-6355  
\fullparcore, c-6363,  
c-6367  
\g@emptyify, a-2713,  
a-3795, a-5048,  
a-5055, a-6429,  
a-6813, a-7066,  
a-7191, a-7365, c-604,  
c-605  
\g@relaxen, a-3926,  
a-4701, a-4704,  
a-4714, a-5590,  
a-7067, a-7190, c-613,  
c-614  
\gaddtomacro, 21, a-2713,  
a-3588, a-4005,  
a-5220, a-5226, c-571  
\gag@index, a-2261,  
a-5169, a-7948, a-7970  
\Game, b-447  
\Gamma, c-4766  
\gamma, c-4784  
\garamath, c-4987, 188  
\ge, c-4689, c-4690, c-4691,  
c-4703  
\geeng, c-4691  
\emptyify, c-605  
\GeneralName, a-6199,  
a-6200, a-6229,  
a-6283, a-6696  
\generalname, a-6161,  
a-6167, a-6216,  
a-6325, a-6444  
\geometry, b-476  
\geq, c-4690, c-4701,  
c-4702, c-4703  
\GetFileInfo, 24,  
a-6796, a-6944, a-7226  
\getprevdepth, c-6480,  
c-6975  
\gimel, b-447  
\glet, a-2537, a-2939,  
a-3794, a-4828,  
a-5028, a-5406,  
a-6027, a-6071,

File Key: a=gmdoc.sty, b=gmdocc.cls, c=gmutils.sty, d=gmiflink.sty, e=gmverb.sty,  
f=gmoldcomm.sty

|                                                                                                                                                                                                                                                                                              |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                    |                                                   |
|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|---------------------------------------------------|
| a-6931, a-6936,<br>a-6950, c-563, c-3689                                                                                                                                                                                                                                                     | a-4652, a-4655,<br>a-4657, c-623, c-1130,<br>c-1142, c-3483,<br>c-3516, c-3532,<br>c-3608, c-3629,<br>c-3684, c-3697,<br>c-3768, c-3897,<br>c-4055, c-4089,<br>c-4091, c-4096,<br>c-4098, c-4245,<br>c-5955, c-5978                                                                                                                                                                                                                                                                                                                | \gma@gmathhook, c-4937,<br>c-4971, c-4972, c-4998 |
| \glossary@prologue,<br>a-6281, a-6361,<br>a-6399, a-6406, a-6933                                                                                                                                                                                                                             | \gma@quantifierhook,<br>c-4866, c-4872,<br>c-4967, c-4969,<br>c-5006, c-5014, c-5015                                                                                                                                                                                                                                                                                                                                                                                                                                               |                                                   |
| \glossaryentry, a-5145                                                                                                                                                                                                                                                                       | \gma@tempa, c-4855,<br>c-4857, c-4860,<br>c-4862, c-4897,<br>c-4901, c-4929, c-4932                                                                                                                                                                                                                                                                                                                                                                                                                                                |                                                   |
| \GlossaryMin, 18, a-6350,<br>a-6350, a-6361                                                                                                                                                                                                                                                  | \gma@tempb, c-4898, c-4901                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                         |                                                   |
| \GlossaryParms, 18,<br>a-6362, a-6413                                                                                                                                                                                                                                                        | \gmath, b-305, c-4951,<br>c-4975, c-4979, 188                                                                                                                                                                                                                                                                                                                                                                                                                                                                                      |                                                   |
| \GlossaryPrologue, 18,<br>a-6398                                                                                                                                                                                                                                                             | \gmath@delc, c-4604, c-4635                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                        |                                                   |
| \glueexpr, a-2424,<br>a-2432, a-7646,<br>c-5428, c-6327,<br>c-6405, c-6418                                                                                                                                                                                                                   | \gmath@delcif, c-4629                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                              |                                                   |
| \gluestretch, c-6328                                                                                                                                                                                                                                                                         | \gmath@delimif, c-4638                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                             |                                                   |
| \glyphname, c-5017, c-6198                                                                                                                                                                                                                                                                   | \gmath@do, c-4516, c-4567,<br>c-4570, c-4573,<br>c-4672, c-4674,<br>c-4675, c-4676,<br>c-4677, c-4678,<br>c-4679, c-4680,<br>c-4681, c-4682,<br>c-4683, c-4684,<br>c-4706, c-4707,<br>c-4708, c-4709,<br>c-4710, c-4713,<br>c-4715, c-4717,<br>c-4721, c-4731,<br>c-4732, c-4734, c-4758                                                                                                                                                                                                                                           |                                                   |
| \gm@clearpagesduetoopenright<br>c-3469, c-3532                                                                                                                                                                                                                                               | \gmath@doif, c-4552,<br>c-4589, c-4673,<br>c-4686, c-4689,<br>c-4692, c-4693,<br>c-4720, c-4722,<br>c-4723, c-4724,<br>c-4725, c-4726,<br>c-4727, c-4729,<br>c-4730, c-4736,<br>c-4737, c-4738,<br>c-4739, c-4740,<br>c-4741, c-4742,<br>c-4743, c-4746,<br>c-4750, c-4752,<br>c-4759, c-4762,<br>c-4763, c-4764,<br>c-4766, c-4767,<br>c-4768, c-4769,<br>c-4770, c-4772,<br>c-4773, c-4774,<br>c-4775, c-4776,<br>c-4782, c-4783,<br>c-4784, c-4785,<br>c-4786, c-4787,<br>c-4788, c-4789,<br>c-4790, c-4791,<br>c-4792, c-4793, |                                                   |
| \gm@dontnumbersectionsoutof<br>c-3467, c-3516                                                                                                                                                                                                                                                | \gmath@sec, c-3890, c-3901,<br>c-3902                                                                                                                                                                                                                                                                                                                                                                                                                                                                                              |                                                   |
| \gm@DOX, b-183, b-196,<br>b-203, b-206, b-210,<br>b-213, b-221, b-227,<br>b-232, b-239, b-265,<br>b-274, b-314, b-315,<br>b-318, b-320, b-323,<br>b-326, b-330, b-353                                                                                                                        | \gmath@secmark, c-3854,<br>c-3874, c-3880,<br>c-3886, c-3899                                                                                                                                                                                                                                                                                                                                                                                                                                                                       |                                                   |
| \gm@EOX, b-184, b-357                                                                                                                                                                                                                                                                        | \gmath@secmarkh, c-3882                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                            |                                                   |
| \gm@gobmacro, c-4138,<br>c-4144                                                                                                                                                                                                                                                              | \gmath@secstar, c-3858,<br>c-3868, c-3875,<br>c-3887, c-3901, c-3902                                                                                                                                                                                                                                                                                                                                                                                                                                                               |                                                   |
| \gm@hyperrefstepcounter,<br>c-3459, c-3462                                                                                                                                                                                                                                                   | \gmath@secx, c-3890, c-3891                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                        |                                                   |
| \gm@hypertarget, d-162,<br>d-165                                                                                                                                                                                                                                                             | \gmath@secxx, c-3857,<br>c-3885, c-3892                                                                                                                                                                                                                                                                                                                                                                                                                                                                                            |                                                   |
| \gm@iflink, d-187, d-188,<br>d-190                                                                                                                                                                                                                                                           | \gmath@smarttilde, c-6953,<br>c-6971                                                                                                                                                                                                                                                                                                                                                                                                                                                                                               |                                                   |
| \gm@ifnac, c-889, c-892                                                                                                                                                                                                                                                                      | \gmath@straightensec,<br>c-3896, c-3905                                                                                                                                                                                                                                                                                                                                                                                                                                                                                            |                                                   |
| \gm@ifref, d-178, d-179,<br>d-181                                                                                                                                                                                                                                                            | \gmath@targetheading,<br>c-3460, c-3463                                                                                                                                                                                                                                                                                                                                                                                                                                                                                            |                                                   |
| \gm@ifstar, a-2296,<br>a-3972, a-4422,<br>a-4589, a-4804,<br>a-4865, a-4888,<br>a-4958, a-4995,<br>a-5012, a-5098,<br>a-5103, a-5238,<br>a-5274, a-7441,<br>a-7852, c-935, c-1031,<br>c-1111, c-2676,<br>c-3901, c-4199,<br>c-4250, c-6275,<br>c-6486, c-6603,<br>e-485, e-735, e-859, e-913 | \gmath@testdefined, c-641,<br>c-659, c-2273, e-1046                                                                                                                                                                                                                                                                                                                                                                                                                                                                                |                                                   |
| \gm@ifundefined,<br>a-3865, a-3913,                                                                                                                                                                                                                                                          | \gmath@testundefined, c-656                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                        |                                                   |
|                                                                                                                                                                                                                                                                                              | \gmath@UrlFix, c-6709,<br>c-6714, c-6728                                                                                                                                                                                                                                                                                                                                                                                                                                                                                           |                                                   |
|                                                                                                                                                                                                                                                                                              | \gmath@UrlSetup, c-6750,<br>c-6796                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                 |                                                   |
|                                                                                                                                                                                                                                                                                              | \gmath@verb@eol, a-3613,<br>a-7486, e-734, e-814,<br>e-836                                                                                                                                                                                                                                                                                                                                                                                                                                                                         |                                                   |
|                                                                                                                                                                                                                                                                                              | \gmath@xistar, a-5423, a-5426                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                      |                                                   |
|                                                                                                                                                                                                                                                                                              | \gma, c-4980                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                       |                                                   |
|                                                                                                                                                                                                                                                                                              | \gma@arrowdash, c-4994,<br>c-5004, c-5010, c-5012                                                                                                                                                                                                                                                                                                                                                                                                                                                                                  |                                                   |
|                                                                                                                                                                                                                                                                                              | \gma@bare, c-4976, c-4978                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                          |                                                   |
|                                                                                                                                                                                                                                                                                              | \gma@bracket, c-4978,<br>c-4979                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                    |                                                   |
|                                                                                                                                                                                                                                                                                              | \gma@checkbracket,<br>c-4977, c-4981                                                                                                                                                                                                                                                                                                                                                                                                                                                                                               |                                                   |
|                                                                                                                                                                                                                                                                                              | \gma@dollar, c-4975,<br>c-4976, c-4981                                                                                                                                                                                                                                                                                                                                                                                                                                                                                             |                                                   |

File Key: a=gmdoc.sty, b=gmdocc.cls, c=gmutils.sty, d=gmiflink.sty, e=gmverb.sty,  
f=gmoldcomm.sty

c-4794, c-4795,  
 c-4796, c-4797,  
 c-4798, c-4799,  
 c-4800, c-4801,  
 c-4802, c-4803,  
 c-4804, c-4805,  
 c-4806, c-4807, c-4808  
 $\backslash\text{gmath@fam}$ , c-4430,  
 c-4520, c-4558,  
 c-4564, c-4665, c-4778  
 $\backslash\text{gmath@famm}$ , c-4667,  
 c-4673, c-4686,  
 c-4689, c-4692,  
 c-4693, c-4720,  
 c-4722, c-4723,  
 c-4724, c-4725,  
 c-4726, c-4727,  
 c-4729, c-4730,  
 c-4736, c-4737,  
 c-4738, c-4739,  
 c-4740, c-4741,  
 c-4742, c-4743,  
 c-4746, c-4750,  
 c-4752, c-4759,  
 c-4762, c-4763,  
 c-4764, c-4766,  
 c-4767, c-4768,  
 c-4769, c-4770,  
 c-4772, c-4773,  
 c-4774, c-4775,  
 c-4776, c-4780,  
 c-4782, c-4783,  
 c-4784, c-4785,  
 c-4786, c-4787,  
 c-4788, c-4789,  
 c-4790, c-4791,  
 c-4792, c-4793,  
 c-4794, c-4795,  
 c-4796, c-4797,  
 c-4798, c-4799,  
 c-4800, c-4801,  
 c-4802, c-4803,  
 c-4804, c-4805,  
 c-4806, c-4807, c-4808  
 $\backslash\text{gmath@famnum}$ , c-4431,  
 c-4527, c-4543,  
 c-4613, c-4620, c-4649  
 $\backslash\text{gmath@font}$ , c-4635,  
 c-4645, c-4756,  
 c-4757, c-4811,  
 c-4812, c-4865,  
 c-4879, c-4882,  
 c-4886, c-4887, c-4888  
 $\backslash\text{gmath@ft}$ , c-4562,  
 c-4567, c-4570, c-4573  
 $\backslash\text{gmath@getfamnum}$ ,  
 c-4430, c-4523,  
 c-4610, c-4646  
 $\backslash\text{gmath@restore}$ , c-4575,  
 c-4577, c-4579, c-4585  
 $\backslash\text{gmath@version}$ , c-4565,  
 c-4662, c-4667,  
 c-4778, c-4780  
 $\backslash\text{gmathbase}$ , c-4514,  
 c-4828, c-4830,  
 c-4837, c-4952  
 $\backslash\text{gmathcats}$ , c-4963  
 $\backslash\text{gmathFams}$ , c-4436, c-4515  
 $\backslash\text{gmathfamshook}$ , c-4509,  
 c-4512  
 $\backslash\text{gmathfurther}$ , c-4844,  
 c-4953  
 $\backslash\text{gmathhook}$ , c-4972  
 $\backslash\text{gmathscripts}$ , c-4959  
 $\backslash\text{gmboxedspace}$ , a-7446,  
 a-7449, a-7503, a-7519  
 $\backslash\text{gmcc@article}$ , b-213  
 $\backslash\text{gmcc@CLASS}$ , b-198,  
 b-200, b-387, b-395  
 $\backslash\text{gmcc@class}$ , b-196,  
 b-203, b-206, b-210,  
 b-213  
 $\backslash\text{gmcc@cronos}$ , b-318  
 $\backslash\text{gmcc@cursor}$ , b-330  
 $\backslash\text{gmcc@debug}$ , b-227  
 $\backslash\text{gmcc@dff}$ , b-292, b-293,  
 b-303  
 $\backslash\text{gmcc@fontspec}$ , b-353  
 $\backslash\text{gmcc@lsu}$ , b-326  
 $\backslash\text{gmcc@minion}$ , b-314  
 $\backslash\text{gmcc@mptt}$ , b-274  
 $\backslash\text{gmcc@mwart}$ , b-203  
 $\backslash\text{gmcc@mwbk}$ , b-210  
 $\backslash\text{gmcc@mwcclsfalse}$ , b-387  
 $\backslash\text{gmcc@mwcclstrue}$ , b-200  
 $\backslash\text{gmcc@mwrrep}$ , b-206  
 $\backslash\text{gmcc@myriad}$ , b-323  
 $\backslash\text{gmcc@nochanges}$ , b-239  
 $\backslash\text{gmcc@noindex}$ , b-232  
 $\backslash\text{gmcc@oldfontsfalse}$ ,  
 b-265, b-287  
 $\backslash\text{gmcc@oldfontstrue}$ , b-415  
 $\backslash\text{gmcc@outeroff}$ , b-221  
 $\backslash\text{gmcc@pagella}$ , b-315  
 $\backslash\text{gmcc@resa}$ , b-199, b-200  
 $\backslash\text{gmcc@setfont}$ , b-286,  
 b-314, b-315  
 $\backslash\text{gmcc@sysfonts}$ , b-265  
 $\backslash\text{gmcc@tout}$ , b-284, b-291,  
 b-309  
 $\backslash\text{gmcc@trebuchet}$ , b-320  
 $\backslash\text{gmd@toc}$ , a-2629,  
 a-2631, a-2632  
 $\backslash\text{gmd@ABIOnce}$ , a-2695,  
 a-2696, a-2713, a-6813  
 $\backslash\text{gmd@adef@altindex}$ ,  
 a-4354, a-4364,  
 a-4365, a-4367,  
 a-4368, a-4371, a-4373  
 $\backslash\text{gmd@adef@checkDOXopts}$ ,  
 a-4202, a-4218  
 $\backslash\text{gmd@adef@checkIbracket}$ ,  
 a-4187, a-4208  
 $\backslash\text{gmd@adef@cs}$ , a-4163  
 $\backslash\text{gmd@adef@cshookfalse}$ ,  
 a-3870  
 $\backslash\text{gmd@adef@cshooktrue}$ ,  
 a-4163  
 $\backslash\text{gmd@adef@currdef}$ ,  
 a-3996, a-4002,  
 a-4006, a-4010,  
 a-4011, a-4013,  
 a-4015, a-4018,  
 a-4021, a-4044,  
 a-4061, a-4067,  
 a-4080, a-4082,  
 a-4149, a-4230,  
 a-4235, a-4334,  
 a-4340, a-4355,  
 a-4358, a-4366, a-4369  
 $\backslash\text{gmd@adef@defaulttype}$ ,  
 a-3973, a-3974, a-3993  
 $\backslash\text{gmd@adef@deftext}$ ,  
 a-4327, a-4347  
 $\backslash\text{gmd@adef@dfKVpref}$ ,  
 a-4184, a-4197,  
 a-4225, a-4229  
 $\backslash\text{gmd@adef@dk}$ , a-4179  
 $\backslash\text{gmd@adef@dofam}$ ,  
 a-4200, a-4259,  
 a-4267, a-4317, a-4333  
 $\backslash\text{gmd@adef@dox}$ , a-4190  
 $\backslash\text{gmd@adef@fam}$ , a-4199,  
 a-4257, a-4260,  
 a-4265, a-4268,  
 a-4315, a-4318,  
 a-4341, a-4342  
 $\backslash\text{gmd@adef@indextext}$ ,  
 a-4353, a-4374, a-4377  
 $\backslash\text{gmd@adef@KVfam}$ , a-4078  
 $\backslash\text{gmd@adef@KVpref}$ , a-4057  
 $\backslash\text{gmd@adef@prefix}$ , a-4042  
 $\backslash\text{gmd@adef@scanDKfam}$ ,  
 a-4294, a-4314  
 $\backslash\text{gmd@adef@scanDOXfam}$ ,  
 a-4192, a-4220, a-4243  
 $\backslash\text{gmd@adef@scanfamact}$ ,  
 a-4248, a-4264  
 $\backslash\text{gmd@adef@scanfamoth}$ ,  
 a-4245, a-4256

**File Key:** a=gmdoc.sty, b=gmdocc.cls, c=gmutils.sty, d=gmiflink.sty, e=gmverb.sty,  
 f=gmoldcomm.sty

\gmd@adef@scanKvpref,  
a-4180, a-4191,  
a-4210, a-4219, a-4224  
\gmd@adef@scancode,  
a-4290, a-4298, a-4322  
\gmd@adef@selfrestore,  
a-4412, a-4416,  
a-4599, a-4602  
\gmd@adef@setkeysdefault,  
a-3999, a-4026  
\gmd@adef@setKV,  
a-4062, a-4086,  
a-4144, a-4147  
\gmd@adef@settype,  
a-4114, a-4116,  
a-4118, a-4120,  
a-4122, a-4124,  
a-4126, a-4128,  
a-4130, a-4132, a-4140  
\gmd@adef@text, a-4171  
\gmd@adef@TYPE, a-4017,  
a-4141  
\gmd@adef@type, a-4090  
\gmd@adef@typenr,  
a-4091, a-4113  
\gmd@adef@typevals,  
a-4091  
\gmd@auxext, a-6708,  
a-6710, a-6764, a-6775  
\gmd@blubra, a-7744,  
a-7755, a-7757,  
a-7758, a-7759  
\gmd@bslashEOL, a-3510,  
a-3559, a-3562  
\gmd@changes@init,  
a-6136, a-6139,  
a-6438, a-6453, a-6591  
\gmd@charbychar,  
a-2899, a-2955,  
a-3036, a-3091,  
a-3898, a-4163,  
a-4171, a-4210,  
a-4220, a-4226,  
a-4261, a-4269,  
a-4319, a-4329, a-4615  
\gmd@checkifEOL,  
a-2771, a-3142  
\gmd@checkifEOLmixed,  
a-3047, a-3163  
\gmd@chgs, a-6453, a-6456  
\gmd@chgs@parse,  
a-6442, a-6462,  
a-6464, a-6467  
\gmd@chschangeline,  
a-6529, a-6537,  
a-6546, a-6570

\gmd@closingspacewd,  
a-2884, a-3538,  
a-3548, a-3550  
\gmd@codecheckifds,  
a-5484  
\gmd@codeskip, a-2910,  
a-3199, a-3316,  
a-3343, a-3371  
\gmd@continuenarration,  
a-2603, a-2766, a-3019  
\gmd@countnarrline@,  
a-2788, a-2828  
\gmd@counttheline,  
a-3059, a-3091, a-3098  
\gmd@cpnarrline,  
a-2768, a-2826,  
a-2833, a-2848,  
a-3143, a-3164, a-3593  
\gmd@ctallsetup,  
a-2836, a-2852,  
a-5405, a-7246  
\gmd@currentlabel@before,  
a-2481, a-2537  
\gmd@currenvxistar,  
a-5419, a-5425  
\gmd@DefineChanges,  
a-6134, a-6337  
\gmd@detectors, a-3899,  
a-4004, a-4005,  
a-4155, a-4559,  
a-4562, a-4570, a-4703  
\gmd@filename,  
a-6704, a-6707  
\gmd@dip@hook, a-5883,  
a-5890, a-5891  
\gmd@doc@includeaux,  
a-6716, a-6884, a-6886  
\gmd@docstripdirective,  
a-3136, a-3157,  
a-5488, a-6029  
\gmd@docstripinner,  
a-6037, a-6039  
\gmd@docstripshook,  
a-6079  
\gmd@docstripverb,  
a-6036, a-6074  
\gmd@doindextext,  
a-4381, a-5053, a-5058  
\gmd@doIndexRelated,  
a-6788, a-6805, a-6864  
\gmd@dolspaces, a-2604,  
a-2899, a-2986  
\gmd@DoTeXCodeSpace,  
a-2587, a-2732,  
a-2741, a-2746, a-5408  
\gmd@ea@bwrap, a-7655,  
a-7662, a-7665

a-7688, a-7696,  
a-7700, a-7704,  
a-7708, a-7712,  
a-7717, a-7745, a-7773  
\gmd@ea@ewrap, a-7658,  
a-7661, a-7665,  
a-7689, a-7697,  
a-7701, a-7705,  
a-7709, a-7713,  
a-7718, a-7721,  
a-7723, a-7725,  
a-7728, a-7730,  
a-7733, a-7739,  
a-7750, a-7752, a-7773  
\gmd@ea@hashes, a-7648,  
a-7656, a-7669, a-7747  
\gmd@ea@wraps, a-7660,  
a-7663, a-7667, a-7772  
\gmd@ea@xxxwd, a-7670,  
a-7688, a-7717  
\gmd@eatlspace, a-2991,  
a-3010, a-3015  
\gmd@endpe, a-3172,  
a-3177, a-3205,  
a-3212, a-3219  
\gmd@EOLorcharbychar,  
a-3063, a-3078  
\gmd@evpaddonce,  
a-5580, a-5586  
\gmd@fileinfo, a-7253,  
a-7265  
\gmd@finishifstar,  
a-3742, a-3770, a-3780  
\gmd@FIrescan, a-7270,  
a-7287  
\gmd@glossary, a-5159,  
a-5163, a-6198  
\gmd@glossCStest,  
a-6194, a-6197,  
a-6213, a-6222  
\gmd@gobbleuntilM,  
a-3468, a-3469  
\gmd@grefstep, a-2789,  
a-2798, a-2844,  
a-2849, a-2921,  
a-3103, a-3113  
\gmd@guardedinput,  
a-2510, a-2531  
\gmd@iedir, a-5637,  
a-5657, a-5830  
\gmd@ifinmeaning, f-38  
\gmd@ifonetoken,  
a-5507, a-5522,  
a-5557, a-7835  
\gmd@ifsingle, a-5542,  
a-5560

**File Key:** a=gmdoc.sty, b=gmdocc.cls, c=gmutils.sty, d=gmiflink.sty, e=gmverb.sty,  
f=gmoldcomm.sty

\gmd@iihook, a-2515,  
a-2624, a-7340  
\gmd@inputname, a-2479,  
a-4199, a-6526,  
a-6536, a-6543  
\gmd@inverb, a-7444,  
a-7447, a-7471  
\gmd@jobname, a-6703,  
a-6707  
\gmd@justadot, a-4825,  
a-4828, a-4857,  
a-5028, a-5637  
\gmd@KVprefdefault,  
a-4049, a-4057,  
a-4059, a-4184,  
a-4197, a-4481  
\gmd@lastenvir, a-7619  
\gmd@lbracecase,  
a-4171, a-4183,  
a-4196, a-4286,  
a-4289, a-4293,  
a-4297, a-4301  
\gmd@ldspaceswd,  
a-2919, a-2929,  
a-2930, a-2943,  
a-2975, a-2990,  
a-3012, a-3018  
\gmd@maybequote,  
a-3733, a-3754,  
a-3766, a-3794,  
a-3795, a-4947  
gmd@mc, a-7597  
\gmd@mcdiag, a-7601,  
a-7616, a-7618, a-7622  
\gmd@mchook, a-7600  
\gmd@modulehashone,  
a-6041, a-6045,  
a-6076, a-6080  
\gmd@narrcheckifds,  
a-3152, a-3155  
\gmd@narrcheckifds@ne,  
a-3128, a-3134  
\gmd@nlperc, a-7451,  
a-7472, a-7504, a-7520  
\gmd@nocodeskip,  
a-2905, a-2912,  
a-3201, a-3203,  
a-3337, a-3345,  
a-3365, a-3373  
\gmd@oldmcfinis, a-5450  
\gmd@oncenum, a-5587,  
a-5589, a-5591,  
a-5596, a-5598, a-5601  
\gmd@parfixclosingspace,  
a-2870, a-3537  
\gmd@percenthack,  
a-3044, a-3121  
\gmd@preambleABD,  
e-929, e-930, e-937  
\gmd@preverypar,  
a-2327, a-2791,  
a-3147, a-3165,  
a-3209, a-3228,  
a-3402, a-3410, a-3412  
\gmd@providefii,  
a-7304, a-7309  
\gmd@quotationname,  
a-7403, a-7411, a-7415  
\gmd@resa, a-3992,  
a-3994, a-4043,  
a-4046, a-4058,  
a-4059, a-4065,  
a-4066, a-4068,  
a-4071, a-4079,  
a-4081, a-4083,  
a-4085, a-4148,  
a-4151, a-5066,  
a-5069, a-5071  
\gmd@resetlinecount,  
a-2498, a-3270, a-3283  
\gmd@ResumeDfng,  
a-4633, a-4635  
\gmd@revprefix, a-4750,  
a-4752  
\gmd@setChDate, a-6249,  
a-6252, a-6271  
\gmd@setclosingspacewd,  
a-3549  
\gmd@setclubpenalty,  
a-2486, a-2576,  
a-2580, a-2611  
\gmd@setilrr, a-2999,  
a-3190, a-3248, a-3539  
\gmd@skipgmltext,  
a-7364, a-7365, a-7375  
\gmd@skiplines, a-2860,  
a-2863  
\gmd@spacewd, a-2972,  
a-2989, a-3012  
\gmd@texcodeEOL,  
a-2902, a-3080  
\gmd@texcodespace,  
a-2742, a-2748,  
a-2898, a-2983,  
a-2987, a-3011, a-3891  
\gmd@textEOL, a-2553,  
a-2659, a-3148,  
a-3170, a-3505,  
a-5403, a-6045,  
a-6080, a-6604  
\gmd@toCTAN@, a-6439,  
a-6441  
\gmd@typesettexcode,  
a-2869, a-3005, a-3021  
\gmd@writeckpt, a-6809,  
a-6854  
\gmd@writeFI, a-7269,  
a-7278  
\gmd@writemauxinpaux,  
a-6764, a-6829  
\gmd@wykrzykniki,  
a-6542, a-6548, a-6561  
\gmdindexpagecs,  
a-4743, a-4749  
\gmdindexrefcs, a-4740,  
a-4743, a-4747  
\gmdmarginpar, 16, 23,  
a-5298, a-5304, a-5311  
\gmdnoindent, 24, a-7423  
\gmdoccMargins, b-473  
\gmdoccMar|  
gins@params,  
b-469, b-474, b-476  
\gmdocIncludes, 9, a-7027  
gmglo.list, 91  
GMlabel, d-150  
\gmhypertarget, a-3304,  
d-161, 233  
\gmiflink, a-4747, d-187, 233  
\gmifref, d-178, 233  
\gml@StoreCS, c-1076,  
c-1099, c-1159  
\gml@storemacros,  
c-1077, c-1088,  
c-1097, c-1102, c-1162  
\gmlonely, 23, a-7360, a-7372  
\gmobeyspaces, a-2741,  
c-5409, e-585, e-737  
\gmoc@checkenv, f-40, f-54  
\gmoc@checkenvinn,  
f-56, f-58  
\gmoc@defbslash, f-34, f-86  
\gmoc@maccname, f-47, f-60  
\gmoc@notprinted, f-38,  
f-45  
\gmoc@ocname, f-48, f-69  
\gmoc@resa, f-59, f-60, f-69  
\gmshowlists, c-2588  
\GMtextsuperscript,  
c-4332  
\gmu@acroinner, c-5641,  
c-5651, c-5652, c-5660  
\gmu@acrospace,  
c-5631, c-5640,  
c-5640, c-5644  
\gmu@activespace,  
c-772, c-2922  
\gmu@activespaceblank,  
c-2906, c-2921  
\gmu@among@, c-408,  
c-408, c-410

File Key: a=gmdoc.sty, b=gmdocc.cls, c=gmutils.sty, d=gmiflink.sty, e=gmverb.sty,  
f=gmoldcomm.sty

\gmu@AT@ampulex, c-6627, c-6629, c-6633  
\gmu@ATfootnotes, c-6620, c-6626, c-6636, c-6639, c-6643  
\gmu@basefont, c-5345, c-5360  
\gmu@bihyphen@char, c-2985, c-2994  
\gmu@bihyphen@corr, c-2989, c-3001, c-3015  
\gmu@calc@scale, c-4398, c-4403, c-5349  
\gmu@calculateslant, c-5183, c-5259  
\gmu@cepstnof, c-5280, c-5358, c-5362  
\gmu@checkaftersec, c-3683, c-3742  
\gmu@checkM, c-968, c-972  
\gmu@currfont@descaled, c-5359, c-5369  
\gmu@datef, c-6309  
\gmu@datefs1, c-6223, c-6227, c-6246, c-6250, c-6272  
\gmu@dekfrac, c-4222, c-4246, c-4250  
\gmu@dekfraccsimple, c-4250, c-5709, c-5723  
\gmu@denominatorkern, c-4224, c-4272, c-5712  
\gmu@discretionaryhyphen, c-2976, c-2982, c-3010, c-3011, c-6757  
\gmu@discretionaryslash, c-3034, c-3045  
\gmu@disMiner, c-7011, c-7012  
\gmu@dofakeonum, c-5163, c-5169  
\gmu@dogmathbase, c-4656, c-4827, c-4828  
\gmu@doUrlMath, c-6801, c-6803, c-6805, c-6849  
\gmu@doUrlMathAc, c-6806, c-6813, c-6822, c-6831, c-6880  
\gmu@dywiz, c-6177, c-6181  
\gmu@extremnum, c-3130, c-3148, c-3149, c-3153, c-3154  
\gmu@fileext, c-5758, c-5768, c-5786  
\gmu@filename, c-5757, c-5771, c-5783, c-5786, c-5789, c-5798  
\gmu@filepartname, c-5825, c-5833, c-5848, c-5865  
\gmu@fnhook, c-6653, c-6659  
\gmu@fontscale, c-4408, c-5364  
\gmu@fontscalebr, c-4408, c-4409, c-4416, c-4464, c-4477, c-4489, c-4501  
\gmu@fontstring, c-4379, c-4464, c-4477, c-4480, c-4489, c-4501, c-4508  
\gmu@fontstring@, c-4380, c-4382, c-4671  
\gmu@fontstring@®, c-4383, c-4391  
\gmu@forallkerning, c-4867, c-4869  
\gmu@forarg, c-424, c-434  
\gmu@foreach, c-423, c-431, c-431, c-433, c-438  
\gmu@forer, c-432, c-436, c-438  
\gmu@fracfontsetup, c-5711, c-5715, c-5718  
\gmu@getaddvs, c-3734, c-3734, c-3740  
\gmu@gettext, c-5756, c-5766  
\gmu@getfontdata, c-4415, c-4451, c-4470, c-4482, c-4493  
\gmu@getfontscale, c-4393, c-4394, c-4410, c-4420, c-4422  
\gmu@getfontstring, c-4378, c-4423  
\gmu@getslant, c-5248, c-5260  
\gmu@gobdef, c-257, c-263  
\gmu@hashes, c-1988, c-1992, c-2023  
\gmu@hashesbraced, c-1997, c-2000, c-2025  
\gmu@if, c-287, c-1704, c-1706  
\gmu@if@onum, c-5152, c-5234  
\gmu@ifstored, c-1118, c-1129, c-1137, c-4470, c-4482, c-4531  
\gmu@lowstar, c-910, c-930, c-7024, c-7055  
\gmu@lowstarfake, c-911, c-7034, c-7040  
\gmu@luzniej, c-6017, c-6020, c-6022  
\gmu@maxnum, c-3153  
\gmu@maybestripcomma, c-5289, c-5343  
\gmu@measurewd, c-5185, c-5197, c-5213, c-5227, c-6811, c-6816, c-6829, e-765  
\gmu@medmuskip, c-6782, c-6788  
\gmu@minnum, c-3116, c-3154  
\gmu@ms, c-2007, c-2013, c-2017, c-2021, c-2038, c-2039, c-2040, c-2041, c-2042  
\gmu@nl@reserveda, c-1231, c-1234, c-1239, c-1242  
\gmu@nocite@ampulex, c-3366, c-3379  
\gmu@numeratorkern, c-4223, c-4271, c-5711  
\gmu@PDFdetector, c-4311, c-4321  
\gmu@pdfdetector, c-4308, c-4320  
\gmu@prevsec, c-3664, c-3666, c-3688, c-3695, c-3725  
\gmu@printslashes, c-3039, c-3041, c-3041, c-3043, c-3046  
\gmu@quiettrue, c-215  
\gmu@resa, a-4233, a-4239, a-4338, a-4344, c-5602, c-5603, c-5605  
\gmu@reserveda, c-392, c-393, c-458, c-459, c-985, c-987, c-993, c-995, c-1089, c-1092, c-1095, c-1635, c-1640, c-2193, c-2194, c-2195, c-2196, c-2338, c-2352, c-2426, c-2429, c-2433, c-2452, c-2463, c-2737, c-2742, c-3640, c-3689, c-3690, c-3693, c-3979, c-3981, c-3983, c-3984, c-3985, c-5299, c-5337  
\gmu@reservedb, c-2741, c-2742, c-5307, c-5317, c-5328, c-5339

**File Key:** a=gmdoc.sty, b=gmdocc.cls, c=gmutils.sty, d=gmiflink.sty, e=gmverb.sty, f=gmoldcomm.sty

\gmu@reservedc, c-5288,  
c-5320, c-5322,  
c-5331, c-5333  
\gmu@reservedd, c-5291,  
c-5303  
\gmu@reservede, c-5295,  
c-5305  
\gmu@restorespecials,  
c-2790  
\gmu@restoreUpUpUp,  
c-6724, c-6725  
\gmu@RPfor, c-5586,  
c-5600, c-5612, c-5623  
\gmu@scalar, c-5943,  
c-5947, c-5948, c-5954  
\gmu@scalematchX,  
c-5933, c-5945, c-5969  
\gmu@scapLetters,  
c-5905, c-5915, c-5920  
\gmu@scapSpaces,  
c-5918, c-5923, c-5927  
\gmu@scapss, c-5926, c-5931  
\gmu@scscale, c-5962,  
c-5968  
\gmu@septify, c-2793,  
c-4964, c-6164, e-1084  
\gmu@setbasefont,  
c-5345, c-5347  
\gmu@setheading,  
c-3739, c-3745, c-3746  
\gmu@setsetSMglobal,  
c-1075, c-1080, c-1158  
\gmu@setSMglobal,  
c-1082, c-1084, c-1102  
\gmu@SMdo@scope,  
c-1201, c-1203,  
c-1206, c-1207, c-1221  
\gmu@SMdo@setscope,  
c-1190, c-1205, c-1219  
\gmu@SMglobalfalse,  
c-1043, c-1057,  
c-1084, c-1093,  
c-1121, c-1152, c-1209  
\gmu@SMglobaltrue,  
c-1019, c-1082  
\gmu@smttempa, c-1047,  
c-1056, c-1146, c-1151  
\gmu@star@loop, c-7050,  
c-7052, c-7056  
\gmu@storeifnotyet,  
c-1135, c-3008,  
c-4561, c-4696,  
c-4701, c-4851,  
c-4874, c-4893,  
c-4916, c-4928, c-4999  
\gmu@storespecials,  
c-2787

\gmu@stripchar, c-4428,  
c-4431, c-4814  
\gmu@theendlinechar,  
c-6752, c-6759  
\gmu@theskewchar,  
c-4424, c-4462,  
c-4475, c-4487, c-4499  
\gmu@thickmuskip,  
c-6783, c-6789  
\gmu@thinmuskip,  
c-6781, c-6787  
\gmu@thou@fiver,  
c-6517, c-6520,  
c-6528, c-6528  
\gmu@thou@i, c-6522,  
c-6543, c-6551  
\gmu@thou@ii, c-6522,  
c-6543, c-6551  
\gmu@thou@o, c-6522,  
c-6543, c-6551  
\gmu@thou@put, c-6521,  
c-6525, c-6540  
\gmu@thou@putter,  
c-6524, c-6534,  
c-6541, c-6548, c-6556  
\gmu@thousep, c-6553, c-6560  
\gmu@tilde, c-5380, c-5396  
\gmu@twostring, c-2110,  
c-2118, c-2136  
\gmu@url@flexbreak,  
c-6908, c-6911  
\gmu@url@rigidbreak,  
c-6906  
\gmu@urlbreakable,  
c-6869, c-6911  
\gmu@whonly, c-5804, c-5805  
\gmu@xedeckfracplain,  
c-4199, c-4253  
\gmu@xedeckfracstar,  
c-4199, c-4214  
\gmu@xefracdef,  
c-4215, c-4229,  
c-4230, c-4231,  
c-4232, c-4233,  
c-4234, c-4235,  
c-4236, c-4237,  
c-4238, c-4239,  
c-4240, c-4241,  
c-4242, c-4243  
\gmv@dismath, e-869,  
e-872, e-876  
\gmv@disverb, e-872, e-875  
\gmv@edismath, e-870, e-878  
\gmv@exhyphenpe, e-583,  
e-587  
\gmv@hashhalfing,  
e-1064, e-1094

\gmv@hyphen, a-7550,  
c-3083, e-1113  
\gmv@hyphenchar,  
c-3089, c-6764,  
e-459, e-1112  
\gmv@hyphenpe, e-582, e-586  
\gmv@packname, e-534,  
e-535, e-539  
\gmv@storedhyphenchar,  
e-458, e-600, e-809  
\GMverbatimimspecials,  
b-565, e-1045  
\gn@melet, a-4593,  
a-4594, c-1238, e-768  
\gobble, a-6604, c-247,  
c-249, c-2330,  
c-4930, e-1066  
\gobblespace, c-1676,  
c-2223, c-2226,  
c-2227, c-2231  
\gobbletwo, c-250  
\grab@default, c-1700,  
c-1728, c-1739,  
c-1740, c-1745, c-1746  
\grab@defaults, c-1702,  
c-1736  
\grab@prefix, c-1669, c-1763  
\grefstepcounter,  
a-2813, c-530, c-546  
\grelaxen, a-6222,  
a-6231, c-614, c-3666

\hash, a-7530, c-3109  
\hathat, c-3106  
\hb@xt@, a-7053  
\HeadingNumber, c-3542,  
c-3544  
\HeadingNumbered!  
false, c-3468,  
c-3513  
\HeadingRHeadText, c-3526  
\HeadingText, c-3528  
\HeadingTOCText, c-3527  
\HeShe, c-3433  
\heshe, 7, c-3428  
\hfillneg, c-5433  
\hrefstepcounter,  
a-2844, a-2849, c-545  
\hidden@iffalse, c-508,  
c-2541  
\hidden@iftrue, c-509,  
c-2541  
\Hide@Dfng, a-4589, a-4591  
\Hide@DfngOnce, a-4589,  
a-4598  
\HideAllDefining, 14,  
a-4557

**File Key:** a=gmdoc.sty, b=gmdocc.cls, c=gmutils.sty, d=gmiflink.sty, e=gmverb.sty,  
f=gmoldcomm.sty

```

\HideDef, 14, a-4420
\HideDef*, 14
\HideDefining, 14,
 a-4422, a-4585
\HimHer, c-3435
\himher, c-3430
\HisHer, c-3434
\hisher, c-3429
\HisHers, c-3436
\hishers, c-3431
\HLPrefix, 20, a-3305,
 a-4672, a-4674,
 a-4756, a-5132,
 a-5139, a-5141,
 a-5146, a-5874, a-6690
\hrule, c-3713
\hsize, c-6382, c-6416,
 c-6664, c-6668,
 c-6669, c-6956
\hunskip, c-553, c-5532,
 c-5534, c-5536,
 c-6368, c-6403, c-6412
\Hybrid@DefEnvir,
 a-5507, a-5574
\Hybrid@DefMacro,
 a-5507, a-5570
\HyOrg@maketitle,
 c-6628, c-6629
hyperindex, 65
\hyperlabel@line,
 a-2829, a-2925,
 a-3104, a-3114, a-3299
\hypersetup, a-2239, a-5922
\hyphenpenalty, a-7459,
 b-537, c-3027, c-6103,
 c-6334, e-582, e-586

\idiaeres, b-440, b-455
\if@aftercode, a-2904,
 a-2998, a-3002,
 a-3188, a-3194,
 a-3237, a-3252,
 a-3353, a-3366,
 a-3539, a-7640,
 a-7645, a-7653
\if@afterindent, c-3670
\if@afternarr, a-2907,
 a-2998, a-3002,
 a-3188, a-3193,
 a-3358, a-3365
\if@codeskipput,
 a-2425, a-2433,
 a-2442, a-2889,
 a-2909, a-3199,
 a-3329, a-3364,
 a-5357, a-5368, a-5501
\if@countalllines,
 a-2131, a-2778
\if@dc@alllong@,
 c-1586, c-1671
\if@dc@ignore, c-1601,
 c-1679, c-1685, c-1692
\if@dc@quiet@, c-1587,
 c-2321
\if@debug, b-225, b-479,
 b-485, b-486
\if@dsdir, a-2459, a-5487
\if@filesw, a-5129,
 a-6764, a-6774,
 a-6810, c-5478,
 c-5770, c-5782,
 c-5790, c-5826,
 c-5837, c-5866
\if@gmcnnochanges,
 b-237, b-522
\if@gmu@mmhbox, c-4248,
 c-4258, c-4262,
 c-5716, c-6583
\if@ilgroup, a-2948,
 a-3189, a-3195,
 a-3238, a-3246,
 a-3253, a-3541
\if@indexallmacros,
 a-2161, a-5815
\if@linesnotnum,
 a-2117, a-3297, a-4737
\if@ltxDocInclude,
 a-6789, a-6795,
 a-6799, a-6989
\if@mainmatter, c-3468
\if@marginparsused,
 a-2173, a-5290
\if@newline, a-2451,
 a-2827, a-2921,
 a-3081, a-3100, a-3111
\if@nobreak, c-3667
\if@noindex, a-2147, a-2260
\if@noskipsec, e-659
\if@nostanza, a-2442,
 a-3332
\if@openright, c-3470
\if@pageinclindex,
 a-4671, a-4722, a-5138
\if@pageindex, a-2152,
 a-3300, a-4668,
 a-4739, a-5157,
 a-5867, a-5870,
 a-5871, a-5873
\if@printalllinenos,
 a-2132, a-2824, a-5333
\if@RecentChange,
 a-6164, a-6248
\if@specialpage, c-3531
\if@twoside, c-3557
\if@uresetlinecount,
 a-2124, a-3269
\if@variousauthors,
 a-7200, a-7213
\IfAmong, a-7686, a-7693,
 a-7699, a-7703,
 a-7707, a-7711,
 a-7716, a-7759, c-399,
 c-424, c-458, c-1699,
 c-1701, c-1861,
 c-1920, c-1945,
 c-1952, c-2195,
 c-2300, c-2306,
 c-2307, e-1092
\ifcsname, a-4700,
 a-4713, a-7602,
 c-629, c-2119, c-2137,
 c-2681, c-4587,
 c-5900, e-764
\ifdate, c-6269, c-6279
\IfDCMessages, c-1590,
 c-2274, c-2279,
 c-2323, 145
\ifdefined, a-4932,
 a-7549, a-7551,
 b-290, c-190, c-255,
 c-647, c-698, c-763,
 c-1582, c-3081,
 c-3083, c-3089,
 c-3288, c-4151,
 c-4695, c-4700,
 c-4940, c-5613,
 c-6093, c-6154,
 c-6400, c-6425,
 c-6628, c-6648,
 c-6743, c-6764,
 c-6774, e-1005
\ifdtraceoff, b-486
\ifdtraceon, b-485
\iffontchar, c-911,
 c-4077, c-4216,
 c-4567, c-4570,
 c-4573, c-4635,
 c-4645, c-4756,
 c-4757, c-4811,
 c-4865, c-4879,
 c-4882, c-4886,
 c-4887, c-4888,
 c-6196, c-6808, c-6824
\ifgmcc@mwccls, b-193,
 b-386, b-390, b-412
\ifgmcc@oldfonts,
 b-263, b-426, b-491
\ifgmd@adef@cshook,
 a-3863, a-4161

```

**File Key:** a=gmdoc.sty, b=gmocc.cls, c=gmutils.sty, d=gmiflink.sty, e=gmverb.sty,  
f=gmoldcomm.sty

\ifgmd@adef@star, a-3982, a-4033  
\ifgmd@glosscs, a-3856  
\ifgmu@postsec, c-3685, c-3724, c-3732  
\ifgmu@quiet, c-213, c-1122, c-1143  
\ifgmu@SMglobal, c-1017, c-1041, c-1048, c-1081, c-1119, c-1147, c-1206  
\ifHeadingNumbered, c-3512, c-3540  
\ofilrr, a-2999, a-3003, a-3190, a-3234, a-3539  
\IfIntersect, a-7757, a-7758, c-412, c-1764, c-1766, c-2290, c-2293, c-2294, c-2449, e-1086, e-1089  
\IfIntersect@next, c-422, c-425, c-426  
\IfLong, c-2186, 145  
\IfNoValueF, c-2174, c-2181  
\IfNoValueT, a-7762, c-2172, c-2184  
\IfNoValueTF, c-2162, c-2172, c-2174, c-2176  
\ifprevhmode, a-3028, a-3122, a-3220  
\ifSecondClass, c-5695  
\IfValueF, a-6600, c-2184, c-3767, c-6229, c-6252  
\IfValueT, a-2673, a-6596, a-7722, a-7724, a-7726, a-7732, a-7738, c-2178, c-2181, c-2428, c-2575, c-2791, c-2983, c-2987, c-3000, c-3248, c-3250, c-3251, c-3252, c-3253, c-4163, c-4442, c-4457, c-4465, c-4469, c-4502, c-4586, c-4660, c-4954, c-5156, c-6005, c-6010, c-6230, c-6237, c-6238, c-6253, c-6261, c-6262, c-6360, c-6391, c-6499, c-6617, c-6780, e-1030, e-1033  
\IfValueTF, a-5111, a-5121, a-5198, a-5208, a-5248, a-5259, a-5313, a-6459, a-7649, c-2176, c-2375, c-2575, c-2991, c-2993, c-4335, c-4454, c-4495, c-4524, c-4569, c-4572, c-4611, c-5504, c-6360, c-6392, c-6445, c-6475, c-6501, c-6577, c-6955, e-785  
\ignoreactiveM, c-967, c-974  
iI, 144  
\ikern, c-5736  
\ilju, 22, a-3250  
\ilrr, 22, a-3236  
ilrr, 22  
\ilrrfalse, a-3254  
\ilrrtrue, a-3243  
\im@firstpar, a-3884, a-3886, a-3888, a-4910, a-4911, a-4914  
\IMHO, c-5671  
\in, c-4752, c-5014  
\incl@DocInput, a-6798, a-7001, a-7024, a-7028, a-8210, a-8211, a-8227  
\incl@filedivtitle, a-7159, a-7188  
\incl@titletotoc, a-7146, a-7160  
\inlasthook, c-5787, c-5809, c-5863  
\InclMaketitle, a-6792, a-7137  
\includecountfix, c-5890  
\incmd, 22, a-7526  
\incs, 22, a-7509, a-7524  
\indent, c-6651  
\index@macro, a-3888, a-4651, a-4914, a-4991, a-5077  
\index@prologue, a-5858, a-5865, a-5917, a-6927  
indexallmacros, 11, a-2163  
IndexColumns, 21  
\indexcontrols, a-3793, a-3851  
\indexdiv, a-5861, a-5865, a-6406  
\indexentry, a-5131  
\IndexInput, 10, a-7335  
\IndexLinksBlack, 21, a-5878, a-5918, a-5922, a-6362  
\IndexMin, 21, a-5912, a-5912, a-5917  
\IndexParms, 21, a-5919, a-5926, a-6413  
\IndexPrefix, 20, a-4676, a-4728  
\IndexPrologue, 21, a-5858, 116  
\inenv, 22, a-7524  
\infty, c-4725  
\inhash, a-7530  
\inputlineno, a-2805, a-2806, a-2843  
\interlinepenalty, e-690  
\inverb, 22, a-7439  
\inverbpenalty, a-7459, a-7469  
\iota, c-4792  
\itemindent, c-3931, c-3949  
\itemize\*, c-3941  
\iteracro, c-5627, c-5638  
\justified, c-6343  
\kappa, c-4793  
\kernel@ifnextchar, a-7304  
\kind@fentry, a-4659, a-4661, a-4665, a-4672, a-4674  
\Kvfam, 13, a-4078  
\Kvpref, 13, a-4057  
\labelsep, c-3933, c-3951  
\labelwidth, c-3932, c-3933, c-3950, c-3951  
\Lambda, c-4769  
\lambda, c-4794  
\larger, c-2867, c-4856, c-4861, c-4903, c-4904, c-4907, c-4908, c-4909, c-4910, 165  
\largerr, c-2871, c-4905, c-4906, 165  
\last@defmark, a-4704, a-4845, a-4850, a-4851, a-6169, a-6181, a-6182, a-6183, a-6228, a-6231  
\LaTeXe, c-3965, c-4018  
\LaTeXpar, 23, c-4029  
\ldate, c-6303, c-6313

File Key: a=gmdoc.sty, b=gmdocc.cls, c=gmutils.sty, d=gmiflink.sty, e=gmverb.sty, f=gmoldcomm.sty

\le, c-4686, c-4687, c-4688,  
c-4698  
\leeng, c-4688  
\leftarrow, c-4741, c-5010  
\leftline, c-6445, c-6500  
\leftmargin, c-3930, c-3948  
\leftmargini, e-705  
\leftrightarrow,  
c-4743, c-4889  
\leftslanting, c-5123,  
c-5125  
\leftslanting@, c-5057,  
c-5123, c-5125  
\leq, c-4687, c-4696,  
c-4697, c-4698  
\levelchar, 21, a-3656,  
a-3852, a-6155,  
a-6204, a-6218  
\lim, c-4819  
\linebreak, c-6477  
\linedate, c-6275, c-6291,  
c-6306  
\linedate@, c-6275,  
c-6276, c-6277  
\linedate@@, c-6275, c-6276  
\linedate@hook, c-6278,  
c-6287  
\LineNumFont, 20,  
a-2830, a-3292,  
a-3295, a-7916,  
b-306, 114  
\lineskip, a-7087  
linesnotnum, 10, a-2119  
\list, c-3929, c-3947  
\listparindent, c-3934,  
c-3952  
\lit, c-5088  
\litcorrection, c-5059,  
c-5070, c-5077,  
c-5092, c-5097  
\litdimen, c-5058,  
c-5060, c-5065, c-5066  
\litkern, c-5061, c-5066  
\litshape, c-5068,  
c-5086, c-5088, c-5113  
\liturgiques, c-5585  
\LoadClass, b-394, b-399  
\longafterfi, c-353, c-355  
\longpauza, c-6131,  
c-6132, c-6139, c-6140  
\looseness, c-6023,  
c-6034, c-6360, c-6392  
\lozenge, c-4758  
\lpauza, c-6072  
\lsl, c-5091  
\lsu, b-326

\ltxLookSetup, 9,  
a-6991, a-6997  
\ltxPageLayout, 10,  
a-6630, a-6993  
\LuaTeX, c-4109  
luzniej, c-6027  
luzniej\*, c-6032  
\luzniejcore, c-6019,  
c-6027  
\macro, a-5499, a-5522, c-4144  
macro, 16, a-5499  
macro\*, a-5522  
\macro@iname, a-3733,  
a-3751, a-3754,  
a-3766, a-3888,  
a-4914, a-4920,  
a-4947, a-4991, a-5077  
\macro@pname, a-3735,  
a-3755, a-3767,  
a-3865, a-3867,  
a-3868, a-3877,  
a-3888, a-3890,  
a-3891, a-3894,  
a-4016, a-4348,  
a-4349, a-4351,  
a-4352, a-4373,  
a-4378, a-4383,  
a-4610, a-4903,  
a-4905, a-4909,  
a-4914, a-4971,  
a-4972, a-4975,  
a-4978, a-4991, f-38,  
f-39  
\macrocode, a-5389, f-67  
macrocode, 8, 25, a-5367  
macrocode\*, a-5356  
\MacrocodeTopsep, a-7896  
\MacroFont, a-7886, 113  
\MacroIndent, a-7894, 114  
\MacroTopsep, a-2363,  
a-2382, a-2424,  
a-5500, a-5509, 114  
\main, a-8073  
\MakeGlossaryCon|  
trols, 18, a-6143,  
a-6153  
\MakePercentComment,  
a-8105  
\MakePercentIgnore,  
a-6141, a-8104  
\MakePrivateLetters,  
15, 20, a-2589,  
a-3682, a-3971,  
a-4588, a-4632,  
a-4803, a-4864,  
a-4887, a-4957,

a-4994, a-5011,  
a-5093, a-5102,  
a-5237, a-5273,  
a-5410, a-5503,  
a-5628, a-5823,  
a-6142, a-7834, a-7851  
\MakePrivateOthers,  
a-3690, a-4805,  
a-4866, a-4889,  
a-4958, a-4995,  
a-5013, a-5105,  
a-5238, a-5275,  
a-5503, a-7842, a-7852  
\MakeShortVerb, 12,  
e-484, e-859, e-916,  
e-917, 237  
\makestarlow, b-567,  
c-7022, e-1050  
\maketitle, 9, a-2007,  
a-2012, a-5795,  
a-6792, a-7048,  
a-8215, c-6627, 100  
\MakeUppercase, c-5909  
\mand, 24, a-7659  
\mapsto, c-5003  
\marg, c-3192, c-3229, c-3252  
\marginparpush, a-5292  
\marginpartt, 16, a-5312,  
a-5322, b-340, b-495,  
b-497  
\marginparwidth,  
a-5293, a-6644  
\mark@envir, a-2927,  
a-3109, a-5043  
maszynopis, c-6325  
\math@arg, c-3282, c-3283  
\mathalpha, c-4766,  
c-4767, c-4768,  
c-4769, c-4770,  
c-4772, c-4773,  
c-4774, c-4775,  
c-4776, c-4782,  
c-4783, c-4784,  
c-4785, c-4786,  
c-4787, c-4788,  
c-4789, c-4790,  
c-4791, c-4792,  
c-4793, c-4794,  
c-4795, c-4796,  
c-4797, c-4798,  
c-4799, c-4800,  
c-4801, c-4802,  
c-4803, c-4804,  
c-4805, c-4806,  
c-4807, c-4808  
\mathbin, c-4672, c-4673,  
c-4720, c-4722,

**File Key:** a=gmdoc.sty, b=gmdocc.cls, c=gmutils.sty, d=gmiflink.sty, e=gmverb.sty,  
f=gmoldcomm.sty

c-4723, c-4729,  
 c-4739, c-4762,  
 c-4763, c-4764,  
 c-4880, c-4883,  
 c-4925, c-4926,  
 c-5005, c-5014  
 $\backslash$ mathchar@type, c-4526,  
 c-4542, c-4648,  
 c-4658, c-4949  
 $\backslash$ mathchoice, c-4853,  
 c-4876, c-4917, c-5001  
 $\backslash$ mathclose, c-4713,  
 c-4717, c-4737,  
 c-4904, c-4906,  
 c-4908, c-4910, c-4923  
 $\backslash$ mathfrak, c-5623  
 $\backslash$ mathindent, b-418  
 $\backslash$ mathit, c-4466  
 $\backslash$ mathop, c-4653, c-4731,  
 c-4853  
 $\backslash$ mathopen, c-4710, c-4715,  
 c-4736, c-4903,  
 c-4905, c-4907,  
 c-4909, c-4922  
 $\backslash$ mathord, c-4675, c-4676,  
 c-4677, c-4678,  
 c-4679, c-4680,  
 c-4681, c-4682,  
 c-4683, c-4684,  
 c-4724, c-4725,  
 c-4730, c-4738,  
 c-4758, c-4759  
 $\backslash$ mathpunct, c-4706,  
 c-4707, c-4708, c-4709  
 $\backslash$ mathrel, c-4674, c-4686,  
 c-4689, c-4692,  
 c-4693, c-4721,  
 c-4726, c-4727,  
 c-4732, c-4734,  
 c-4740, c-4741,  
 c-4742, c-4743,  
 c-4746, c-4750,  
 c-4752, c-4889,  
 c-4931, c-5004,  
 c-5010, c-5011,  
 c-5012, c-5013, c-5017  
 $\backslash$ mathrm, c-4503, c-4653,  
 c-4864  
 $\backslash$ mathversion, c-4660  
 $\backslash$ max, c-4817  
 $\backslash$ maybe@marginpar,  
 a-3890, a-3912  
 $\backslash$ mcdiagOff, a-7622  
 $\backslash$ mcdiagOn, a-7618  
 $\backslash$ mch, b-557  
 $\backslash$ medmuskip, c-2969,  
 c-6782, c-6788, c-6797  
 $\backslash$ meta, c-2894, c-2945,  
 c-3192, c-3200,  
 c-3207, c-3211, 115  
 $\backslash$ meta@font@select,  
 c-2907, c-2933  
 $\backslash$ meta@fontsetting,  
 c-2902, c-2918,  
 c-2931, c-2933, c-2948  
 $\backslash$ metachar, a-7729, b-557,  
 c-2947, c-3249,  
 e-1035, e-1048  
 $\backslash$ metacharfont, c-2947,  
 c-2948  
 $\backslash$ min, c-4818  
minion, b-314, 121  
 $\backslash$ mkern, c-4934  
 $\backslash$ mod@math@codes,  
 a-6084, a-6086, a-6088  
 $\backslash$ Module, a-6042, a-6084  
 $\backslash$ ModuleVerb, a-6077, a-6086  
 $\backslash$ month, a-1995, a-6573, a-6577  
mptt, b-274  
 $\backslash$ mskip, c-2969  
 $\backslash$ mu, c-4795  
 $\backslash$ multiply, a-6255,  
 a-6260, c-4008,  
 c-4011, c-5960,  
 c-6022, c-6033,  
 c-6361, c-6393  
 $\backslash$ mw@getflags, c-3688  
 $\backslash$ mw@HeadingBreakAfter,  
 c-3533, c-3553,  
 c-3568, c-3572,  
 c-3603, c-3689  
 $\backslash$ mw@HeadingBreakBefore,  
 c-3530, c-3602, c-3690  
 $\backslash$ mw@HeadingLevel,  
 c-3510, c-3513  
 $\backslash$ mw@HeadingRunIn,  
 c-3548, c-3602  
 $\backslash$ mw@HeadingType,  
 c-3529, c-3664,  
 c-3696, c-3697, c-3710  
 $\backslash$ mw@HeadingWholeWidth,  
 c-3551, c-3603  
 $\backslash$ mw@normalheading,  
 c-3555, c-3564,  
 c-3567, c-3571, c-3745  
 $\backslash$ mw@processflags, c-3604  
 $\backslash$ mw@runinheading,  
 c-3549, c-3746  
 $\backslash$ mw@secdef, c-3609,  
 c-3610, c-3611, c-3617  
 $\backslash$ mw@section, c-3608  
 $\backslash$ mw@sectionxx, c-3509  
 $\backslash$ mw@secundef, c-3613,  
 c-3625, c-3628  
 $\backslash$ mw@setflags, c-3614  
mwart, b-203, 121  
mwbk, b-210, 121  
mwrep, a-1984, b-206, 121  
myriad, b-323  
 $\backslash$ n@melet, a-4017, a-4018,  
 a-5392, a-5393,  
 a-6183, c-1230,  
 c-1245, c-1246,  
 c-2214, c-2226,  
 c-3639, c-3641,  
 c-3899, c-4219, e-611  
 $\backslash$ nacute, b-441, b-456  
 $\backslash$ nameshow, c-2593  
 $\backslash$ nameshowthe, c-2594  
 $\backslash$ napapierkicore,  
 c-5995, c-6007  
 $\backslash$ napapierkistretch,  
 c-5993, c-5996  
 $\backslash$ narrationmark, 20,  
 a-2316, a-3019,  
 a-3122, a-7464,  
 a-7478, a-7480,  
 a-7484, a-7492,  
 a-7550, a-7653  
 $\backslash$ narrativett, a-3621,  
 a-3624, a-3626,  
 a-5322, a-6819,  
 a-6823, a-6942,  
 a-6945, a-7476,  
 a-7669, b-344, b-345,  
 c-3087, c-3098,  
 c-3192, c-3200,  
 c-3207, c-3211  
 $\backslash$ narra!  
 tivett@storedhyphenchar,  
 c-3088, c-3095  
 $\backslash$ nawj, c-6036  
 $\backslash$ nazwired, c-6208  
 $\backslash$ ne, c-4728  
 $\backslash$ neb, c-4927  
 $\backslash$ neg, c-4729, c-4933  
 $\backslash$ nego, c-4730  
 $\backslash$ neq, c-4727, c-4728, c-4926  
 $\backslash$ neqb, c-4926, c-4927  
NeuroOncer, a-5589  
 $\backslash$ newbox, a-4436  
 $\backslash$ newcount, a-4431,  
 a-5601, a-5915,  
 a-6245, a-6352,  
 a-6485, c-1598,  
 c-1599, c-6017  
 $\backslash$ newcounter, a-3273,  
 a-3276, a-3277,  
 a-4463, a-6489,

**File Key:** a=gmdoc.sty, b=gmocc.cls, c=gmutils.sty, d=gmiflink.sty, e=gmverb.sty,  
f=gmoldcomm.sty

a-7597, a-8083,  
 c-3456, c-5900, d-150  
`\newdimen`, a-4432,  
 a-5912, a-6350  
`\newgeometry`, b-474  
`\newgif`, a-2451, a-2459,  
 a-3028, a-3329,  
 a-3332, a-3353,  
 a-3358, c-472  
`\newlength`, a-2340,  
 a-2344, a-2350,  
 a-2972, a-2975,  
 a-4439, e-699  
`\newline`, a-7492  
`\newlinechar`, a-7280,  
 a-7292  
`\newread`, a-4437  
`\newskip`, a-2362, a-2363,  
 a-3548, a-4433,  
 c-6426, c-6491, e-703,  
 e-707  
`\newtoks`, a-2327, a-4435,  
 c-1583, c-1596  
`\newwrite`, a-4438, c-5478  
`\next@dc`, c-1669, c-1700,  
 c-1702, c-1705,  
 c-1707, c-1708,  
 c-1712, c-1716, c-1732  
`\NextBgroup`, c-5856  
`\nfss@text`, c-2903  
`\nieczer`, c-5594  
`\nlperc`, 22, a-7492  
`\nlpercent`, 22, a-7494,  
 a-7528, a-7530  
`\nobreakbslash`, e-398, 236  
`\nobreakbrace`, e-405, 236  
`\nobreakspace`, c-5926,  
 c-6955, c-6965  
`\nochanges`, b-239, 120  
`\nocite`, c-3372  
`\noeffect@info`, a-7922,  
 a-7940, a-7941,  
 a-7942, a-7943,  
 a-8088, a-8093,  
 a-8094, a-8098  
`\NoEOF`, a-8311  
`\nofileparts`, c-5875  
`\nohy`, c-5740  
`\noindex`, 10, a-2149, b-232,  
 120  
`\nolimits`, c-4868, c-4872  
`\nolinkurl`, c-6603  
`\nomarginpar`, 11, a-2190  
`NoNumSecs`, c-3456  
`\NonUniformSkips`, 19,  
 a-2412  
`\nostanza`, 19, a-2439  
`\not@onlypreamble`,  
 c-3328, c-3332,  
 c-3333, c-3334,  
 c-3335, c-3336, c-4830  
`\NoValue`, c-1740, c-1778,  
 c-1786, c-1821,  
 c-1828, c-1888,  
 c-1940, c-1976,  
 c-2156, c-2157,  
 c-2160, c-2168,  
 c-2206, c-2224,  
 c-4437, c-4514,  
 c-4559, c-4656,  
 c-4951, e-976, e-987  
`\noverbatimspecials`,  
 a-3893, a-8216,  
 e-1042, 238  
`\nu`, c-4796  
`\numexpr`, a-2806, a-7213,  
 c-1993, c-2001,  
 c-2007, c-2023,  
 c-2025, c-2038,  
 c-3115, c-3144,  
 c-3312, c-5262,  
 c-5266, c-5267,  
 c-5368, c-5369,  
 c-6187, c-6188,  
 c-6200, c-6539,  
 c-7054, c-7057, e-1112  
`\oarg`, c-3198  
`\obeyspaces`, a-2733,  
 a-2747, a-5442,  
 c-772, e-427, e-429,  
 e-431, e-827  
`\ocircum`, b-442, b-459  
`\OCRInclude`, a-8209  
`\oddsidemargin`, a-6645  
`\oe`, b-463  
`\old@MakeShortVerb`,  
 a-8146, e-890, e-912  
`\oldcomments`, f-28  
`\olddekclubs`, e-860, 237  
`\olddocIncludes`, 9, 25,  
 a-7023  
`\OldDocInput`, 8, 25,  
 a-7024, a-8143  
`\oldLaTeX`, c-3964  
`\oldLaTeXe`, c-3965  
`\OldMacrocodes`, 25, a-8148  
`\OldMakeShortVerb`,  
 e-860, e-911, 237  
`\oldmc`, a-5389, a-5397  
`\oldmc`, 25, a-5389  
`\oldmc*`, a-5392  
`\oldmc@def`, a-5447, a-5453  
`\oldmc@end`, a-5448, a-5454  
`\Omega`, c-4776  
`\omega`, c-4808  
`\OnlyDescription`, 22,  
 a-8002  
`\opt`, 24, a-7664  
`\oumlaut`, b-443, b-457  
`\outeroff`, a-1984, b-221, 121  
`\PackageE`, c-6986  
`\PackageError`, a-4250,  
 a-4941, a-6721,  
 a-6727, a-6742,  
 a-6907, c-3347,  
 c-6084, c-6115  
`\PackageInfo`, a-7914,  
 a-7922, e-539  
`\PackageWarning`,  
 c-1123, c-1144,  
 c-2275, c-2858,  
 c-2862, c-6718  
`\PackageWarningNoLine`,  
 a-6145  
`\pagebreak`, c-3556,  
 c-3568, c-3572  
`\pagegoal`, c-5503, c-5506  
`\PageIndex`, a-7964, a-7966  
`\pageindex`, 11, a-2154  
`\pagella`, b-315, 121  
`\pagestyle`, b-516, c-6995  
`\pagetotal`, c-5503, c-5506  
`\par`, a-2423, a-2431,  
 a-2441, a-2532,  
 a-2581, a-2888,  
 a-3001, a-3151,  
 a-3172, a-3185,  
 a-3210, a-3261,  
 a-3540, a-5357,  
 a-5359, a-5368,  
 a-5370, a-5502,  
 a-5509, a-5723,  
 a-5939, a-5942,  
 a-6603, a-7048,  
 a-7085, a-7090,  
 a-7094, a-7397,  
 a-7412, a-7416  
`\paragraph`, a-6957, c-6309  
`\paragraphmark`, c-3855  
`\ParanoidPostsec`,  
 b-412, c-3721  
`\parg`, c-3205  
`\parsep`, e-657  
`\partial`, c-4738  
`\partmark`, c-3855  
`\partopsep`, a-2397,  
 c-3930, c-3948, e-662

\PassOptionsToPack |  
age, b-233, b-353,  
b-363, b-367, c-4163  
\pauza, c-6055  
\pauza@skipcore,  
c-6045, c-6100,  
c-6101, c-6102  
\pauzacore, c-6046,  
c-6059, c-6068,  
c-6074, c-6103,  
c-6109, c-6131,  
c-6134, c-6139,  
c-6142, c-6337, c-6338  
\pauzadial, c-6061, c-6067  
\pdef, a-2316, a-2430,  
a-2553, a-2664,  
a-2678, a-2683,  
a-3626, a-4398,  
a-4416, a-5270,  
a-6431, a-6453,  
a-6582, a-7335,  
a-7439, a-7451,  
a-7471, a-7492,  
a-7494, a-7509,  
a-7558, a-7564,  
a-7566, c-240, c-254,  
c-304, c-386, c-399,  
c-442, c-472, c-489,  
c-508, c-509, c-530,  
c-545, c-553, c-794,  
c-836, c-887, c-910,  
c-935, c-951, c-960,  
c-967, c-972, c-1009,  
c-1019, c-1030,  
c-1073, c-1110, c-1135,  
c-1155, c-1177, c-1181,  
c-1244, c-1605,  
c-1755, c-2186,  
c-2514, c-2722,  
c-2793, c-2831,  
c-2871, c-2872,  
c-2894, c-2921,  
c-2947, c-2961,  
c-2964, c-3017,  
c-3032, c-3039,  
c-3053, c-3093,  
c-3101, c-3106,  
c-3192, c-3198,  
c-3200, c-3201,  
c-3205, c-3210,  
c-3283, c-3460,  
c-3967, c-3987,  
c-4179, c-4191,  
c-4244, c-4254,  
c-4265, c-4294,  
c-4325, c-4652,  
c-4844, c-4959,  
c-4963, c-5035,  
c-5057, c-5068,  
c-5085, c-5088,  
c-5091, c-5110, c-5119,  
c-5123, c-5248,  
c-5345, c-5389,  
c-5532, c-5534,  
c-5536, c-5628,  
c-5665, c-5674,  
c-5740, c-5741,  
c-5899, c-5931,  
c-6055, c-6067,  
c-6072, c-6083,  
c-6096, c-6114,  
c-6193, c-6222,  
c-6245, c-6271,  
c-6275, c-6276,  
c-6277, c-6291,  
c-6303, c-6316,  
c-6319, c-6343,  
c-6367, c-6513,  
c-6568, c-6603,  
c-6621, c-6633,  
c-6643, c-6647,  
c-6661, c-6666,  
c-6681, c-6708,  
c-6713, c-6718,  
c-6906, c-6908,  
c-6975, c-6986,  
c-7011, e-398, e-405,  
e-445, e-946, e-964,  
e-980, e-992, e-1042  
\pdfTeX, 23, c-4083  
\pdfLaTeX, c-4086  
\pdfTeX, 23, c-4085  
\Phi, c-4774  
\phi, c-4805  
\Pi, c-4799  
\pi, c-4798  
\pk, 23, a-1992, a-1993,  
a-2023, a-6696, c-3053  
\PlainTeX, 23, c-4068  
\pm, c-4739  
\polskadata, c-6222, c-6242  
\possfil, c-3112  
\possvfil, c-3114, c-6280,  
c-6284  
Pp!1L\long\par, 144  
\ppauza, c-6083  
\ppauza@dash, c-6051,  
c-6089, c-6123,  
c-6135, c-6143  
\ppauza@skipcore,  
c-6050, c-6120, c-6121  
\pprovide, a-4400, c-269,  
c-5622, c-6198  
\prec, c-5015  
\predisplaypenalty,  
e-584, e-594  
\prefix, a-4042  
\prependtomacro,  
a-7750, c-589, c-5157,  
c-6620  
\prevdepth, c-6977  
\prevhmodegfalse,  
a-2895, a-2944,  
a-3038, a-3176, a-3205  
\prevhmodegtrue, a-3037  
\PrintChanges, 17,  
a-2033, a-6425,  
a-6429, a-6866  
\PrintDescribeEnv, 114  
\PrintDescribeMacro, 114  
\PrintEnvName, 114  
\PrintFilesAuthors, 9,  
a-7204  
\PrintIndex, a-2037,  
a-5951, a-6866  
\printindex, a-5953,  
a-5954, a-6866  
\printlinenumber,  
a-2923, a-3107,  
a-3291, a-3297  
\PrintMacroName, 114  
\printsaces, c-3023,  
c-3032  
\ProcessOptionsX, b-369  
\protected, a-3432,  
a-3559, a-3562,  
c-240, c-269, c-484,  
c-503, c-1630, c-2244,  
c-2319, c-2394,  
c-2511, c-4647,  
c-5031, c-5033,  
c-5234, c-6982,  
e-1013, e-1015,  
e-1017, e-1032, e-1036  
\provide, a-4399, a-7660,  
b-548, c-254, c-269  
\providecolor, e-936  
\providecounter,  
c-5893, c-5899  
\ProvideFileInfo, 24,  
a-7300, a-7316  
\ProvidesClass, b-50  
\ProvideSelfInfo, a-7316  
\ps@plain, a-7075  
\ps@titlepage, a-7075  
\Psi, c-4775  
\psi, c-4807  
\PutIfValue, c-2178,  
c-4456, c-4459,  
c-4461, c-4463,  
c-4472, c-4473,

**File Key:** a=gmdoc.sty, b=gmdocc.cls, c=gmutils.sty, d=gmiflink.sty, e=gmverb.sty,  
f=gmoldcomm.sty

c-4474, c-4476,  
 c-4479, c-4484,  
 c-4485, c-4486,  
 c-4488, c-4497,  
 c-4498, c-4500,  
 c-4507, c-4662,  
 c-6229, c-6238,  
 c-6252, c-6262, e-1039  
  
 \qemph, 7, a-2678  
 \qemph@, a-2680, a-2683  
 \qfootnote, 7, a-2664  
 \qfootnote@, a-2666, a-2669  
 \quad, b-514, b-515, c-6208  
 \quantifierhook,  
     c-4968, c-4992  
 \QueerCharOne, a-3465,  
     a-3472, a-3474  
 \QueerCharTwo, a-3431,  
     a-3438, a-3440  
 \QueerEOL, 8, a-2501,  
     a-2631, a-3503,  
     a-5358, a-5369,  
     a-5953, a-6427,  
     a-7029, a-8311, a-8312  
 quiet, c-215  
 quotation, 24, a-7404  
 \quote@char, a-3732,  
     a-3753, a-3765,  
     a-3789, a-4946  
 \quote@charbychar,  
     a-4921, a-4923,  
     a-4937, a-4948  
 \quote@mname, a-4905,  
     a-4919, a-4982, a-5071  
 \quotechar, 21, a-3654,  
     a-3794, a-3852,  
     a-4677, a-4724,  
     a-4987, a-5074,  
     a-6154, a-6213  
 \quoted@eschar, a-4677,  
     a-4724, a-4987,  
     a-4988, a-5074, a-5075  
  
 \raggedbottom, b-504  
 \rdate, c-6291  
 \real, c-5954, c-5956  
 \RecordChanges, 17,  
     a-6147, a-6337,  
     a-6338, a-6866,  
     b-522, b-525  
 \reflectbox, c-4093, c-4100  
 \relaxen, a-4422, a-4595,  
     a-7747, a-8217,  
     a-8218, b-446, c-609,  
     c-2305, c-3015,  
     c-3599, c-6167,  
  
 c-6522, c-6710,  
 c-6715, e-376, e-873,  
 e-1051  
 \relpenalty, c-6798  
 \relsize, c-2831, c-2832,  
     c-2867, c-2868,  
     c-2869, c-2870,  
     c-2871, c-2872, 165  
 \rem@special, e-520,  
     e-545, e-560  
 \renewcommand, a-3343,  
     a-3345, a-4446,  
     a-6150, b-509, c-5612  
 \RequirePackage,  
     a-2109, a-2111,  
     a-2221, a-2224,  
     a-2241, a-2253,  
     a-2256, a-2264,  
     a-5903, b-178, b-408,  
     b-429, b-432, b-468,  
     b-481, b-489, b-542,  
     c-4165, c-5475,  
     c-5605, c-5938, e-331,  
     e-932  
 \resetlinecountwith,  
     a-3272  
 \resizebox, c-4299, c-5193  
 \resizegraphics, c-4294  
 \Restore@Macro, c-1113,  
     c-1116, c-1159, c-1169  
 \Restore@Macros,  
     c-1155, c-1157  
 \Restore@MacroSt,  
     c-1114, c-1141  
 \RestoreCatM, c-6982,  
     c-6986  
 \RestoreEnvironment,  
     c-1181  
 \RestoreMacro, a-3893,  
     a-4424, a-4570,  
     a-4571, a-4613,  
     a-4637, a-4638,  
     a-5846, a-7343,  
     a-8219, a-8227,  
     c-1110, c-1183, c-1757,  
     c-3865, c-3969,  
     c-4172, c-4174,  
     c-4177, c-4598,  
     c-4658, c-5855, f-64  
 \RestoreMacros, a-5178,  
     c-1155  
 \restoreparindent, c-6429  
 \RestoringDo, a-6805, c-1218  
 \ResumeAllDefining,  
     14, a-4568  
 \ResumeDef, 14, a-4424  
  
 \ResumeDefining, 14,  
     a-4424, a-4631  
 \reversemarginpar, a-5291  
 \rho, c-4800  
 \rightarrow, c-4742, c-5012  
 \rightline, b-543,  
     c-6291, c-6453  
 \rmopname, c-4652,  
     c-4817, c-4818,  
     c-4819, c-4820,  
     c-4821, c-4822,  
     c-4823, c-4824, c-4825  
 \romannumeral, c-2007,  
     c-2038, c-3928,  
     c-3946, c-5137  
 \romorzero, c-5136,  
     c-5174, c-5179  
 \rotatebox, c-4866,  
     c-4872, c-4880,  
     c-4883, c-5016  
 \rrthis, c-6661  
 \rs@size@warning,  
     c-2850, c-2855, c-2858  
 \rs@unknown@warning,  
     c-2845, c-2862  
 \runindate, c-6308  
  
 \scalebox, e-1135, e-1136  
 \scan@macro, a-3059,  
     a-3709, f-87  
 \scan@macro@, a-3710,  
     a-3715  
 \scantnoline, c-6681  
 \scantokens, a-7292,  
     c-306, c-5164, c-5172,  
     c-6682, c-6758, e-625,  
     e-1022, e-1097  
 \scantwo, c-304  
 \scanverb, a-3894,  
     a-5249, a-5250,  
     a-5260, a-5261,  
     a-5313, a-7729,  
     a-7736, a-7742,  
     c-3249, e-1072, 238  
 \scshape, c-4066, c-5005  
 \secondclass, c-5694  
 \SecondClasstrue, c-5696  
 \sectionmark, c-3855  
 \SelfInclude, 9, a-2015,  
     a-6976  
 semiverbatim, 238  
 \setbasefont, c-5347  
 \SetFileDiv, 23, a-6896,  
     a-6899, a-6901,  
     a-6909, a-6970, a-6992  
 \setkeys, a-3993, a-4000,  
     a-4027

\setmainfont, b-293  
\SetMathAlphabet,  
c-4466, c-4503  
\setmonofont, b-337  
\setprevdepth, c-6457,  
c-6505, c-6977  
\setsansfont, b-319,  
b-322, b-325, b-328  
\SetSectionFormat!  
ting, c-3599,  
c-3600, c-3764,  
c-3769, c-3778,  
c-3786, c-3793,  
c-3800, c-3805  
\setspacekip, c-6998,  
e-455  
\SetSymbolFont, c-4454,  
c-4495  
\settexcodehangi,  
a-2329, a-2334,  
a-2931, a-2939, a-3221  
\SetTOCIndents, b-514,  
b-515  
\SetTwoheadSkip,  
c-3748, c-3777,  
c-3785, c-3792  
\sfname, c-3032, c-3042  
\sgtleftxii, a-6027, a-6071  
\shortleftarrow, c-5011  
\shortpauza, c-6133, c-6141  
\shortrightarrow, c-5013  
\shortthousep, c-6572  
\showboxbreadth, c-2588  
\showboxdepth, c-2588  
>ShowFont, c-5570  
\showlists, c-2588  
\showthe, c-2594  
\Sigma, c-4772  
\sigma, c-4801  
\sim, c-4740  
\sin, c-4820  
\skewchar, c-4424,  
c-4462, c-4475,  
c-4487, c-4499  
\SkipFilesAuthors, 9,  
a-7206  
\skipgmlonely, 23,  
a-2021, a-7362  
\skiplines, 26, a-2855  
\SliTeX, 23, c-4065  
\smaller, c-2868, c-5665,  
c-5719, c-6622, 165  
\smallerr, a-7163,  
c-2872, c-5969, 165  
\smallskipamount,  
a-2391, a-2392,  
c-5426, c-5427, c-5428  
\smartunder, *b*-534, c-724  
\SMglobal, a-4021,  
a-4559, a-4570,  
a-4571, a-4605,  
a-4613, a-4637,  
a-4638, c-1019  
\SortIndex, a-8037  
\spaceskip, c-7005  
\special, a-3069, a-3070  
\special@index, a-4682,  
a-5158, a-5162, a-5335  
\SpecialEnvIndex, a-8035  
\SpecialEscapechar,  
a-7900  
\SpecialIndex, a-8031  
\SpecialMainEnvIn!  
dex,  
a-8026  
\SpecialMainIndex, a-8023  
\SpecialUsageIndex,  
a-8033  
\sphack, c-2244, c-2294,  
c-2394, c-2449  
\spifletter, c-833, c-6584  
\sqrtsign, c-4813  
\square, b-543, c-4759  
StandardModuleDepth,  
a-8083  
\stanza, 19, 24, a-2021,  
a-2023, a-2430, a-7398  
\stanzaskip, 19, a-2350,  
a-2355, a-2381,  
a-2382, a-2383,  
a-2388, a-2389,  
a-2396, a-2432,  
a-2890, e-688, e-699,  
e-700  
star, 13, a-4033  
\step@checksum, a-3722,  
a-6492  
\StopEventually, 22,  
a-7982, a-8002  
\Store@Macro, c-1036,  
c-1039, c-1076  
\Store@Macros, c-1073,  
c-1074  
\Store@MacroSt, c-1037,  
c-1046  
\StoreCatM, c-6981, c-6987  
\stored@code@delim,  
a-5406  
\Stored@Macro, c-1168,  
c-1169  
\storedcsname, a-7413,  
a-7417, c-1172,  
c-3864, c-4470,  
c-4482, c-5592, c-6603  
\StoredMacro, c-1168  
\StoreEnvironment,  
a-7402, c-1177  
\StoreMacro, a-4021,  
a-4418, a-4559,  
a-4605, a-5840,  
a-7338, a-8210,  
c-1030, c-1137,  
c-1179, c-1753,  
c-3968, c-4022,  
c-4023, c-4081,  
c-4164, c-4531,  
c-4949, c-5591,  
c-5873, c-6337,  
c-6602, f-36  
\StoreMacros, a-2258,  
a-5176, c-1073, c-3855  
\StoringAndRelax!  
ingDo, a-6788,  
c-1198  
\StraightEOL, 7, a-2631,  
a-2666, a-2680,  
a-3490, a-5953,  
a-6142, a-6427,  
a-7360, a-7373,  
a-7396, a-7639, a-8145  
\strip@pt, c-5058,  
c-5073, c-5079,  
c-5099, c-5277, c-5365  
\subdivision, 23,  
a-6956, a-7588  
\subitem, a-5940  
\subs, c-671, c-726, 238  
\subsectionmark, c-3855  
\subsubdivision, 23,  
a-6957, a-7591  
\subsubitem, a-5941  
\subsubsectionmark,  
c-3855  
\sum, c-4865  
\sup, c-675, 238  
\symgmathroman, c-4814  
\sysfonts, b-265, 121  
\tableofcontents,  
a-2011, a-2629,  
a-2630, a-6865  
\tan, c-4824  
\task, f-90  
\tau, c-4803  
\TB, 23, c-4074  
\TeXbook, 23, c-4073, c-4074  
\textcode@hook, a-2599,  
a-2609, a-8223, b-567  
\Text@CommonIndex,  
a-4995, a-4998

\Text@CommonIndexStar,  
a-4995, a-5002  
\text@indexenvir,  
a-4967, a-4969,  
a-5003, a-5264, a-7870  
\text@indexmacro,  
a-4902, a-4963,  
a-4999, a-5252, a-7861  
\Text@Marginize,  
a-3915, a-4352,  
a-5046, a-5249,  
a-5250, a-5260,  
a-5261, a-5279,  
a-5285, a-5309,  
a-5580, a-7859, a-7867  
\Text@MarginizeNext,  
a-5572, a-5577, a-5579  
\Text@UsgEnvir, a-5240,  
a-5255  
\Text@UsgIndex, a-4958,  
a-4961  
\Text@UsgIndexStar,  
a-4958, a-4966  
\Text@UsgMacro, a-5240,  
a-5243  
\textbullet, c-6193, c-6203  
\textcolor, c-5594, e-939  
\TextCommonIndex, 16,  
a-4993  
\TextIndent, 19, a-2340,  
a-3225, a-3377  
\textlarger, c-2869  
\textlit, c-5085  
\TextMarginize, 15, a-5270  
\textsl, c-4073  
\textsmaller, c-2870  
\textstyle, c-4020  
\textsuperscript,  
c-4330, c-4334,  
c-4343, c-6622  
\textsuperscript@@,  
c-4325, c-4330,  
c-4335, c-4343  
\texttilde, c-5389  
\texttt, a-3626  
\TextUsage, 14, a-5233  
\TextUsgIndex, 16,  
a-4956, a-8033  
\textvisibleSpace, c-766  
\textwidth, a-3248,  
a-6634, a-6928,  
c-6448, c-6467, c-6475  
\tg, c-4822  
\thanks, a-7084, a-7101,  
a-7144, a-7151, a-7322  
\theCodelineNo, a-7915, 114  
\thecodelinenum,  
a-2830, a-3292, a-7917  
\thedate, c-6313  
\thefilediv, a-6823,  
a-6921, a-6923,  
a-6925, a-6942,  
a-6945, a-7127  
\theglossary, a-6867  
\theglossary, a-6358  
\theindex, a-5916  
\thesection, b-509  
\Theta, c-4768  
\theta, c-4790  
\thfileinfo, 24, a-7322  
\thickmuskip, c-4935,  
c-6783, c-6789, c-6797  
\thinmuskip, c-6781,  
c-6787, c-6797  
\thous, c-6573  
\thousep, c-6513, c-6568,  
c-6588  
\thr@®, c-3924, c-3942  
\time, c-6187, c-6188  
\tinycae, c-5936  
\title, a-1992, a-7098, a-8215  
\titlesetup, a-7083,  
a-7110, b-519  
\toCTAN, 19, a-6431  
\TODO, c-5438  
\toks, c-3735, c-3736,  
c-3742, c-3743  
\tolerance, a-2493,  
c-6022, c-6033,  
c-6329, c-6361, c-6393  
\traceoff, b-486  
\traceon, b-485  
\trebuchet, b-320, 121  
\trimmed@everypar,  
a-3408, a-3410  
\true{textsuper!}  
script, c-4340,  
c-4342  
\TrzaskaTilde, c-6952  
\ttverbatim, a-2583,  
a-3620, a-3621,  
a-3623, a-3624,  
a-5408, a-5896,  
a-5897, a-7475,  
e-445, e-695, e-732  
\ttverbatim@hook,  
e-460, e-468, e-471,  
e-1133  
\twocoltoc, a-1991,  
c-5474, c-5483  
\twopar, c-6384  
\twopar@atleast,  
c-6383, c-6388, c-6402  
\twopar@default, c-6411  
\twopar@defts, c-6381,  
c-6404  
\twoparcore, c-6395,  
c-6401, c-6410  
\twoparinit, c-6379  
\type, 13, a-4090  
\type@bslash, a-7480,  
c-3067, e-389, e-398,  
e-798  
\type@lbrace, c-3086,  
c-3192, e-363, e-405  
\tytul, c-6319  
\tytulowa, c-6205  
\udigits, c-4191, c-4194  
\un@defentryze, a-4666,  
a-4699  
\un@usgentryze, a-4662,  
a-4712  
\UnDef, 14, a-4408, a-4418,  
a-4422, a-4424,  
a-4571, a-4595  
\undeksmallskip, c-5427  
\UndoDefaultIndex|  
Exclusions, 17,  
a-5839  
\unexpanded, a-3915,  
a-3916, a-6444,  
a-6461, a-6462,  
a-6464, a-6842,  
a-6856, c-225, c-580,  
c-591, c-1639, c-2322,  
c-2349, c-2366,  
c-2373, c-2375,  
c-2412, c-2413,  
c-2461, c-2539,  
c-2559, c-2560,  
c-2564, c-2565,  
c-2568, c-3084,  
c-3640, c-4306,  
c-4309, c-4311,  
c-4312, c-4315,  
c-4589, c-6540,  
c-6814, c-6819,  
c-6831, c-6832, e-637,  
e-940  
\ungag@index, a-5178,  
a-7972  
\UniformSkips, 19,  
a-2379, a-2400,  
a-2405, a-2412  
\unless, a-2442, a-3189,  
a-3238, a-3253,  
a-6720, c-659, c-1122,  
c-1143, c-1582,  
c-1692, c-1712,

c-2119, c-2137,  
 c-2321, c-4152,  
 c-4156, c-4879,  
 c-4882, c-4886,  
 c-5506, c-5613,  
 c-5900, c-6425, e-764  
`\UnPdef`, 14, a-4416  
`\uparrow`, c-4746  
`\upshape`, c-4470, c-4493,  
 c-5114  
`\Upsilon`, c-4773  
`\upsilon`, c-4804  
`\uresetlinecount`, 10, a-2126  
`\Url`, c-4940, c-4941  
`\url`, c-6602, c-6603  
`\Url@Format`, c-6741, c-6918  
`\Url@HyperHook`, c-6915  
`\Url@moving`, c-6923  
`\Url@z`, c-6913  
`\urladdstar`, c-6599, c-6606  
`\UrlBigBreakPenalty`,  
 c-6798, c-6803  
`\UrlBigBreaks`, c-6734,  
 c-6803  
`\UrlBreakPenalty`,  
 c-6798, c-6801, c-6806  
`\UrlBreaks`, c-6731, c-6801  
`\UrlFix`, b-563, c-6708,  
 c-6710, c-6713,  
 c-6715, c-6718, c-6718  
`\UrlFont`, b-345, c-6742,  
 c-6929  
`\UrlHyphenchar`, c-6756,  
 c-6763  
`\UrlLeft`, c-6751  
`\Urlmuskip`, c-6797  
`\UrlNoBreaks`, c-6735,  
 c-6805  
`\UrlRight`, c-6760  
`\UrlSlashKern`, c-6927  
`\UrlSpecials`, c-6736,  
 c-6807  
`\urlstyle`, c-6921, c-6936  
`\usage`, a-8075  
`\usc`, c-5674, c-5676  
`\usccro`, c-5676  
`\usecounter`, c-3935  
`\UsqEntry`, 21, a-4784,  
 a-8075  
`\uumlaut`, b-444, b-458  
  
`\value`, a-2805  
`\varepsilon`, c-4020,  
 c-4078, c-4787  
`\varnothing`, c-4724, c-5006  
`\varsigma`, c-4802  
`\vartheta`, c-4791  
  
`\vee`, c-4764, c-4880, c-4933  
`\verb`, 22, a-2258, a-3615,  
 a-3623, a-3893,  
 a-5896, c-4164,  
 c-4172, e-486, e-487,  
 e-511, e-517, e-519,  
 e-729, e-907  
`\verb*`, e-729  
`\verb@balance@group`,  
 e-805, e-808, e-841,  
 e-843  
`\verb@egroup`, a-3614,  
 e-808, e-841, e-844  
`\verb@eol@error`, e-814  
`\verb@eolOK`, e-828, e-836  
`\verb@lasthook`, a-5312,  
 e-736, e-737, e-741,  
 e-1093, e-1094  
`\verbatim`, a-8221, e-581  
`\verbatim`, e-581  
`\verb@*`, e-594  
`\verbatim@currenvir`,  
 e-621, e-632, e-633,  
 e-637  
`\verbatim@edef`, e-634,  
 e-640  
`\verbatim@end`, e-635, e-641  
`\verbatim@mathhack`,  
 c-6746, c-6773  
`\verbatim@mathhack@`,  
 c-6775, c-6779  
`\verb@`  
 tim@nolig@list,  
 e-446, e-848  
`\verbatim@specials`,  
 a-3620, a-3621,  
 a-3623, a-3624,  
 a-4970, a-5052,  
 a-5896, a-5897,  
 c-3081, c-6743,  
 c-6745, e-696, e-733,  
 e-1002  
`\verbatim@specials@`,  
 e-1006, e-1010  
`\verb@`  
 tim@specials@iii,  
 e-962, e-964  
`\verb@`  
 tim@specials@iv,  
 e-975, e-980  
`\verb@`  
 tim@specials@list,  
 a-4932, a-4935,  
 c-6774, c-6775, e-971,  
 e-976, e-982, e-987,  
 e-994, e-999, e-1005,  
 e-1006, e-1042  
  
`\verb@`  
 tim@specials@v,  
 e-986, e-992  
`\verb@`  
 tim@specials@vi,  
 e-977, e-988, e-995,  
 e-998  
`\verbatimchar`, 21,  
 a-3877, a-4651,  
 a-4909, a-6212,  
 a-6214, a-8057, 116  
`\verbatimhangindent`,  
 a-2330, e-691, e-707,  
 e-709  
`\verbatimleftskip`,  
 a-8220, e-673, e-703,  
 e-705  
`\verbatimspecials`, 11,  
 e-946, e-1047, 121, 237  
`\verbicodecorr`, a-3257  
`\verbDiscretionary@`  
 Hyphen, e-1109,  
 e-1116, 238  
`\verbbeolOK`, 12, e-836, 236  
`\VerbHyphen`, a-2304,  
 e-351, 236  
`\verbhyphen`, a-2316,  
 a-7442, a-7483,  
 e-345, e-353, e-363,  
 e-388  
`\verbLongDashes`,  
 a-1987, e-1119  
`\VerbT`, e-471  
`\VerbT1`, e-471  
`\vert`, e-1048  
`\vfil`, c-3114  
`\vfilneg`, c-3128  
`\visiblespace`, a-2989,  
 a-7478, c-764, c-768,  
 c-3017, c-6737, e-423,  
 e-938, e-940  
`\VisSpacesGrey`, 11,  
 a-2756, e-934, 237  
`\visspacesgrey`, 237  
`\Vs`, c-5031  
`\vs`, c-3017, c-3023, c-3027  
  
`\wd`, c-3999, c-4002, c-4033,  
 c-4038, c-4046,  
 c-4047, c-4870,  
 c-4995, c-4996,  
 c-5008, c-5009, c-6414  
`\Web`, 23, c-4070  
`\We@`  
 bern@Lieder@ChneOelze,  
 a-4496  
`\wedge`, c-4763, c-4883, c-4933

|                            |                               |                               |
|----------------------------|-------------------------------|-------------------------------|
| \whenonly, c-5803          | \XeTeXmathchardef, c-4533     | \xiiunder, c-678, c-680       |
| \whern, c-6484             | \XeTeXmathcode, c-4536,       | \xleq, c-4692, c-4695, c-4697 |
| \whern@parbox, c-6455,     | c-4541                        | \xxt@visiblespace,            |
| c-6462, c-6501, c-6502     | \XeTeXradical, c-4814         | c-763, c-764                  |
| \wherncore, c-6439,        | \XeTeXthree, b-450, c-4158    | \xxt@visiblespace@fallback,   |
| c-6486, c-6487             | \XeTeXversion, c-180,         | c-765                         |
| \whernfont, c-6446,        | c-181, c-190, c-698,          |                               |
| c-6454, c-6479, c-6489     | c-4151, c-4152,               |                               |
| \whernskip, c-6487,        | c-6093, c-6154,               |                               |
| c-6491, c-6492             | c-6400, e-1046                |                               |
| \whernup, c-6494           | \xgeq, c-4693, c-4700, c-4702 |                               |
| \widowpenalty, a-2488,     | \Xi, c-4770                   |                               |
| c-3116                     | \xi, c-4797                   |                               |
| withmarginpar, 11, a-2188  | \xiiand, c-747                |                               |
| \WPheadings, c-3763        | \xiibackslash, c-734, c-738   |                               |
| \Ws, c-5033                | \xiiclub, a-3655, a-6155,     |                               |
| \wyzejnizej, c-5977        | e-436                         |                               |
| \Wz, c-5035                | \xiishash, a-6721, c-753,     |                               |
|                            | e-1066                        |                               |
| \xathousep, c-6568, c-6582 | \xiilbrace, a-7483,           |                               |
| \xdef@filekey, a-6795,     | a-7485, c-709,                |                               |
| a-6799, a-6819             | c-3260, e-793                 |                               |
| \Xedekfracc, b-554, c-4199 | \xiipercent, a-5489,          |                               |
| \XeLaTeX, c-4097           | a-5491, a-6571,               |                               |
| \XeTeX, 23, c-4090         | a-6575, c-743, e-345          |                               |
| \XeTeXdelcode, c-4615,     | \xiirbrace, c-710             |                               |
| c-4619                     | \xiinspace, a-3891, c-750,    |                               |
| \XeTeXdelimiter, c-4648    | c-768, c-2605, c-2606,        |                               |
| \XeTeXglyph, c-6200        | c-2618                        |                               |
| \XeTeXglyphindex, c-6200   | \xiistring, a-3755,           |                               |
| \XeTeXifprefix, c-4155     | a-4845, a-4903,               |                               |
| \XeTeXinputencoding,       | a-4971, a-5028, c-2604        |                               |
| c-191                      |                               |                               |
|                            |                               | \cdot, c-5536                 |
|                            |                               | \-, c-6114, c-6160, c-6162,   |
|                            |                               | c-6163, c-6164                |
|                            |                               | \--, c-6096, c-6128, c-6159,  |
|                            |                               | c-6162, c-6163, c-6164        |
|                            |                               | \` , e-1051                   |