

Grzegorz ‘Natrór’ Murzynowski

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

October 2008

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	13
Index ex/inclusions	15
The DocStrip directives	16
The changes history	16
The parameters	18
The narration macros	21
A queerness of \label	23
doc-compatibility	23
The driver part	24
The code	26
The package options	26
The dependencies and preliminaries	28
The core	31
Numbering (or not) of the lines .	41
Spacing with \everypar	42
Life among queer EOLs	43
Adjustment of verbatim and \verb	45
Macros for marking of the macros .	46
Automatic detection of definitions	50
Indexing of cses	61
Index exclude list	74
Index parameters	77
The DocStrip directives	79
The changes history	80
The checksum	85
Macros from ltxdoc	87
\DocInclude and the ltxdoc-like setup	87
Redefinition of \maketitle	92
The file's date and version information	95
Miscellanea	96
doc-compatibility	100
gmdocing doc.dtx	105
Polishing, development and bugs .	106
(No) <eof>	107
b. The gmdocc Class For gmdoc	
Driver Files	108
Intro	108
Usage	108
The Code	109
c. The gmutils Package	114
Intro	114
Contents of the gmutils.zip archive	114
A couple of abbreviations	115
\fistofone and the queer \catcodes	115
Global Boolean switches	116
From the ancient xparse of TeXLive 2007	118
Ampulex Compressa-like modifications of macros	122
\@ifnextcat, \@ifnextac	123
\afterfi and pals	125
\gm@undefined—a test that doesn't create any hash entry unlike \@undefined	126
Environments redefined	126
Almost an environment or redefinition of \begin	126
\@ifenvir and improvement of \end	127
From relsize	127
Some 'other' stuff	128
Metasymbols	130
Macros for printing macros and filenames	130
Storing and restoring the meanings of cses	132
Not only preamble!	135
Third person pronouns	136
Improvements to mwcls sectioning commands	137
An improvement of MW's \SetSectionFormatting	139
Negative \addvspace	140

My heading setup for mwcls	141
Compatibilising standard and mwcls sectionings	142
enumerate* and itemize*	144
The logos	145
Expandable turning stuff all into ‘other’	147
Brave New World of X _E T _E X	147
Fractions	148
\reszegraphics	149
Settings for mathematics in main font	149
Minion and Garamond Premier kerning and ligature fixes	153
Varia	153
\@ifempty	157
\include not only .tex’s	157
Faked small caps	158
See above/see below	159
luzniej and napa- pierki—environments used in page breaking for money	160
Typesetting dates in my memoirs .	162
A left-slanted font	167
Thousand separator	168
Storing and restoring the catcodes of specials	169
hyperref’s \nolinkurl into \url* .	169
d. The gmiflink Package	170
Introduction, usage	170
Contents of the gmiflink.zip archive	170
The code	171
e. The gmverb Package	173
Intro, usage	173
Contents of the gmverb.zip archive	174
The code	175
Preliminaries	175
The breakables	175
Almost-Knuthian \ttverbatim .	176
The core: from shortvrb	176
doc- and shortvrb-compatibility .	181
Grey visible spaces	182
f. The gmeometric Package	183
Introduction, usage	183
Contents of the gmeometric.zip archive	183
Usage	184
The code	184
g. The gmoldcomm Package	187
Change History	189
Index	195

a. The gmdoc.sty Package¹

October 6, 2008

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 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.lpl.html>
for the details of that license.

L^APPL status: "author-maintained".

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

```
68 \ifnum\catcode`\\@=11\% Why this test here—will come out in chapter The driver.  
71 \NeedsTeXFormat{LaTeX2e}  
72 \ProvidesPackage{gmdoc}  
73 [2008/10/04 v0.99p a documenting package (GM)]  
74 \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 very little adjustment, in some rather 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, gmdocc.cls and gmold-comm.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.)

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.

¹ This file has version number v0.99p dated 2008/10/04.

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 \Xe\TeX 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 \Xe\TeX on gmdoc.sty once more. (Using \La\TeX instead of \Xe\TeX 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  
gmeometric (for the driver of The  $\text{\La\TeX}_2\epsilon$  Source)
```

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. Palestrina, *Missa papae Marcelli* ;-):

```
source2e_gm.doc.tex  
docstrip_gm.doc.tex  
doc_gm.doc.tex  
  
gmoldcomm.sty  
(gmsource2e.ist is generated from source2e_gm.doc.tex)
```

These drivers typeset the respective files from the
.../texmf-dist/source/latex/base
directory of the $\text{\TeX}{}Live2007$ 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 $\text{\TeX}{}%$ 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 gm.doc 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 authomatic 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 hommage 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 $\text{\TeX}{}book$* ’s meaning, i.e., a control sequence whose meaning is $\backslash(e/g/x)$ 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 $\text{\TeX}{}book$* , I hope: $\text{\TeX}{}%$ is a religion of Book ;-).)

I’ll use a shorthand for ‘control sequence’, cs.

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

Talking of declarations, I’ll use the **OCSR** acronym as a shorthand for ‘observes/ing common $\text{\TeX}{}%$ scoping rules’.

² As Grieg’s Piano Concerto is a hommage to the Schumann’s.

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 \heshe etc. commands provided by gutils), so let one be not surprised if 'he' sees 'herself' altered in the same sentence :-).

Preparing of the source file

When $(\text{\La})\text{\TeX}$ with gmdoc.sty package loaded typesets the comment lines, the code delimiter is ommitted. 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 ommitted.

The user should just write his splendid code and brilliant commentary. In the latter she may use usual $(\text{\La})\text{\TeX}$ 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.

Moreover, if he wants to add a meta-comment i.e., a text that doesn't appear in the code layer nor in the narrative, she 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`\catcodes` 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 (it 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}` ... `\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

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.

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 he 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{mybrilliantpackage.sty}`.

(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 `\DocInclude` command, analogous to L^AT_EX'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`.

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 rel-

`\PrintFilesAuthors`

\SkipFilesAuthors
event \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 filemaker also precedes the index entries, if the (default) codeline index option is in force.

\includeonly As with \include, you may declare \includeonly{\<filenames separated by commas>} for the draft versions.

If you want to put the driver into the same .sty or .cls file (see chapter 641 to see how), you may write \DocInput{\jobname.sty}, or \DocInclude{\jobname.sty}, 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.

At the default settings, the \Doc/SelfIncluded 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)).

\ltxLookSetup
\olddocIncludes 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.

\gmdocIncludes 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.

\AtBeginOnce If you need to add something at the beginning of input of only one file, put before the respective input command an \AtBeginOnce{\<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). \AtBeginOnce 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 <code>(char1)</code> (^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, she may turn it off with the linesnotnum option.
uresetlinecount	However, if he agrees to have the lines numbered, she may wish to reset the counter of lines himself, e.g., when she documents many source files in one document. Then he 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, she 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 he 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 initialized 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.
indexallmacros	By default, gmdoc excludes some 300 cses from being indexed. They are the most common cses, L ^A T _E X internal macros and T _E X primitives. To learn what cses are excluded actually, see lines 5382–5508 .
	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. 15 to feed this ambiguity with a couple of declarations.
withmarginpar	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 allright but not all the classes support marginpars. That is the reason why this package enables marginparing 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	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 documentclass.
codespacesblank \CodeSpacesBlank	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).
codespacesgrey \CodeSpacesGrey	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 viispacesgrey and by default it's defined as {gray}{.5}, you can change it with xcolor's \definecolor. There is also an ocsr declaration \CodeSpacesGrey.
\VisSpacesGrey	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 a little tricks, you can also specify codespacesgrey, codespacesblank in gmdoc options (in this order).
The packages required	
gmverb	gmdoc requires (loads if they're not loaded yet) some other packages of mine, namely gmuilts, gmverb, analogous to Frank Mittelbach's shortvrb, and gmiflink for conditional making of hyperlinks. It also requires hyperref, multicol, color and makeidx.
\verb+eolOK	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.
\MakeShortVerb	As the standard LATEX 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+eolOK declaration. The plain \verb typesets spaces blank and \verb* makes them visible, as in the standard version(s).
\dekclubs	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*\\
\DeleteShortVerb	(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 <i>The TExbook</i> 's format, gmverb provides a macro \dekclubs that expands to the example displayed above.
gmuilts	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 shortvrb 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.
	The gmuilts package provides a couple of macros similar to some basic (L)ATEX ones, rather strictly technical and (I hope) tricky, such as \afterfi, \ifnextcat, \addtomacro etc. It's this package that provides the macros for formatting of names of macros and files, such as \cs, \marg, \pk etc.

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 her favourite options <i>before</i> loading gmdoc. If he does not, gmdoc shall load it with <i>my</i> 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,`
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`.

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.

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 \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*\}@{*optionname*}. 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.

\DeclareKVOFam star Next key recognized by \DeclareDefining 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.

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 \c@namedef to be detected (the original L^AT_EX version), declare

```
\DeclareDefining*[star=false]\c@namedef
```

or

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

(as stated above, * is equivalent [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 \UnDef. To resume detection of \def you are provided also a shorthand, \ResumeDef (but \ResumeDefining\def also works).

If you define things not with easily detectable commands, you can mark them 'manually', with the \Def ine 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 $\wedge\wedge A$.

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. 10).

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.

All three take one argument and all three may be starred. The non-starred versions are intended to take a control sequence as the argument and the starred to take whatever (an environment name or a $\wedge\wedge A$ -like and also a cs).

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 unstarring.

The \Define and \CodeUsage commands, if unstarring, 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 look 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 $\wedge\wedge A$ -likes and cses, too. Currently, the most important difference is that the unstarring 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?

`\CodeMarginize` 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.

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.

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.

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`.

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 its argument—it's indexed automatically anyway). It's `\outer`.

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

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

It's understandable⁴ 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

⁴ After reading `doc`'s documentation ;-).

takes one argument consisting of a list of control sequences not to be indexed. The items 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 \DoNotIndexes 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{\langle version\rangle}{\langle YYYY/MM/DD date\rangle}{\langle text\rangle}
```

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 \langle date\rangle in parentheses and appends the \langle text\rangle 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 (gmdoc.cls does it for you)]. To read in and print the sorted change history (in two columns), 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 [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 analagous to the \Index... versions [see sec. [The parameters](#) p. 20]. (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, 17
```

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{\langle version\rangle}{\langle year\rangle/\langle month\rangle/\langle day\rangle}
```

declaration.

If the ChangesStartDate is set to a date contemporary to T_EX i.e., not earlier than September 1982⁵, then a note shall appear at the beginning of the changes history that informs the reader of ommitting 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{\langle version?\rangle}{1000/00/00}
```

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

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{\langle number\rangle} before the code and T_EX 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.

⁵ DEK in *T_EX The Program* mentions that month as of T_EX Version 0 release.

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

```
% \chshchange{vo.98j}{2006/10/19}{4372}  
% \chshchange{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}*.

The parameters

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

$\backslash\stanzaskip$

$\backslash\stanzaskip$ is a vertical space inserted when a blank (code) line is met. It's equal $0.75\medskipamount$ by default (with the *entire* \medskipamount 's stretch- and shrinkability). Subsequent blank code lines do not increase this space.

$\backslash\CodeTopsep$

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

$\backslash\UniformSkips$
 $\backslash\NonUniformSkips$

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

$\backslash\stanza$
 $\backslash\chunkskip$

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

$\backslash\nostanza$

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

$\backslash\CodeIndent$
 $\backslash\TextIndent$

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

There's also a parameter for the indent of the narration, $\backslash\TextIndent$, but you should use it only in emergency (otherwise what would be the margins for?). It's 0sp by default.

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

$\backslash\EOFMark$

given by the $\backslash\EOFMark$ macro.

$\backslash\everyeof$

If you do use the ε - \TeX 's primitive $\backslash\everyeof$, be sure the contents of it begins with $\backslash\relax$ 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 when- and wherever you change your mind you should express it with a new \CodeDelim declaration.

The starred version of \CodeDelim changes also the verb ‘hyphen’, the char appearing at the verbatim line breaks that is.

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 \ (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 codelinenumber so the macro printing it is \thecodelinenumber. By default we don’t change its L^AT_EX’s definition which is equivalent \arabic{codelinenumber}.

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 subentries, e.g., the name of the package. The third may or even should store a text to distinguish equal codeline 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 ltxdoc 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).

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 641, 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.

The text put at the beginning of the index is declared with a one-argument `\IndexPrologue`. Its default text at current index option you may [admire](#) on page 195. 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 `\AtDIProllogue` 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 [5620](#) 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.

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 `\OnlyDescription` 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 inline comments gmdoc provides the \inverb<charX>...<charX> (the name stands for 'inline 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 inline 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.
\nlpercent	And for line breaking at \cs and \env there is \nlpercent to ensure % if the line breaks at the beginning of a \cs or \env and \+ to use inside their 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 \+.
\ilrr	By default the multiline inline 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 inline comment to be justified. Moreover, because of the hanging indent the lines of multiline inline comments are relatively short, you may get lots of overfulls. Therefore there is a Boolean switch \ilrr (ocsr), whose name stands for 'InLine RaggedRight' and the inline comments (and their codelines) are typeset justified in the scope of \ilrrfalse which is the default. When you write \ilrrtrue, then all inline 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 inline 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 standalone narration.
\pk	To print packages' names sans serif there is a \pk one-argument command, and the \file command intended for the filenames.
\catletter	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.
\catactive	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.
\division	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.
\subdivision	
\subsubdivision	

It's resolved in `gmutils` 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 `scls`⁶). 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` 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` (`gmutils`, to be precise) provides some `TEX`-related logos:

<code>\AmSTeX</code>	typesets A M S -T _E X,
<code>\BibTeX</code>	B IBT _E X,
<code>\SliTeX</code>	S LI T _E X,
<code>\PlainTeX</code>	P LAIN T _E X,
<code>\Web</code>	W E B,
<code>\TeXbook</code>	<i>The T_EXbook</i> ,
<code>\TB</code>	<i>The T_EXbook</i>
<code>\eTeX</code>	ϵ -T _E X,
<code>\pdfTeX</code>	pdf ϵ -T _E X
<code>\pdfTeX</code>	pdfT _E X
<code>\XeTeX</code>	X ϵ T _E X (the first E will be reversed if the graphics package is loaded or X ϵ T _E X is at work)
<code>\LaTeXpar</code>	and
<code>\ds</code>	(L)A T _E X.
<code>\copyrnote</code>	DocStrip not quite a logo, but still convenient.

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

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`.

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 `\MacroTopsep`. Both of them take care of not cumulating the `vspaces`.

`\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` 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).

⁶ See `gmutils` for some subtle details.

\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\.

\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{\mcnumber}{\stuff} defines control sequence \gmd@mchook{\mcnumber} with the meaning \stuff which is put at the end of macrocode and oldmc number \mcnumber (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 she 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 \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-compatibiliy](#).

1828 {*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]{gmdoc}
\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 sanitizes the stuff. More precisely, it uses the \dospecials list, so it sanitizes also the braces.

Moreover, when the countalllines(*) option is in force, \skipfiles... \endskipfiles 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.

```

1879 \ifnum\catcode`@=12
1882 \documentclass[countalllines, codespacesgrey, outeroff, debug,
      mwrep,
1883 pagella, fontsing=quiet]{gmdocc}
1885 \twocoltoc
1886 \title{The \pk{gmdoc} Package \i.e., \pk{gmdoc.sty} and
1887 \pk{gmdocc.cls}}
1888 \author{Grzegorz `Natrór' Murzynowski}
1889 \date{\ifcase\month\relax\or January\or February\or March\or
      April\or May\or
1890 June\or July\or August\or September\or October\or November\or
1891 December\fi\,2008}
      \%includeonly{gmoldcomm}
1895 \begin{document}
1901 \maketitle
1903 \setcounter{page}{2}% hyperref cries if it sees two pages numbered 1.
1905 \tableofcontents
1906 \DoIndex\maketitle
1909 \SelfInclude
1911 \DocInclude{gmdocc}

```

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

```

1915 \skipgmlonely[\stanza, The remarks about installation and
      compiling
1916   of the documentation are analogous to those in the chapter
1917   \pk{gmdoc.sty} and therefore omitted. \stanza]
1918 \DocInclude{gmutils}
1919 \DocInclude{gmiflink}
1920 \DocInclude{gmverb}
1921 \DocInclude{gmeometric}
1922 \DocInclude{gmoldcomm}
1923 \typeout{%
  Produce change log with^^J%
  makeindex -r -s gmglolist -o \jobname.gls \jobname.glo^^J

```

```

1926  (gmglo.list should be put into some texmf/makeindex_
      directory.)^^J}
1927  \PrintChanges
1928  \typeout{%
1929    Produce_index_with^^J%
1930    makeindex -r \jobname^^J}
1931  \PrintIndex
1933  \afterfi{%
1934  \end{document}}

```

MakeIndex shell commands:

```

1936  makeindex -r gmdoc
1937  makeindex -r -s gmglo.list -o gmdocDoc.gls gmdocDoc.glo
(gmglo.list should be put into some texmf/makeindex directory.)

```

And “That’s all, folks” ;-).

```

1944 }\fi% of \ifnum\catcode`@=12, of the driver that is.

```

The code

For debug

```

1954 \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 ^{^^M} (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 his splendid code and adds a brilliant comment in %ed lines and that’s all. Of course, if she wants to make a \section or \emphasise, he 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 beginnig 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

```

2003 \RequirePackage{gmutils}[2008/08/30]%
  includes redefinition of \newif to
  make the switches \protected.
2005 \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 2010 \newif\if@linesnotnum  
linesnotnum 2012 \DeclareOption{linesnotnum}{\@linesnotnumtrue}  
And maybe he or she wishes to declare resetting the line counter along with some  
sectioning counter him/herself.  
\if@uresetlinecount 2017 \newif\if@uresetlinecount  
uresetlinecount 2019 \DeclareOption{uresetlinecount}{\@uresetlinecounttrue}  
And let the user be given a possibility to count the comment lines.
```

```
\if@countalllines 2024 \newif\if@countalllines  
\if@printalllinenos 2025 \newif\if@printalllinenos  
countalllines 2027 \DeclareOption{countalllines}{%  
2028 \@countalllinestrue  
2029 \@printalllinenosfalse}  
countalllines* 2031 \DeclareOption{countalllines*}{%  
2032 \@countalllinestrue  
2033 \@printalllinenostrue}
```

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

```
\if@noindex 2039 \newif\if@noindex  
noindex 2041 \DeclareOption{noindex}{\@noindextrue}  
\if@pageindex 2044 \newif\if@pageindex  
pageindex 2046 \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 2053 \newif\if@indexallmacros  
indexallmacros 2055 \DeclareOption@indexallmacros{\@indexallmacrostrue}
```

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

```
\if@marginparsused 2065 \@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:

```
2072 \@ifclassloaded{article}{\@marginparsusedtrue}{  
2075 \@ifclassloaded{report}{\@marginparsusedtrue}{  
2077 \@ifclassloaded{book}{\@marginparsusedtrue}{}}
```

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

```
withmarginpar 2080 \DeclareOption{withmarginpar}{\@marginparsusedtrue}  
nomarginpar 2082 \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 2092 \DeclareOption{codespacesblank}{%
2093   \AtEndOfPackage{\% to allow codespacesgrey, \code{codespacesblank}
2094   \AtBeginDocument{\CodeSpacesBlank}}}

codespacesgrey 2097 \DeclareOption{codespacesgrey}{%
2100   \AtEndOfPackage{\% to put the declaration into the begin-document hook after
2101     definition of \visible{space}.}
2102   \AtBeginDocument{\CodeSpacesGrey}}}

2104 \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`)

```
2113 \RequirePackage{gmutils}[2008/08/08]
```

A standard package for defining colours,

```
2116 \RequirePackage{xcolor}
```

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

```
2118 \definecolor{deepblue}{rgb}{0,0,.85}
```

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

```

2131 \@ifpackageloaded{hyperref}{\hypersetup{colorlinks=true,
2132   linkcolor=deepblue, urlcolor=blue, filecolor=blue}}{%
2133   \RequirePackage[colorlinks=true, linkcolor=deepblue, 
2134     urlcolor=blue,
2135     filecolor=blue, pdfstartview=FitH, pdfview=FitBH,
2136     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`.

```
2145 \RequirePackage{gmiflink}
```

And a slight redefinition of `verbatim`, `\verb(*)` and providing of `\MakeShortVerb(*)`.

```
2148 \RequirePackage{gmverb}[2008/08/20]
```

```
2150 \if@noindex
```

```
2151   \AtBeginDocument{\gag@index}\% for the latter macro see line 4913.
```

```
2153 \else
```

```
2154   \RequirePackage{makeidx}\makeindex
```

```
2155 \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 2186 \def\CodeDelim{\@ifstar\Code@Delim@St\Code@Delim}
\Code@Delim 2188 \def\Code@Delim#1{%
2189   {\escapechar\m@ne
2190     \xa\gdef\@xa\code@delim\@xa{\string#1}}}
(\@xa is \expandafter, see gutils.)
\Code@Delim@St 2193 \def\Code@Delim@St#1{\Code@Delim{#1}\VerbHyphen{#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:

```
2202 \CodeDelim*\%
```

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 2210 \newtoks\gmd@preverypar
\settexcodehangi 2212 \newcommand*\settexcodehangi{%
2213   \hangindent=\verbatimhangindent\hangafter=\@ne}%
we'll use it in the
      inline comment case. \verbatimhangindent is provided by the gmverb
      package and = 3em by default.
2217 \@ifndef\@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):

```
2223 \newlength\TextIndent
```

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

```
2227 \newlength\CodeIndent
2230 \CodeIndent=1.5em\relax
```

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

```
2233 \@ifundefined{stanzaskip}{\newlength\stanzaskip}{}%
```

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

```
2238 \stanzaskip=\medskipamount
2239 \advance\stanzaskip by-.25\medskipamount% to preserve the stretch- and shrink-
      ability.
```

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

```
2245 \newskip\CodeTopsep
2246 \newskip\MacroTopsep
```

And let's set them. For aesthetic minimality⁷ 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.

⁷ 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).

```

\UniformSkips 2262 \def\UniformSkips{%
\CodeTopsep 2264 \CodeTopsep=\stanzaskip
\MacroTopsep 2265 \MacroTopsep=\stanzaskip
2266 \abovedisplayskip=\stanzaskip
%%\abovedisplayshortskip remains untouched as it is 0.0pt plus 3.0pt by default.
2271 \belowdisplayskip=\stanzaskip
2272 \belowdisplayshortskip=.5\stanzaskip% due to DEK's idea of making the
    short below display skip half of the normal.
2274 \advance\belowdisplayshortskip\by\smallskipamount
2275 \advance\belowdisplayshortskip\by-\smallskipamount% We advance \be-
    % lowdisplayshortskip forth and back to give it the \smallskipamount's
    shrink- and stretchability components.
2279 \topsep=\stanzaskip
2280 \partopsep=\z@
2281 }

We make it the default,

```

2283 \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.

2288 \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.

2295 \newcommand*\NonUniformSkips{\relax\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 2305 \newcommand*\chunkskip{%
2306   \skipo=\MacroTopsep
2307   \if@codeskipput\advance\skipo\by-\CodeTopsep\fi
2308   \par\addvspace{\skipo}\@codeskipputtrue}

```

And, for a smaller part of text,

```

\stanza 2311 \newcommand*\stanza{%
2312   \skipo=\stanzaskip
2313   \if@codeskipput\advance\skipo\by-\CodeTopsep\fi
2314   \par\addvspace{\skipo}\@codeskipputtrue}

```

Since the stanza skips are inserted automatically most often (cf. lines 2729, 3156, 2749, 3037, 3209), sometimes you may need to forbid them.

```

\nostanza 2319 \newcommand*\nostanza{%
2320   \par
2321   \if@codeskipput\unless\if@nostanza\vskip-\CodeTopsep\relax\fi%
2322   \fi
2323   \@codeskipputtrue\@nostanzagtrue
2324   \@afternarrgfalse\@aftercodegtrue}%
    In the 'code to narration' case the
    first switch is enough but in the countercase '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
2331 \newgif\if@newline

\newgif is \newif with 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 2339 \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.

\DocInput 2352 \newcommand*\DocInput{\bgroup\@makeother_\\Doc@Input}
2354 \begingroup\catcode`\\^M=\active%
2355 \firstofone{\endgroup%
\Doc@Input 2356 \newcommand*{\Doc@Input}[1]{\egroup\begingroup%
2359 \edef\gmd@inputname{\#1}% we'll use it in some notifications.
2361 \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.
2366 \gmd@setclubpenalty% we wrapped the assignment of \clubpenalty in
a macro because we'll repeat it twice more.
2368 \@clubpenalty\clubpenalty\widowpenalty=3333% Most paragraphs of
the code will be one-line most probably and many of the narration, too.

2373 \tolerance=1000% as in doc.
2376 \@xa\@makeother\csname\code@delim\endcsname%
2378 \gmd@resetlinecount% due to the option uresetlinecount we reset the
linenumber counter or do nothing.
2381 ^M \QueerEOL% It has to be before the begin-input-hook to allow change by that
hook.
2386 \@beginputhook% my first use of it is to redefine \maketitle just at this point
not globally.
2388 \everypar=\@xa{\@xa\codetonarrskip\the\everypar}-%
2390 \edef\gmd@guardedinput{
2391 @nx\@@input\#1\relax% @nx is \noexpand, see gutils. \@@input is the
true TeX's \input.
2395 \gmd@iihook% cf. line 6935
2396 @nx\EOFMark% to pretty finish the input, see line 2556.
2398 @nx\CodeDelim\@xa\@nx\csname\code@delim\endcsname% to ensure the
code delimiter is the same as at the beginning of input.
2403 @nx^M\code@delim%
2405 }% we add guardians after \inputing a file; somehow an error occurred without
them.
2407 \catcode`\%=9% for doc-compatibility.
2408 \setcounter{CheckSum}{0}% we initialize the counter for the number of the
escape chars (the assignment is \global).

```

2410   \everyeof{\relax}%
2411   \cxa\cxa\cxa^M\gmd@guardedinput%
2412   \par%
2413   \cendinpushhook% It's a hook to let postpone some stuff till the end of input.
2414   We use it e.g. for the doc-(not)likeliness notifications.
2415   \glet\currentlabel=\gmd@currentlabel@before% we restore value from
2416   before this group. In a very special case this could cause unexpected be-
2417   haviour of crossrefs, but anyway we acted globally and so acts hyperref.
2418   \endgroup%
2419 }% end of \Doc@Input's definition.
2420 }% 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  $\wedge M$  will be let to.  $\gmd@textEOL$  will be used also for checking the  $\wedge M$  case ( $\@ifnextchar$  does  $\ifx$ ).

\gmd@textEOL 2433 \pdef\gmd@textEOL{\u a space just like in normal TeX. We put it first to cooperate
2434   with  $\wedge M$ 's  $\expandafter\ignorespaces$ . It's no problem since a space  $\u 10$ 
2435   doesn't drive TeX out of the vmode.
2436   \ifhmode\@afternarrgtrue\@codeskipputfalse\f i% being in the horizontal
2437   mode means we've just typeset some narration so we turn the respective
2438   switches: the one bringing the message 'we are after narration' to
2439   True (@afternarr) and the 'we have put the code-narration glue' to False
2440   (@codeskipput). Since we are in a verbatim group and the information
2441   should be brought outside it, we switch the switches globally (the letter g in
2442   both).
2443   \@newlinegtrue% to \refstep the lines' counter at the proper point.
2444   \@dsdirgtrue% to handle the DocStrip directives.
2445   \@xa\@trimandstore\the\everypar\@trimandstore% we store the previous
2446   value of \everypar register to restore it at a proper point. See line 3245 for
2447   the details.
2448   \begingroup%
2449   \gmd@setclubpenalty% Most paragraphs will be one-line most probably. Since
2450   some sectioning commands may change \clubpenalty, we set it again here
2451   and also after this group.
2452   \aftergroup\gmd@setclubpenalty%
2453   \let\par\@@par% inside the verbatim group we wish \par to be genuine.
2454   \ttverbatim% it does \tt and makes specials other or \active-and-breakable.
2455   \gmd@DoTeXCodeSpace%
2456   \@makeother\|% because \ttverbatim doesn't do that.
2457   \MakePrivateLetters% see line 3506.
2458   \@xa\@makeother\code@delim% we are almost sure the code comment char is
2459   among the chars having been 12ed already. For 'almost' see the \IndexInput
2460   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.

2461   \@xa\@ifnextchar\@xa{\code@delim}{%
2462     \gmd@continuenarration}%
2463     \gmd@dolspaces% it will launch \gmd@typesettexcode.
2464   }% end of \@ifnextchar's else.
2465 }% end of \gmd@textEOL's definition.

```

```

\gmd@setclubpenalty 2484 \def\gmd@setclubpenalty{\clubpenalty=3333}
For convenient adding things to the begin- and endinput hooks:
\AtEndInput 2488 \def\AtEndInput{\g@addto@macro\@endinpushhook}
\@endinpushhook 2489 \def\@endinpushhook{}

Simili modo

\AtBeginInput 2492 \def\AtBeginInput{\g@addto@macro\@beginpushhook}
\@beginpushhook 2493 \def\@beginpushhook{}

For the index input hooking now declare a macro, we define it another way at line
6935.
2497 \emptyify\gmd@iihook

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

```

```

\tableofcontents 2502 \AtBeginInput{\let\gmd@toc\tableofcontents
2503   \def\tableofcontents{%
2504     \@ifQueerEOL{\StraightEOL\gmd@toc\QueerEOL}%
2505     {\gmd@toc}}}

As you'll learn from lines 3341 and 3328, 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
 $\wedge M$  is the rôle of \ifEOLactive:
```

```

@ifEOLactive 2517 \long\def\ifEOLactive#1#2{%
2518   \ifnum\catcode` $\wedge M$ =\active\afterfi{#1}\else\afterfi{#2}\fi}
2520 \foone\obeylines{%
@ifQueerEOL 2521 \long\def\ifQueerEOL#1#2{%
2522   \ifEOLactive{\ifx $\wedge M$ \gmd@textEOL\afterfi{#1}\else\afterfi{%
2523     #2}\fi}%
2523   {#2}}% of \ifQueerEOL
2524 }% of \foone

```

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

```

\gmd@ABIOnce 2535 \emptyify\gmd@ABIOnce
2536 \AtBeginInput\gmd@ABIOnce
\AtBeginInputOnce 2538 \long\def\AtBeginInputOnce#1{%
2551   \gaddtomacro\gmd@ABIOnce{\emptyify\gmd@ABIOnce#1}}
Many tries of finishing the input cleanly led me to setting the guardians as in line
2403 and to

```

```

\EOFMark 2556 \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 7730.

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

```

\gmd@DoTeXCodeSpace 2568 \begin{obeyspaces}%
2569 \gdef\CodeSpacesVisible{%
2570 \def\gmd@DoTeXCodeSpace{%
2571   \obeyspaces\let_=\\breakablevisspace}}%

```

```

\CodeSpacesBlank 2578 \gdef\CodeSpacesBlank{%
2579   \let\gmd@DoTeXCodeSpace\gmobeyspaces%
2580   \let\gmd@texcodespace=\% the latter \let is for the \if...s.

\CodeSpacesSmall 2583 \gdef\CodeSpacesSmall{%
2584   \def\gmd@DoTeXCodeSpace{%
2585     \obeyspaces\def{\, \hskip\z@\}}%
2586   \def\gmd@texcodespace{\, \hskip\z@\}}%
2588   \end{obeyspaces}
\CodeSpacesGrey 2590 \def\CodeSpacesGrey{%
2593   \CodeSpacesVisible
2594   \VisSpacesGrey% defined in gmverb
2595 }%
2596 Note that \CodeSpacesVisible doesn't revert \CodeSpacesGrey.
2600 \CodeSpacesVisible

```

How the continuing of the narration should look like?

```

\gmd@continuenarration 2604 \def\gmd@continuenarration{%
2605   \endgroup
2606   \gmd@cpnarrline% see below.
2607   \xa{@trimandstore\the\everypar@trimandstore
2608   \everypar=\xa{\xa{@codetonarrskip\the\everypar}%
2609   \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 2477. And in line 2857 it's been read as 12. That's why it works in spite of that % is of category 'ignored'.)

```
2616 \if@countalllines
```

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

```

\gmd@countnarrline@ 2626 \def\gmd@countnarrline@{%
2627   \gmd@grefstep{codelinenum}@newlinefalse
2628   \everypar=\xa{%
2629     \xa{@codetonarrskip\the\gmd@preverypar}% the \hyperlabel@-
% line macro puts a hypertarget in a \raise i.e., drives TeX into
% the horizontal mode so \everypar shall be issued. Therefore we
% should restore it.
2634   }% of \gmd@countnarrline@
\gmd@grefstep@ 2636 \def\gmd@grefstep#1{%
2637   instead of diligent redefining all possible commands
2638   and environments we just assign the current value of the respective TeX's
2639   primitive to the codelinenum counter. Note we decrease it by -1 to get
2640   the proper value for the next line. (Well, I don't quite know why, but it
2641   works.)
2642   \ifnum\value{#1}<\inputlineno
2643     \csname_c@#1\endcsname\numexpr\inputlineno-1\relax
2644     \ifvmode\leavevmode\fi% this line is added 2008/08/10 after an all-
2645     night debuggery ;-) that showed that at one point \gmd@grefstep
2646     was called in vmode which caused adding \penalty 10000 to
2647     the main vertical list and thus forbidding pagebreak during entire
2648     % oldmc.

```

```

2651      \grefstepcounter{#1}%
2652      \fi}%
2653      We wrap stepping the counter in an \ifnum to avoid repetition of
2654      the same ref-value (what would result in the “multiply defined labels”
2655      warning).
2656
2657      The \grefstepcounter macro, defined in gmverb, is a global version of \refstep-
2658      counter, observing the redefinition made to \refstepcounter by hyperref.
2659
2660      \if@printalllinenos% Note that checking this switch makes only sense when
2661      countalllines is true.
2662
2663      \def\gmd@cpnarrline{%
2664          \if@newline
2665              \gmd@countnarrline@
2666              \hyperlabel@line
2667              {\LineNumFont\thecodelinenumber}\,,\ignorespaces}%
2668          \fi}
2669
2670      \else% not printalllinenos
2671          \emptyify\gmd@cpnarrline
2672      \fi
2673
2674      \def\gmd@ctallsetup{%
2675          In the oldmc environments and with the \FileInfo declaration
2676          (when countalllines option is in force) the code is gobbled
2677          as an argument of a macro and then processed at one place (at the end
2678          of oldmc e.g.) so if we used \inputlineno, we would have got all the
2679          lines with the same number. But we only set the counter not \refstep
2680          it to avoid putting a hypertarget.
2681          \setcounter{codelinenumber}{\inputlineno}%
2682          it's global.
2683          \let\gmd@grefstep\hgrefstepcounter}
2684
2685      \else% not countalllines (and therefore we won't print the narration lines' num-
2686          bers either)
2687          \emptyify\gmd@cpnarrline
2688          \let\gmd@grefstep\hgrefstepcounter%
2689          if we don't want to count all the lines,
2690          we only \ref-increase the counter in the code layer.
2691          \emptyify\gmd@ctallsetup
2692      \fi% of \if@countalllines
2693
2694      \def\skipelines{\bgroup
2695          \let\do\@makeother\dospecials%
2696          not \@sanitize because the latter
2697          doesn't recatcode braces and we want all to be quieten.
2698          \gmd@skipelines}
2699
2700      \edef\gmu@tempa{%
2701          \long\def\@nx\gmd@skipelines##1\bslash\endskipelines{\egroup}}
2702      \gmu@tempa
2703
2704      And typesetting the TEX code?
2705
2706      \foone\obeylines{%
2707          \def\gmd@typesettexcode{%
2708              \gmd@parfixclosingspace%
2709              it's to eat a space closing the paragraph, see be-
2710              low. It contains \par.
2711
2712          \everypar={\@settexcodehangi}%
2713          At first attempt we thought of giving
2714          the user a \toks list to insert at the beginning of every code line, but
2715          what for?
2716
2717
2718
2719
2720
2721
2722
2723
2724
2725
2726
2727
2728
2729
2730
2731
2732
2733
2734
2735
2736
2737
2738
2739
2740
2741
2742
2743
2744
2745
2746
2747
2748
2749
2750
2751
2752
2753
2754
2755
2756
2757
2758
2759
2760
2761
2762
2763
2764
2765
2766
2767
2768
2769
2770
2771
2772
2773
2774
2775
2776
2777
2778
2779
2780
2781
2782
2783
2784
2785
2786
2787
2788
2789
2790
2791
2792
2793
2794
2795
2796
2797
2798
2799
2800
2801
2802
2803
2804
2805
2806
2807
2808
2809
2810
2811
2812
2813
2814
2815
2816
2817
2818
2819
2820
2821
2822
2823
2824
2825
2826
2827
2828
2829
2830
2831
2832
2833
2834
2835
2836
2837
2838
2839
2840
2841
2842
2843
2844
2845
2846
2847
2848
2849
2850
2851
2852
2853
2854
2855
2856
2857
2858
2859
2860
2861
2862
2863
2864
2865
2866
2867
2868
2869
2870
2871
2872
2873
2874
2875
2876
2877
2878
2879
2880
2881
2882
2883
2884
2885
2886
2887
2888
2889
2890
2891
2892
2893
2894
2895
2896
2897
2898
2899
2900
2901
2902
2903
2904
2905
2906
2907
2908
2909
2910
2911
2912
2913
2914
2915
2916
2917
2918
2919
2920
2921
2922
2923
2924
2925
2926
2927
2928
2929
2930
2931
2932
2933
2934
2935
2936
2937
2938
2939
2940
2941
2942
2943
2944
2945
2946
2947
2948
2949
2950
2951
2952
2953
2954
2955
2956
2957
2958
2959
2960
2961
2962
2963
2964
2965
2966
2967
2968
2969
2970
2971
2972
2973
2974
2975
2976
2977
2978
2979
2980
2981
2982
2983
2984
2985
2986
2987
2988
2989
2990
2991
2992
2993
2994
2995
2996
2997
2998
2999
2999
3000
3001
3002
3003
3004
3005
3006
3007
3008
3009
3010
3011
3012
3013
3014
3015
3016
3017
3018
3019
3020
3021
3022
3023
3024
3025
3026
3027
3028
3029
3030
3031
3032
3033
3034
3035
3036
3037
3038
3039
3040
3041
3042
3043
3044
3045
3046
3047
3048
3049
3050
3051
3052
3053
3054
3055
3056
3057
3058
3059
3060
3061
3062
3063
3064
3065
3066
3067
3068
3069
3070
3071
3072
3073
3074
3075
3076
3077
3078
3079
3080
3081
3082
3083
3084
3085
3086
3087
3088
3089
3090
3091
3092
3093
3094
3095
3096
3097
3098
3099
3099
3100
3101
3102
3103
3104
3105
3106
3107
3108
3109
3110
3111
3112
3113
3114
3115
3116
3117
3118
3119
3120
3121
3122
3123
3124
3125
3126
3127
3128
3129
3130
3131
3132
3133
3134
3135
3136
3137
3138
3139
3139
3140
3141
3142
3143
3144
3145
3146
3147
3148
3149
3149
3150
3151
3152
3153
3154
3155
3156
3157
3158
3159
3159
3160
3161
3162
3163
3164
3165
3166
3167
3168
3169
3169
3170
3171
3172
3173
3174
3175
3176
3177
3178
3179
3179
3180
3181
3182
3183
3184
3185
3186
3187
3188
3189
3189
3190
3191
3192
3193
3194
3195
3196
3197
3198
3199
3199
3200
3201
3202
3203
3204
3205
3206
3207
3208
3209
3209
3210
3211
3212
3213
3214
3215
3216
3217
3218
3219
3219
3220
3221
3222
3223
3224
3225
3226
3227
3228
3229
3229
3230
3231
3232
3233
3234
3235
3236
3237
3238
3239
3239
3240
3241
3242
3243
3244
3245
3246
3247
3248
3249
3249
3250
3251
3252
3253
3254
3255
3256
3257
3258
3259
3259
3260
3261
3262
3263
3264
3265
3266
3267
3268
3269
3269
3270
3271
3272
3273
3274
3275
3276
3277
3278
3279
3279
3280
3281
3282
3283
3284
3285
3286
3287
3288
3289
3289
3290
3291
3292
3293
3294
3295
3296
3297
3298
3299
3299
3300
3301
3302
3303
3304
3305
3306
3307
3308
3309
3309
3310
3311
3312
3313
3314
3315
3316
3317
3318
3319
3319
3320
3321
3322
3323
3324
3325
3326
3327
3328
3329
3329
3330
3331
3332
3333
3334
3335
3336
3337
3338
3339
3339
3340
3341
3342
3343
3344
3345
3346
3347
3348
3349
3349
3350
3351
3352
3353
3354
3355
3356
3357
3358
3359
3359
3360
3361
3362
3363
3364
3365
3366
3367
3368
3369
3369
3370
3371
3372
3373
3374
3375
3376
3377
3378
3379
3379
3380
3381
3382
3383
3384
3385
3386
3387
3388
3388
3389
3390
3391
3392
3393
3394
3395
3396
3397
3398
3398
3399
3399
3400
3401
3402
3403
3404
3405
3406
3407
3408
3409
3409
3410
3411
3412
3413
3414
3415
3416
3417
3418
3419
3419
3420
3421
3422
3423
3424
3425
3426
3427
3428
3429
3429
3430
3431
3432
3433
3434
3435
3436
3437
3438
3438
3439
3439
3440
3441
3442
3443
3444
3445
3446
3447
3448
3448
3449
3449
3450
3451
3452
3453
3454
3455
3456
3457
3458
3459
3459
3460
3461
3462
3463
3464
3465
3466
3467
3468
3469
3469
3470
3471
3472
3473
3474
3475
3476
3477
3478
3478
3479
3479
3480
3481
3482
3483
3484
3485
3486
3487
3488
3488
3489
3489
3490
3491
3492
3493
3494
3495
3496
3497
3497
3498
3498
3499
3499
3500
3501
3502
3503
3504
3505
3506
3507
3508
3509
3509
3510
3511
3512
3513
3514
3515
3516
3517
3518
3519
3519
3520
3521
3522
3523
3524
3525
3526
3527
3528
3529
3529
3530
3531
3532
3533
3534
3535
3536
3537
3538
3538
3539
3539
3540
3541
3542
3543
3544
3545
3546
3547
3548
3548
3549
3549
3550
3551
3552
3553
3554
3555
3556
3557
3558
3559
3559
3560
3561
3562
3563
3564
3565
3566
3567
3568
3569
3569
3570
3571
3572
3573
3574
3575
3576
3577
3578
3579
3579
3580
3581
3582
3583
3584
3585
3586
3587
3588
3588
3589
3589
3590
3591
3592
3593
3594
3595
3596
3597
3597
3598
3598
3599
3599
3600
3601
3602
3603
3604
3605
3606
3607
3608
3609
3609
3610
3611
3612
3613
3614
3615
3616
3617
3618
3619
3619
3620
3621
3622
3623
3624
3625
3626
3627
3628
3629
3629
3630
3631
3632
3633
3634
3635
3636
3637
3638
3638
3639
3639
3640
3641
3642
3643
3644
3645
3646
3647
3648
3648
3649
3649
3650
3651
3652
3653
3654
3655
3656
3657
3658
3659
3659
3660
3661
3662
3663
3664
3665
3666
3667
3668
3669
3669
3670
3671
3672
3673
3674
3675
3676
3677
3678
3678
3679
3679
3680
3681
3682
3683
3684
3685
3686
3687
3688
3688
3689
3689
3690
3691
3692
3693
3694
3695
3696
3697
3697
3698
3698
3699
3699
3700
3701
3702
3703
3704
3705
3706
3707
3708
3709
3709
3710
3711
3712
3713
3714
3715
3716
3717
3718
3719
3719
3720
3721
3722
3723
3724
3725
3726
3727
3728
3728
3729
3729
3730
3731
3732
3733
3734
3735
3736
3737
3738
3738
3739
3739
3740
3741
3742
3743
3744
3745
3746
3747
3748
3748
3749
3749
3750
3751
3752
3753
3754
3755
3756
3757
3758
3759
3759
3760
3761
3762
3763
3764
3765
3766
3767
3768
3769
3769
3770
3771
3772
3773
3774
3775
3776
3777
3778
3778
3779
3779
3780
3781
3782
3783
3784
3785
3786
3787
3788
3788
3789
3789
3790
3791
3792
3793
3794
3795
3796
3797
3797
3798
3798
3799
3799
3800
3801
3802
3803
3804
3805
3806
3807
3808
3809
3809
3810
3811
3812
3813
3814
3815
3816
3817
3818
3819
3819
3820
3821
3822
3823
3824
3825
3826
3827
3828
3828
3829
3829
3830
3831
3832
3833
3834
3835
3836
3837
3838
3838
3839
3839
3840
3841
3842
3843
3844
3845
3846
3847
3848
3848
3849
3849
3850
3851
3852
3853
3854
3855
3856
3857
3858
3859
3859
3860
3861
3862
3863
3864
3865
3866
3867
3868
3869
3869
3870
3871
3872
3873
3874
3875
3876
3877
3878
3878
3879
3879
3880
3881
3882
3883
3884
3885
3886
3887
3888
3888
3889
3889
3890
3891
3892
3893
3894
3895
3896
3897
3897
3898
3898
3899
3899
3900
3901
3902
3903
3904
3905
3906
3907
3908
3909
3909
3910
3911
3912
3913
3914
3915
3916
3917
3918
3919
3919
3920
3921
3922
3923
3924
3925
3926
3927
3928
3928
3929
3929
3930
3931
3932
3933
3934
3935
3936
3937
3938
3938
3939
3939
3940
3941
3942
3943
3944
3945
3946
3947
3948
3948
3949
3949
3950
3951
3952
3953
3954
3955
3956
3957
3958
3959
3959
3960
3961
3962
3963
3964
3965
3966
3967
3968
3969
3969
3970
3971
3972
3973
3974
3975
3976
3977
3978
3978
3979
3979
3980
3981
3982
3983
3984
3985
3986
3987
3988
3988
3989
3989
3990
3991
3992
3993
3994
3995
3996
3997
3997
3998
3998
3999
3999
4000
4001
4002
4003
4004
4005
4006
4007
4008
4009
4009
4010
4011
4012
4013
4014
4015
4016
4017
4018
4019
4019
4020
4021
4022
4023
4024
4025
4026
4027
4028
4028
4029
4029
4030
4031
4032
4033
4034
4035
4036
4037
4038
4038
4039
4039
4040
4041
4042
4043
4044
4045
4046
4047
4048
4048
4049
4049
4050
4051
4052
4053
4054
4055
4056
4057
4058
4059
4059
4060
4061
4062
4063
4064
4065
4066
4067
4068
4069
4069
4070
4071
4072
4073
4074
4075
4076
4077
4078
4078
4079
4079
4080
4081
4082
4083
4084
4085
4086
4087
4088
4088
4089
4089
4090
4091
4092
4093
4094
4095
4096
4097
4097
4098
4098
4099
4099
4100
4101
4102
4103
4104
4105
4106
4107
4108
4109
4109
4110
4111
4112
4113
4114
4115
4116
4117
4118
4119
4119
4120
4121
4122
4123
4124
4125
4126
4127
4128
4128
4129
4129
4130
4131
4132
4133
4134
4135
4136
4137
4138
4138
4139
4139
4140
4141
4142
4143
4144
4145
4146
4147
4148
4148
4149
4149
4150
4151
4152
4153
4154
4155
4156
4157
4158
4159
4159
4160
4161
4162
4163
4164
4165
4166
4167
4168
4169
4169
4170
4171
4172
4173
4174
4175
4176
4177
4178
4178
4179
4179
4180
4181
4182
4183
4184
4185
4186
4187
4188
4188
4189
4189
4190
4191
4192
4193
4194
4195
4196
4197
4197
4198
4198
4199
4199
4200
4201
4202
4203
4204
4205
4206
4207
4208
4209
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
4238
4239
4239
4240
4241
4242
4243
4244
4245
4246
4247
4248
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
4278
4279
4279
4280
4281
4282
4283
4284
4285
4286
4287
4288
4288
4289
4289
4290
4291
4292
4293
4294
4295
4296
4297
4297
4298
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
4338
4339
4339
4340
4341
4342
4343
4344
4345
4346
4347
4348
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
4378
4379
4379
4380
4381
4382
4383
4384
4385
4386
4387
4388
4388
4389
4389
4390
4391
4392
4393
4394
4395
4396
4397
4398
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
4438
4439
4439
4440
4441
4442
4443
4444
4445
4446
4447
4448
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
4478
4479
4479
4480
4481
4482
4483
4484
4485
4486
4487
4488
4488
4489
4489
4490
4491
4492
4493
4494
4495
4496
4497
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
4538
4539
4539
4540
4541
454
```

```

^^M 2719 \def^^M{% TeX code EOL
2720 @newline{true} to \refstep the counter in proper place.
2721 @dsdirg{true} to handle the DocStrip directives.
2722 \global\gmd@closingspacewd=\z@% we don't wish to eat a closing space
 after a codeline, because there isn't any and a negative rigid \hskip
 added to \parfillskip would produce a blank line.
2726 \ifhmode\par\@codeskipputgfalse\else%
2727 \if@codeskipput%
2728 \else\addvspace{\stanzaskip}\@codeskipputgtrue%
2729 \fi% if we've just met a blank (code) line, we insert a \stanzaskip glue.
2732 \fi%
2733 \prevhmodegfalse% we want to know later that now we are in the vmode.
2736 \@ifnextchar{\gmd@texcodespace}{%
2737 \@dsdirgfalse\gmd@dolspaces}{\gmd@charbychar}%
2738 }% end of ^^M's definition.
2740 \let\gmd@texcodeEOL=^^M for further checks inside \gmd@charbychar.
2741 \raggedright\leftskip=\CodeIndent%
2742 \if@aftercode%
2743 \gmd@nocodeskip1{iaC}%
2744 \else%
2745 \if@afternarr%
2747 \if@codeskipput\else%
2748 \gmd@codeskip1\@aftercodegfalse%
2749 \fi%
2750 \else\gmd@nocodeskip1{naN}%
2751 \fi%
2752 \fi% if now we are switching from the narration into the code, we insert
 a proper vertical space.
2755 \@aftercodegtrue\@afternarrgfalse%
2757 \ifdim\gmd@ldspaceswd>\z@% and here the leading spaces.
2758 \leavevmode\@dsdirgfalse%
2759 \if@newline\gmd@grefstep{codelinenum}\@newline{false}%
2760 \fi%
2761 \printlinenumber% if we don't want the lines to be numbered, the respec-
 tive option \lets this cs to \relax.
2763 \hyperlabel@line%
2765 \mark@envir% index and/or marginize an environment if there is some to
 be done so, see line 4803.
2767 \hskip\gmd@ldspaceswd%
2768 \advance\hangindentby\gmd@ldspaceswd%
2769 \xdef\settexcodehangi{%
2770 \nx\hangindent=\the\hangindent% and also set the hanging indent
 setting for the same line comment case. BTW., this % or rather lack of
 it costed me five hours of debugging and rewriting. Active lineends
 require extreme caution.
2775 \nx\hangafter=1\space}%
2776 \else%
2777 \glet\settexcodehangi=\@settexcodehangi%
 % \printlinenumber here produced line numbers for blank lines
 which is what we don't want.
2780 \fi% of \ifdim
2781 \gmd@ldspaceswd=\z@%
2782 \prevhmodegfalse% we have done \par so we are not in the hmode.

```

```

2784  \@aftercodegtrue% we want to know later that now we are typesetting a code-
2785  line.
2786  \if@ilgroup\aftergroup\egroup\@ilgroupfalse\fi% when we are in the
2787  inline comment group (for ragged right or justified), we want to close it.
2788  But if we did it here, we would close the verbatim group for the code. But
2789  we set the switch false not to repeat \aftergroup\egroup.
2790  \gmd@charbychar% we'll eat the code char by char to scan all the macros and
2791  thus to deal properly with the case \% in which the % will be scanned and
2792  won't launch closing of the verbatim group.
2793  }% of \gmd@typesettexcode.
2794  }% of \foone\obeylines.

```

Now let's deal with the leading spaces once forever. We wish not to typeset $_s$ 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.)

```

2810 \newlength\gmd@spacewd% to store the width of a (leading) \_.
2811 \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 2821 \let\gmd@texcodespace=\breakablevisspace
```

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

```

\gmd@dolspaces 2824 \def\gmd@dolspaces{%
2825   \ifx\gmd@texcodespace\@let@token
2826     \@dsdirgfalse
2827     \afterfi{\settowidth{\gmd@spacewd}{\visible}{\gmd@eatlspace}}%
2828     \gmd@ldspaceswd=\z@
2829   \else\afterfi{%
2830     \aboutthis{%
2831       \if@afternarr\if@aftercode
2832         \ifilrr\bgroup\gmd@setilrr\fi
2833         \fi\fi
2834         \par% possibly after narration
2835       \if@afternarr\if@aftercode
2836         \ifilrr\egroup\fi
2837         \fi\fi
2838       \gmd@typesettexcode}%
2839   \else
2840   }
2841 }
```

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

```

\gmd@eatlspace 2848 \def\gmd@eatlspace#1{%
2849   \ifx\gmd@texcodespace#1%
2850     \advance\gmd@ldspaceswd\by\gmd@spacewd% we don't \advance
2851     it \globally because the current group may be closed iff we meet % and
2852     then we'll won't indent the line anyway.
2853     \afteriffifi\gmd@eatlspace
2854   \else
2855     \if\code@delim\@nx#1%
2856       \gmd@ldspaceswd=\z@
2857       \afterfifi{\gmd@continuenarration#1}%
2858     \else\afterfifi{\gmd@typesettexcode#1}%
2859   
```

```

2860      \fi
2861  \fi}%

```

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

```

2866 \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 2874 \newcommand*\gmd@charbychar[1]{%
2875   \ifhmode\prevhmodegtrue
2876   \else\prevhmodegfalse
2877   \fi
2878   \if\code@delim\@nx#1%
2879     \def\next{%
2880       \gmd@percenthack% occurs when next a \hskip4.875pt is to be put
2881       \gmd@percenthack% to typeset % if a comment continues the codeline.
2882     \endgroup%
2883     \gmd@checkifEOLmixd}% to see if next is ^^M and then do \par.
2884   \else% i.e., we've not met the code delimiter
2885     \ifx\relax#1\def\next{%
2886       \endgroup}% special case of end of file thanks to \everyeof.
2887     \else
2888       \if\code@escape@char\@nx#1%
2889         \gmd@counttheline#1\scan@macro}%
2890       \else
2891         \def\next{%
2892           \gmd@EOLorcharbychar#1}%
2893         \fi
2894       \fi\next}
2895 \def\debug@special#1{%
2896   \ifhmode\special{color\push\gray\o.\#1}%
2897   \else\special{color\push\gray\o.\#1000}\fi}

```

One more inner macro because ^^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 2916 \def\gmd@EOLorcharbychar#1{%
2917   \ifx\gmd@texcodeEOL#1%
2918     \if@newline
2919       \gmd@newlinefalse
2920     \fi
2921     \afterfi{#1}% here we print #1.
2922   \else% i.e., #1 is not a (very active) line end,
2923     \afterfi
2924   \fi
2925   \gmd@newlinefalse
2926   \gmd@newlinefalse
2927   \gmd@newlinefalse
2928   \gmd@newlinefalse

```

2929 \gmd@counttheline#1\gmd@charbychar} or here we print #1. Here we would
 also possibly mark an environment but there's no need of it because declaring
 an environment to be marked requires a bit of commentary and here we are
 after a code $\wedge\wedge M$ with no commentary.

```
2934 \fi}
2935 \def\gmd@counttheline{%
2936   \ifvmode
2937     \if@newline
2938       \leavevmode
2939       \gmd@grefstep{codelinenumber}\@newlinefalse
2940       \hyperlabel@line
2941     \fi
2942     \printlinenumber
2943     \mark@envir
2944   \else% not vmode
2945     \if@newline
2946       \gmd@grefstep{codelinenumber}\@newlinefalse
2947       \hyperlabel@line
2948     \fi
2949   \fi
2950 }
```

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

```
\gmd@percenthack
2959 \def\gmd@percenthack{%
2960   \ifprevhmode\code@delim\aftergroup~% We add a space after %, because
2961   I think it looks better. It's done \aftergroup to make the spaces possible
2962   after the % not to be typeset.
2963   \else\aftergroup\gmd@narrcheckifds@ne% remember that \gmd@precent-
2964   hack is only called when we've the code delimiter and soon we'll close the
2965   verbatim group and right after \endgroup there waits \gmd@checkifEOLmixed.
2966   \fi}
```

```
\gmd@narrcheckifds@ne
2972 \def\gmd@narrcheckifds@ne#1{%
2973   \odsdirgfalse\@ifnextchar<{%
2974     \xa\gmd@docstripdirective\@gobble}{#1}}
```

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

```
\gmd@checkifEOL
2980 \def\gmd@checkifEOL{%
2981   \gmd@cpnarrline
2982   \everypar=\xa{\xa\@codetonarrskip% we add the macro that'll insert a ver-
2983   tical space if we leave the code and enter the narration.
2984   \the\gmd@preverypar}%
2985   \@ifnextchar{\gmd@textEOL}{%
2986     \odsdirgfalse
2987     \par\ignorespaces}%
2988   \gmd@narrcheckifds}}%
```

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

```
\gmd@narrcheckifds
2993 \def\gmd@narrcheckifds{%
2994   \odsdirgfalse\@ifnextchar<{%
2995     \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 3001 \def\gmd@checkifEOLmixd{%
3002   \gmd@cpnarrline
3003   \everypar=\@xa{\@codetonarrskip\the\gmd@preverypar}%
3004   \@afternarrgfalse\@aftercodegtrue
3005   \ifhmode\codeskipputgfalse\fi
3006   \@ifnextchar{\gmd@textEOL}{%
3007     {\raggedright\gmd@endpe\par}% without \raggedright this \par would
3008     be justified which is not appropriate for a long codeline that should be
3009     broken, e.g., 3003.
3010   \prevhmodegfalse
3011   \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 3023   \def\par{%
3024     the narration \par.
3025     \ifhmode% (I added this \ifhmode as a result of a heavy debug.)
3026       \if@afternarr\if@aftercode
3027         \unless\if@ilgroup\bgroup\@ilgrouptrue\fi
3028         \ifilrr\gmd@setilrr\fi
3029       \fi\fi
3030       \@@par
3031       \if@afternarr
3032         \if@aftercode
3033           \if@ilgroup\egroup\fi% if we are both after code and after narration
3034             it means we are after an inline comment. Then we probably end
3035             a group opened in line 3076
3036           \if@codeskipput\else\gmd@codeskip2\@aftercodegfalse%
3037             \fi
3038             \else\gmd@nocodeskip2{naC}%
3039             \fi
3040             \else\gmd@nocodeskip2{naN}%
3041             \fi
3042             \prevhmodegfalse\gmd@endpe% when taken out of \ifhmode, this line
3043               caused some codeline numbers were typeset with \leftskip = 0.
3044             \everypar=\@xa{%
3045               \@xa\@codetonarrskip\the\gmd@preverypar}%
3046             \let\par\@@par%
3047             \fi}%
3048             of \par.
3049   \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 3057 \def\gmd@endpe{%
3058   \ifprevhmode
3059     \settexcodehangi%ndent
3060     \leftskip=\CodeIndent
3061   \else
3062     \leftskip=\TextIndent
3063     \hangindent=\z@

```

```

3065   \everypar=\@xa{%
3066     \@xa\@codetonarrskip\the\gmd@preverypar}%
3068   \fi}

Now a special treatment for an inline comment:

\ifilrr 3072 \newif\ifilrr
\ilrr 3074 \def\ilrr{%
3075   \if@aftercode
3076     \unless\if@ilgroup\bgroup\@ilgrouptrue\fi% If we are 'aftercode', then
          we are in an inline comment. Then we open a group to be able to declare
          e.g. \raggedright for that comment only. This group is closed in line
          3033 or 2786.
3081   \ilrrtrue
3082   \fi}
\if@ilgroup 3084 \newif\if@ilgroup
\gmd@setilrr 3086 \def\gmd@setilrr{\rightskipoptplus\textwidth}
\ilju 3088 \def\ilju{%
  when inline comments are ragged right in general but we want just
  this one to be justified.
  \if@aftercode
    \unless\if@ilgroup\bgroup\@ilgrouptrue\fi
    \ilrrfalse
  \fi}
\verbcodecorr 3095 \def\verbcodecorr{%
  a correction of vertical spaces between a verbatim and
  code. We put also a \par to allow parindent in the next commentary.
  \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.

```

3107 \if@uresetlinecount% with uresetlinecount option...
3108   \relax\gmd@resetlinecount% ... we turn resetting the counter by \DocIn-
      % put off...
\resetlinecountwith 3110 \newcommand*\resetlinecountwith[1]{%
3111   \newcounter{codelinenum}[#1]}% ... and provide a new declaration of the
      counter.
3113 \else% With the option turned off...
\DocInputsCount 3114 \newcounter{DocInputsCount}%
\codelinenum 3115 \newcounter{codelinenum}[DocInputsCount]%
            ... we declare the \DocInputs' number counter and the codeline counter to be reset with stepping of it.
\gmd@resetlinecount 3121 \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.
3125 \fi

```

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

```

\printlinenumber 3129 \newcommand*\printlinenumber{%
3130   \leavevmode\llap{\rlap{\LineNumFont$\phantom{999}$$\llap{%
      \thecodelinenum}}}}
3131   \hspace{-\leftskip}}
\LineNumFont 3133 \def\LineNumFont{\normalfont\tiny}

```

```

3135 \if@linesnotnum\relax\printlinenumber\fi
\hyperlabel@line 3137 \newcommand*\hyperlabel@line{%
3138   \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.
3141   \else
3142     \raisebox{2ex}[1ex][\z@]{\gmpageref[clnum.%%
3143       \HLPrefix\arabic{codelinenum}]{}}%
3144   \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 3154 \newcommand*\gmd@codeskip[1]{%
3155   \@@par\addvspace\CodeTopsep
3156   \codeskipputtrue\@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 3167 \newgif\if@codeskipput
A switch to control \nostanzas:
3170 \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 3175 \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 3180 \if1\if1
3181   \renewcommand*\gmd@codeskip[1]{%
3182     \hbox{\rule{1cm}{3pt}\#1!!!}}
\gmd@nocodeskip 3183 \renewcommand*\gmd@nocodeskip[2]{%
3184   \hbox{\rule{1cm}{0.5pt}\#1:\#2}}
3185 \fi

```

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

```
\if@aftercode 3191 \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 3196 \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 3201 \def\codetonarrskip{%
3202   \if@codeskipput\else

```

```

3203   \if@afternarr\gmd@nocodeskip4{iaN}\else
3204     \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.

```

3207   {\setboxo=\lastbox}%

```

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

```

3209   \gmd@codeskip4%
3211   \else\gmd@nocodeskip4{naC}%
3212   \fi
3213   \fi
3214   \fi
3215   \leftskip\TextIndent% this line is a patch against a bug-or-feature that in cer-
      tain cases the narration \leftskip is left equal the code leftskip. (It happens
      when there're subsequent code lines after an inline comment not ended with
      an explicit \par.) Before vo.99n it was just after line 3209.
3220   \@aftercodegfalse\@nostanzagtrue
3222 }

```

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
3234 \long\def\@trimandstore#1\@trimandstore{%
3235   \def\@trimandstore@hash{#1}%
3236   \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.
3237   \gmd@preeverypar={}%
3238   \else
3239     \afterfi{\@xa\@trimandstore@ne\the\everypar\@trimandstore}%
3240   \fi}

@trimandstore@ne
3241 \long\def\@trimandstore@ne#1#2\@trimandstore{%
3242   \def\trimmed@everypar{#2}%
3243   \ifx\@codetonarrskip#1%
3244     \gmd@preeverypar=\@xa{\trimmed@everypar}%
3245   \else
3246     \gmd@preeverypar=\@xa{\the\everypar}%
3247   \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:

```

3266 \catcode`^\^B=\active% note we re\catcode <char2> globally, for the entire doc-
ument.
3268 \foone{\obeylines}{%
^\^B
3269   {\def\QueerCharTwo{%
3270     \protected\def^\^B##1^\^M{%
3271       \ifhmode\unskip\space\ignorespaces\fi}}% It shouldn't be \ not
3272       to drive TEX into hmode.
3273   }
3274 }
3275 \QueerCharTwo
3276 \AtBeginInput{\@ifEOLactive{\catcode`^\^B\active}{}{\QueerCharTwo}}% We
3277   repeat redefinition of <char2> at begin of the documenting input, because
3278   doc.dtx suggests that some packages (namely inputenc) may re\catcode
3279   such unusual characters.

```

As you see the `^\^B` 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 `countallines` 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.

```

3300 \catcode`^\^A=\active% note we re\catcode <char1> globally, for the entire doc-
ument.
3301 \foone{\obeylines}{%
^\^A
3302   {\def\QueerCharOne{%
3303     {\def^\^A{%
3304       \bgroup\let\do\@makeother\dospecials\gmd@gobbleuntilM}}%
3305     \def\gmd@gobbleuntilM#1^\^M{\egroup\ignorespaces^\^M}}%
3306   }
3307 }
3308 \QueerCharOne
3309 \AtBeginInput{\@ifEOLactive{\catcode`^\^A\active}{\QueerCharOne}}% see
3310   note after line 3278.

```

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 linebreaking of the command arguments. Another—making possible executing some code lines during the documentation pass.

```

\StraightEOL 3328 \def\StraightEOL{%
3329   \catcode`^\^M=5
3330   \catcode`^\^A=14
3331   \catcode`^\^B=14
3332   \def`^\^M{\_}
3333
3334 \foone{\obeylines}{%
\QueerEOL
3335   {\def\QueerEOL{%
3336     \catcode`^\^M=\active%
3337     \let^\^M\gmd@textEOL%
3338     \catcode`^\^A=\active%
3339   }
3340 }
3341
3342
3343
3344

```

```

3345   \catcode`^=active% I only re\catcode <char1> and <char2> hoping no
          one but me is that perverse to make them \active and (re)define. (Let
          me know if I'm wrong at this point.)
3348   \let`^M=\gmd@bslashEOL}%
3361 }

```

To make $\wedge M$ behave more like a ‘normal’ lineend 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 \\advance the \\parfillskip :

```

\gmd@parfixclosingspace
3375 \def\gmd@parfixclosingspace{{%
3376   \advance\parfillskip by-\gmd@closingspacewd
3377   \if@aftercode\ifilrr\gmd@setilrr\fi\fi
3378   \par}%
3379   \if@ilgroup\aftergroup\egroup\@ilgroupfalse\fi% we are in the verba-
          tim group so we close the inline comment group after it if the closing is not
          yet set.
3382 }

```

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

```

\gmd@closingspacewd
\gmd@setclosingspacewd
3386 \newskip\gmd@closingspacewd
3387 \newcommand*\gmd@setclosingspacewd{%
3388   \global\gmd@closingspacewd=\fontdimen2\font%
3389   plus\fontdimen3\font minus\fontdimen4\font\relax}

```

See also line 2722 to see what we do in the codeline case when no closing space is added.

And one more detail:

```

3395 \foone\obeylines{%
3396   \if_1_1%
3397     \protected\def\gmd@bslashEOL{\_\@xa\ignorespaces^M}%
3398   }% of \foone. Note we interlace here \if with a group.
3399 \else%
3400   \protected\def\gmd@bslashEOL{%
3401     \ifhmode\unskip\fi\_\ignorespaces}%
3403 \fi

```

The \\QueerEOL declaration will \\let it to $\wedge M$ to make $\wedge M$ behave properly. If this definition was omitted, $\wedge M$ 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 $\wedge M$ 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.

Adjustment of verbatim and \\verb

To make $\text{\\verbatim}(*)$ typeset its contents with the TeX code’s indentation:

```

\@verbatim
3426 \gaddtomacro\@verbatim{\leftskip=\CodeIndent}

```

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

```

\check@percent
3430 \AtBeginInput{\long\def\check@percent#1{%

```

```

3431   \gmd@cpnarrline% to count the verbatim lines and possibly print their num-
      bers. This macro is used only by the verbatim end of line.
3433   \@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 3436 \def\AddtoPrivateOthers#1{%
3437   \@xa\def\@xa\doprivateothers\@xa{%
3438     \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.

```

3449 \begingroup
3450 \obeylines%
\gm@verb@eol 3451 \AtBeginInput{\def\gm@verb@eol{\obeylines%
3452   \def^{\M{\verb@egroup}\@latex@error{%
3453     \Onx\verb@ended@by@end@of@line}}%
3454   \@ifEOActive{^{\M{\@ehc}}}}}}%
3455 \endgroup
```

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 escapechar) 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 3478 \edef\actualchar{\string\_@}
\quotechar 3479 \edef\quotechar{\string\_"}
\encapchar 3480 \edef\encapchar{\xiiclub}
\levelchar 3481 \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 3493 \def\CodeEscapeChar#1{%
3494   \begingroup
3495   \escapechar\m@ne
\code@escape@char 3496 \xdef\code@escape@char{\string#1}%
3497   \endgroup}
```

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

```
3503 \CodeEscapeChar\\"
```

As mentioned in doc, someone may have some chars $\text{_}\text{_}$ ed.

```
3506 \@ifndef{MakePrivateLetters}{%
3507   \def\MakePrivateLetters{\makeatletter\catcode`*\text{\_}}{}}
```

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 $\text{_}\text{_}$.

```
\MakePrivateOthers 3515 \def\MakePrivateOthers{\let\do=\@makeother\doprivateothers}
```

We use this macro to re`\catcode` the space for marking the environments’ names and the caret for marking chars such as $\text{\~}\text{\^}M$, see line 4967. So let’s define the list:

```
\doprivateothers 3519 \def\doprivatethers{\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.

```
\scan@macro 3532 \def\scan@macro#1{%
  we are sure to scan at least one token and therefore we define
  this macro as one-parameter.
```

Unlike in doc, here we have the escape char $\text{_}\text{_}$ so we may just have it printed during main scan char by char, i.e., in the lines 2925 and 2929.

So, we step the checksum counter first,

```
3538 \step@checksum% (see line 6161 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 $\text{_}\text{_}$ surely would be one of those perversities. Therefore I feel safe to take the character a as a benchmark letter.

```
3547 \ifcat_a\@nx#1%
3548   \quote@char#1%
3549   \xdef\macro@iname{\gmd@maybequote#1}% global for symmetry with line
3550     3567.
3551   \xdef\macro@pname{\string#1}% we’ll print entire name of the macro later.
```

We `\string` it here and in the lines 3571 and 3583 to be sure it is whole $\text{_}\text{_}$ for easy testing for special indexentry formats, see line 4473 etc. Here we are sure the result of `\string` is $\text{_}\text{_}$ since its argument is $\text{_}\text{_}$.

```
3558   \afterfi{\@ifnextcat{a}{\gmd@finishifstar#1}{%
3559     \finish@macroscan}}%
3559   \else% #1 is not a letter, so we have just scanned a one-char cs.
```

Another reasonable `\catcode`s assumption seems to be that the digits are $\text{_}\text{_}$. Then we don’t have to type `(%)``\expandafter\@gobble\string\aa`. We do the `\uccode` trick to be sure that the char we write as the macro’s name is $\text{_}\text{_}$.

```
3566   {\uccode`9=#1%
3567     \uppercase{\xdef\macro@iname{9}}%
3568   }%
3569   \quote@char#1%
3570   \xdef\macro@iname{\gmd@maybequote\macro@iname}%
```

```

3571   \xdef\macro@pname{\xiistring#1}%
3572     \afterfi\finish@macroscan
3573   \fi}

```

The `\xiistring` macro, provided by `gutils`, is used instead of original `\string` because we wish to get \ll_1 ('other' space).

Now, let's explain some details, i.e., let's define them. We call the following macro having known #1 to be \ll_1 .

```

\continue@macroscan 3580 \def\continue@macroscan#1{%
3581   \quote@char#1%
3582   \xdef\macro@iname{\macro@iname\gmd@maybequote#1}%
3583   \xdef\macro@pname{\macro@pname\string#1}%
3584   we know#1 to be  $\ll_1$ , so we
3585   don't need \xiistring.
3586   \@ifnextcat{a}{\gmd@finishifstar#1}{\finish@macroscan}%
3587 }

```

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 3596 \def\gmd@finishifstar#1{%
3597   \if*\@nx#1\afterfi\finish@macroscan% note we protect #1 against expansion.
3598   In gmdoc verbatim scopes some chars are active (e.g. \).
3599   \else\afterfi\continue@macroscan
3600   \fi}

```

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

```

\quote@char 3605 \def\quote@char#1{{\uccode`9=#1% at first I took digit 1 for this \uccodeing
3606   but then #1 meant #(#1) in \uppercase's argument, of course.
3607   \uppercase{%
3608     \gmd@ifinmeaning\of\indexcontrols
3609     {\glet\gmd@maybequote\quotechar}%
3610     {\g@emptyify\gmd@maybequote}%
3611   }%
3612 }%
3613 }

```

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 3624 \AtBeginDocument{\xdef\indexcontrols{%
3625   \bslash\levelchar\encapchar\actualchar\quotechar}}
\gmd@ifinmeaning 3627 \long\def\gmd@ifinmeaning#1\of#2#3#4% explained in the text paragraph
3628   below.
\gmd@in@@ 3631 \long\def\gmd@in@@#1##2\gmd@in@@{%
3632   \ifx^A##2^A\afterfi{#4}%
3633   \else\afterfi{#3}%
3634   \fi}%
3635   \xa\gmd@in@@#1\gmd@in@@}%

```

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.

The meaning of the \gmd@ifinmeaning parameters is as follows:

- #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.

In \quote@char the second argument for \gmd@ifinmeaning is \indexcontrols defined as the (expanded and 'other') sequence of the MakeIndex controls. \gmd@ifinmeaning defines its inner macro \gmd@in@@ to take two parameters separated by the first and the second \gmd@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 \gmd@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 secong 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 \gmd@ifinmeaning is used also in testing whether a detector is already present in the carrier in the mechanism of automatic detection of definitions (line 3831).

```
\ifgmd@glosscs 3688 \newif\ifgmd@glosscs% we use this switch to keep the information whether a his-
tory entry is a cs or not.
```

```
\finish@macroscan 3692 \newcommand*\finish@macroscan{%
```

First we check if the current cs is not just being defined. The switch may be set true in line 3728

```
3695 \ifgmd@adef@cshook% if so, we throw it into marginpar and index as a def en-
try...
3697 \@ifundefined{gmd/iexcl/\macro@pname}{% ... if it's not excluded from in-
dexing.
3699   \xa\Code@MarginizeMacro\x{\macro@pname}%
3700   \xa\defentryze\x{\macro@pname}{1}}{}% here we declare the kind
      of index entry and define \last@defmark used by \changes
3702   \global\gmd@adef@cshookfalse% we falsify the hook that was set true just
      for this cs.
3704 \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 countercrank way because we wish to use more general indexing macro.

```

3709  \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 $:
3710  \def\im@firstpar{[$%
3711    ]}%
3712  \else\def\im@firstpar{}\fi
3713  \cxa\index@macro\im@firstpar\macro@iname\macro@pname
3714  \maybe@marginpar\macro@pname
3715  \if\xiispace\macro@pname\relax\gmd@texcodespace
3716  \else\macro@pname
3717  \fi
3718  \let\next\gmd@charbychar
3719  \gmd@detectors% for automatic detection of definitions. Defined and ex-
3720  plained in the next section. It redefines \next if detects a definition com-
3721  mand and thus sets the switch of line 3692 true.
3722  \next
3723  }

```

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
3741 \def\maybe@marginpar#1{%
3742   \@ifundefined{gmd/2marpar/#1}{}{%
3743     \cxa\Text@Marginize\cxa{\bslash#1}\expandafters
3744       because the \Text@Marginize command applies \string to its argument.
3745       \% \macro@pname, which will be the only possible argument to \maybe-
3746       @marginpar, contains the macro's name without the escapechar so we
3747       added it here.
3748     \cxa\g@relaxen\csname\gmd/2marpar/#1\endcsname% we reset the switch.
3749   }%
3750 }

```

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@(...)`) the family is mandatory.

Let's make a version of `\@ifstar` that would work with `*11`. It's analogous to `\@ifstar`.

```
3791 \foone{\catcode`*\!=\!_}
3792 {\def\@ifstar{\ifnextchar*{\firstoftwo{#1}}{}}
```

`\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 3799 \outer\def\DeclareDefining{\begingroup
3800   \MakePrivateLetters
3801   \@ifstar{
3802     {\gdef\gmd@adef@defaulttype{text}\ Declare@Dfng}%
3803     {\gdef\gmd@adef@defaulttype{cs}\ Declare@Dfng}%
3804 }
```

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

```
\Declare@Dfng 3808 \newcommand*\Declare@Dfng[2][]{%
3809   \endgroup
3810   \Declare@Dfng@inner{#1}{#2}%
3811   \ifgmd@adef@star% this switch may be set false in first \Declare@Dfng@inner
      (it's the star key).
3813   \Declare@Dfng@inner{#1}{#2*}% The catcode of * doesn't matter since it's
      in \csname...\endcsname everywhere.
3817   \fi}
```

```
\Declare@Dfng@inner 3820 \def\Declare@Dfng@inner#1#2{%
3821   \edef\gmd@resa{%
3822     \onx\setkeys[gmd]{adef}{type=\gmd@adef@defaulttype}}%
3823   \gmd@resa
3824   {\escapechar\m@ne
3825     \xdef\gmd@adef@currdef{\string#2}%
3827   }%
3828   \gmd@adef@setkeysdefault
3829   \setkeys[gmd]{adef}{#1}%
3830   \xa\gmd@ifinmeaning
3831   \csname\gmd@detect@\gmd@adef@currdef\endcsname
3833   \of\gmd@detectors{}{%
3834     \xa\gaddtomacro\x\gmd@detectors\@xa{%
3835       \csname\gmd@detect@\gmd@adef@currdef\endcsname}%
      we add a cs
      \gmd@detect@{def name} (a detector) to the meaning of the detec-
      tors' carrier. And we define it to detect the #2 command.
3839   \xa\xdef\csname\gmd@detectname@\gmd@adef@currdef\endcsname{%
```

```

3840   \gmd@adef@currdef}%
3841   \edef\gmu@tempa{%
3842     \global\@nx\@namedef{gmd@detect@\gmd@adef@currdef}{%
3843       \@nx\ifx
3844         \xa\@nx\csname\gmd@detectname@\gmd@adef@currdef%
3845           \endcsname
3846         \@nx\macro@pname
3847         \@nx\n@melet{next}{gmd@adef@\gmd@adef@TYPE}%
3848         \@nx\n@melet{gmd@adef@currdef}{gmd@detectname@%
3849           \gmd@adef@currdef}%
3850         \@nx\fi}%
3851   \gmu@tempa
3852 } \SMglobal\StoreMacro*{gmd@detect@\gmd@adef@currdef}%
3853 we store the cs
3854 to allow its temporary discarding later.
3855 \def\gmd@adef@setkeysdefault{%
3856   \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 4029.

star 3862 \define@boolkey[gmd]{adef}{star}[true] {}

The prefix@<command> keyvalue 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 4282 and therefore \newcounter{foo} occurring in the code will index both foo
and \c@foo (as definition entries).

prefix 3871 \define@key[gmd]{adef}{prefix}[]{%
3872   \edef\gmd@resa{%
3873     \def\@xa\@nx\csname\gmd@adef@prefix@\gmd@adef@currdef%
3874       \endcsname{%
3875         #1}}%
3876   \gmd@resa}

\gmd@KVprefdefault 3878 \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 3886 \define@key[gmd]{adef}{KVpref}[\gmd@KVprefdefault]{%
3887   \edef\gmd@resa{#1}%
3888   \ifx\gmd@resa\gmd@KVprefdefault
3889   \else
3890     \@namedef{gmd@adef@KVprefixset@\gmd@adef@currdef}{#1}%
3891     \gmd@adef@setKV% whenever the KVprefix is set (not default), the declared
3892       command is assumed to be keyvalish.
3893   \fi
3894   \edef\gmd@resa{#1}%
3895   because \gmd@adef@setKV redefined it.
3896   \edef\gmd@resa{%
3897     \def\@xa\@nx\csname\gmd@adef@KVpref@\gmd@adef@currdef%
3898       \endcsname{%
3899         \ifx\gmd@resa\empty
3900           \else#1\@fi}%
3901         as in xkeyval, if the kv prefix is not empty, we add @ to it.
3902 }


```

```
3900 \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 3907 \define@key[gmd]{adef}{KVfam} [] {%
3908   \edef\gmd@resa{\#1}%
3909   \namedef{gmd@adef@KVfamset@\gmd@adef@currdef}{\#1}%
3910   \edef\gmd@resa{%
3911     \def\@xa\@nx\csname\gmd@adef@KVfam@\gmd@adef@currdef%
3912       \endcsname{%
3913         \ifx\gmd@resa\empty
3914           \else\#1\fi}}%
3915   \gmd@resa
3916   \gmd@adef@setKV}%
3917 whenever the KVfamily is set, the declared command is assumed to be keyvalish.
```

```
type 3919 \define@choicekey[gmd]{adef}{type}
3920   [\gmd@adef@typevals\gmd@adef@typenr]
3921   {%
3922     the list of possible types of defining commands
3923     def,
3924     newcommand,
3925     cs, % equivalent to the two above, covers all the cases of defining a cs, including
3926       the PLAIN TEX \new... and LATEX \newlength.
3927     newenvironment,
3928     text, % equivalent to the one above, covers all the commands defining its first
3929       mandatory argument that should be text, \DeclareOption e.g.
3930     define@key, % special case of more arguments important; covers the xkeyval
3931       defining commands.
3932     dk, % a shorthand for the one above.
3933     DeclareOptionX, % another case of special arguments configuration, covers the
3934       xkeyval homonym.
3935     dox, % a shorthand for the one above.
3936     kvo% one of option defining commands of the kvoptions package by Heiko
3937       Oberdiek (a package available on CTAN in the oberdiek bundle).
3938   }
3939   {%
3940     In fact we collapse all the types just to four so far:
3941     \ifcase\gmd@adef@typenr% if def
3942       \gmd@adef@settype{cs}{o}%
3943     \or% when newcommand
3944       \gmd@adef@settype{cs}{o}%
3945     \or% when cs
3946       \gmd@adef@settype{cs}{o}%
3947     \or% when newenvironment
3948       \gmd@adef@settype{text}{o}%
3949     \or% when text
3950       \gmd@adef@settype{text}{o}%
3951     \or% when define@key
3952       \gmd@adef@settype{dk}{\#1}%
3953     \or% when dk
3954       \gmd@adef@settype{dk}{\#1}%
3955     \or% when DeclareOptionX
3956       \gmd@adef@settype{dox}{\#1}%
3957     \or% when dox
3958 }
```

```

3959      \gmd@adef@settype{dox}{1}%
3960      \or% when kvo
3961      \gmd@adef@settype{text}{1}% The kvoptions option definitions take first
                                mandatory argument as the option name and they define a keyval key
                                whose macro's name begins with the prefix/family, either default or
                                explicitly declared. The kvoptions prefix/family is supported in gmdoc
                                with [KVpref=, KVfam=<family>].
3967      \fi}
\gmd@adef@settype 3969 \def\gmd@adef@settype#1#2{%
3970   \def\gmd@adef@TYPE{#1}%
3971   \ifnum1=#2% now we define (or not) a quasi-switch that fires for the keyvalish
                definition commands.
3973   \gmd@adef@setKV
3974   \fi}
\gmd@adef@setKV 3976 \def\gmd@adef@setKV{%
3977   \edef\gmd@resa{%
3978     \def\@xa\@nx\csname\gmd@adef@KV@\gmd@adef@currdef\endcsname{%
3979       }%
3980     \gmd@resa}}

```

We initialize the carrier of detectors:

```
3984 \emptify\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 3990 \newif\ifgmd@adef@cshook
\gmd@adef@cs 3992 \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 seargants. In gmdoc code layer scopes the left brace is active so we only add a hook to its meaning (see line 286 in gmverb) and ??nd here we switch it according to the type of detected definition.

```
\gmd@adef@text 4000 \def\gmd@adef@text{\gdef\gmd@lbracecase{1}\gmd@charbychar}
4002 \foone{%
4003   \catcode`\\active
4005   \catcode`\\<\\active}
4006 }%
```

The detector of xkeyval \define@(...)key:

```
\gmd@adef@dk 4008 \def\gmd@adef@dk{%
4009   \let[\gmd@adef@scanKVpref
4010   \catcode`\\active
4012   \gdef\gmd@lbracecase{2}%
4013   \gmd@adef@dfKVpref\gmd@KVprefdefault% We set the default value of
                                              the xkeyval prefix. Each time again because an assignment
                                              in \gmd@adef@dfKVpref is global.
4016   \gmd@adef@checklbracket}
```

The detector of xkeyval \DeclareOptionX:

```
\gmd@adef@dox 4019 \def\gmd@adef@dox{%
4020   \let[\gmd@adef@scanKVpref
4021   \let<\gmd@adef@scanDOXfam
```

```

4022   \catcode`[\active
4024   \catcode`<\active
4025   \gdef\gmd@lbracecase{1}%
4026   \gmd@adef@dfKVpref\gmd@KVprefdefault% We set the default values of the
4027   xkeyval prefix...
4028   \edef\gmd@adef@fam{\gmd@inputname}% ... and family.
4029   \gmd@adef@ofam
4030   \gmd@adef@checkDOXopts}%
4031 }
4032 }
```

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 4037 \def\gmd@adef@checklbracket{%
4038   \@ifnextchar[%
4039   \gmd@adef@scanKVpref\gmd@charbychar}% note that the prefix scanning
4040   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 seargants. Therefore we have to test presence of both of them.

```

\gmd@adef@checkDOXopts 4047 \def\gmd@adef@checkDOXopts{%
4048   \@ifnextchar[ \gmd@adef@scanKVpref%
4049   {\@ifnextchar<\gmd@adef@scanDOXfam\gmd@charbychar}}}

\gmd@adef@scanKVpref 4053 \def\gmd@adef@scanKVpref#1#2}{%
4054   \gmd@adef@dfKVpref{#2}%
4055   [#2]\gmd@charbychar}

\gmd@adef@dfKVpref 4058 \def\gmd@adef@dfKVpref#1{%
4059   \ifnum#1=0\csname\gmd@adef@KVprefixset@\gmd@adef@currdef%
4060   \endcsname
4061   \relax
4062   \else
4063   \edef\gmu@resa{%
4064   \gdef\@xa\@nx
4065   \csname\gmd@adef@KVpref@\gmd@adef@currdef\endcsname{%
4066   \ifx\relax#1\relax
4067   \else#1%
4068   \fi}}%
4069   \gmu@resa
4070   \fi}

\gmd@adef@scanDOXfam 4072 \def\gmd@adef@scanDOXfam{%
4073   \ifnum#1=1\catcode`>\relax
4074   \let\next\gmd@adef@scanfamoth
4075   \else
4076   \ifnum#1=2\catcode`>\relax
4077   \let\next\gmd@adef@scanfamact
4078   \else
4079   \PackageError{gmdoc}{Neither `other' nor `active'! ! Make
4080   it
4081   `other' with \bslash AddtoPrivateOthers\bslash }%
4082   \fi
4083   \fi}
```

```

4083   \next}
4085 \def\gmd@adef@scanfamoth#1{%
4086   \edef\gmd@adef@fam{\@gobble#1}% there is always \gmd@charbychar first.
4088   \gmd@adef@dofam
4089   <\gmd@adef@fam>%
4090   \gmd@charbychar}
4092 \foone{\catcode`\\>\active}
4093 {\def\gmd@adef@scanfamact#1{%
4094   \edef\gmd@adef@fam{\@gobble#1}% there is always \gmd@charbychar
4095   first.
4096   \gmd@adef@dofam
4097   <\gmd@adef@fam>%
4098   \gmd@charbychar}%
4099 }

```

The hook of the left brace consists of `\ifcase` that logically consists of three subcases:

- 0 —the default: do nothing in particular;
- 1 —the detected defining command has one mandatory argument (is of the `text` type, including `kvoptions` 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
4114 \def\gm@lbracehook{%
4115   \ifcase\gmd@lbracecase\relax
4116   \or% when 1
4117     \afterfi{%
4118       \gdef\gmd@lbracecase{o}%
4119       \gmd@adef@scanname}%
4120   \or% when 2—the first mandatory argument of two (\define@(...)\key)
4121     \afterfi{%
4122       \gdef\gmd@lbracecase{3}%
4123       \gmd@adef@scanDKfam}%
4124   \or% when 3—the second mandatory argument of two (the key name).
4125     \afterfi{%
4126       \gdef\gmd@lbracecase{o}%
4127       \gmd@adef@scanname}%
4128   \fi}
4130 \def\gmd@lbracecase{o}{ we initialize the hook caser.

```

And we define the inner left brace macros:

```

4135 \foone{\catcode`\\[1\catcode`\\]2\catcode`\\]12]
4136 [% Note that till line ?? 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
4143 \def\gmd@adef@scanDKfam#1{%
4144   \edef\gmd@adef@fam[\@gobble#1]{ there is always \gmd@charbychar first.
4145   \gmd@adef@dofam
4146   \gmd@adef@fam}%
4148 \gmd@charbychar]

```

```

\gmd@adef@scancode 4151 \def\gmd@adef@scancode#1}{%
4152   \Cmakeother\%
4153   \Cmakeother\<%

```

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

```

4156   \gmd@adef@deftext [#1]{%
4157     \Cobble#1}%
4158     \gmd@charbychar]
4159   ]

```

```

\gmd@adef@dofam 4162 \def\gmd@adef@dofam{%
4163   \ifnum1=0\csname\gmd@adef@KVfamset@\gmd@adef@currdef\endcsname
4164     \relax% a family declared with \DeclareDefining overrides the one cur-
        rently scanned.
4166   \else
4167     \edef\gmu@resa{%
4168       \gdef\@xa\@nx
4169         \csname\gmd@adef@KVfam@\gmd@adef@currdef\endcsname
4170         {\ifx\gmd@adef@fam\empty
4171           \else\gmd@adef@fam\@%
4172             \fi}}%
4173   \gmu@resa
4174   \fi}

```

```

\gmd@adef@deftext 4176 \def\gmd@adef@deftext#1{%
4177   \edef\macro@pname{\Cobble#1}% we gobble \gmd@charbychar, cf. above.
4178   \@xa\Text@Marginize\@xa{\macro@pname}%
4179   \gmd@adef@indextext
4180   \edef\gmd@adef@altindex{%
4181     \csname\gmd@adef@prefix@\gmd@adef@currdef\endcsname}%

```

and we add the `xkeyval` header if we are in `xkeyval` definition.

```

4184   \ifnum1=0\csname\gmd@adef@KV@\gmd@adef@currdef\endcsname%
4185     \relax% The
4186     cs \gmd@adef@KV@def. command is defined {1} (so \ifnum gets
4187     1=0\relax—true) iff def. command is a keyval definition. In that case we
4188     check for the KVprefix and KVfamily. (Otherwise \gmd@adef@KV@def.
4189     command is undefined so \ifnum gets 1=0\relax—false.)
4190     \edef\gmd@adef@altindex{%
4191       \gmd@adef@altindex
4192       \csname\gmd@adef@KVpref@\gmd@adef@currdef\endcsname}%
4193     \edef\gmd@adef@altindex{%
4194       \gmd@adef@altindex
4195       \csname\gmd@adef@KVfam@\gmd@adef@currdef\endcsname}%
4196     \fi
4197     \ifx\gmd@adef@altindex\empty
4198       \else% we make another index entry of the definiendum with prefix/KVheader.
4199         \edef\macro@pname{\gmd@adef@altindex\macro@pname}%
4200         \gmd@adef@indextext
4201       \fi}

```

```

\gmd@adef@indextext 4203 \def\gmd@adef@indextext{%
4204   \@xa\@defentryze\@xa{\macro@pname}{o}}% declare the definiendum has to
        have a definition entry and in the changes history should appear without
        backslash.

```

```

4207  \gmd@doindexingtext% redefine \do to an indexing macro.
4209  \xa\do\xaf{\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:

```

4223 \DeclareDefining[star=false]\def
\pdef 4224 \DeclareDefining[star=false]\pdef% it's a gutils' shorthand for \protected
      % \def.
\provide 4225 \DeclareDefining[star=false]\provide% a gutils' conditional \def.
\pprovide 4226 \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 4234 \def\UnDef{%
4238   \gmd@adef@selfrestore\def
4239 }

```

4241 \StoreMacro\UnDef% because the 'hiding' commands relax it.

```

\HideDef 4243 \def\HideDef{%
\relaxen 4245   \@ifstar\UnDef{\HideDefining\def\relaxen\UnDef}}

```

```

\ResumeDef 4247 \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 4254 \DeclareDefining[star=false]\newcount
\newdimen 4255 \DeclareDefining[star=false]\newdimen
\newskip 4256 \DeclareDefining[star=false]\newskip
4257 \DeclareDefining[star=false]\newif
\newtoks 4258 \DeclareDefining[star=false]\newtoks
\newbox 4259 \DeclareDefining[star=false]\newbox
\newread 4260 \DeclareDefining[star=false]\newread
\newwrite 4261 \DeclareDefining[star=false]\newwrite
\newlength 4262 \DeclareDefining[star=false]\newlength

```

```

\DeclareDocumentCommand 4263 \DeclareDefining[star=false]\DeclareDocumentCommand

```

4267 \DeclareDefining\newcommand

\renewcommand 4268 \DeclareDefining\renewcommand

4269 \DeclareDefining\providecommand

\DeclareRobustCommand 4270 \DeclareDefining\DeclareRobustCommand

\DeclareTextCommand 4271 \DeclareDefining\DeclareTextCommand

\DeclareTextCommandDefault 4272 \DeclareDefining\DeclareTextCommandDefault

4274 \DeclareDefining*\newenvironment

4275 \DeclareDefining*\renewenvironment

\DeclareOption 4276 \DeclareDefining*\DeclareOption

%\DeclareDefining*\@namedef

\newcounter 4282 \DeclareDefining*[prefix=\bslash_c@]\newcounter% this prefix provides indexing also \c@<counter>.

\define@key 4285 \DeclareDefining[type=dk, prefix=\bslash]\define@key

```

\define@boolkey 4286 \DeclareDefining[type=dk,\prefix=\bslash_if]\define@boolkey% the al-
    ternate index entry will be \if<KVpref>@\<KVfam>@\<key name>
\define@choicekey 4289 \DeclareDefining[type=dk,\prefix=\bslash]\define@choicekey
\DeclareOptionX 4291 \DeclareDefining[type=dox,\prefix=\bslash]\DeclareOptionX% the alter-
    nate 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 4300 \newcommand*\DeclareDOXHead[2][\gmd@KVprefdefault]{%
    \csname\DeclareDefining\endcsname
    [type=dox,\prefix=\bslash,\KVpref=#1,\KVfam=#2]%
\DeclareOptionX 4303 \DeclareOptionX
    }

```

An example:

```
4310 \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

```
Chne0elze 4315 \DeclareOptionX<AntonW>{Chne0elze}
```

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 4324 \DeclareDefining[type=kvo,\prefix=\bslash,\KVpref=]%
    \DeclareStringOption
\DeclareBoolOption 4325 \DeclareDefining[type=kvo,\prefix=\bslash,\KVpref=]%
    \DeclareBoolOption
\DeclareComplementaryOption 4326 \DeclareDefining[type=kvo,\prefix=\bslash,\KVpref=]%
    \DeclareComplementaryOption
\DeclareVoidOption 4327 \DeclareDefining[type=kvo,\prefix=\bslash,\KVpref=]%
    \DeclareVoidOption

```

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

```

4331 \def\DeclareKVOFam#1{%
4332   \def\do##1{%
4333     \csname\DeclareDefining\endcsname
4334     [type=kvo,\prefix=\bslash,\KVpref=,\KVfam=#1]##1}%
4335   \do\DeclareStringOption
4336   \do\DeclareBoolOption
4337   \do\DeclareComplementaryOption
4338   \do\DeclareVoidOption
4339 }

```

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 4376 \def\HideAllDefining{%
 4377   \ifnumo=0\csname\gmd@adef@allstored\endcsname
 4378     \SMglobal\StoreMacro\gmd@detectors
 4379     \global\@namedef{\gmd@adef@allstored}{\_1}%
 4380   \fi
 4381   \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 4387 \def\ResumeAllDefining{%
 4388   \ifnumi=0\csname\gmd@adef@allstored\endcsname\relax
 4389     \SMglobal\RestoreMacro\gmd@detectors
 4390     \SMglobal\RestoreMacro\UnDef
 4391     \global\@namedef{\gmd@adef@allstored}{\_0}%
 4392   \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 4404 \def\HideDefining{\begingroup
 4405   \MakePrivateLetters
 4406   \@ifstar{\Hide@DfngOnce}{\Hide@Dfng}
\Hide@Dfng 4410 \def\Hide@Dfng#1{%
 4411   \escapechar\m@ne
 4412   \gn@melet{\gmd@detect@\string#1}{\relax}%
 4413   \gn@melet{\gmd@detect@\string#1*}{\relax}%
 4414   \ifx\def#1\global\relaxen\UnDef\fi
 4415   \endgroup}
\Hide@DfngOnce 4417 \def\Hide@DfngOnce#1{%
 4418   \gmd@adef@selfrestore#1%
 4419   \endgroup}
 4420 \def\gmd@adef@selfrestore#1{%
 4421   \escapechar\m@ne
 4422   \@ifundefined{\gmd@detect@\string#1}{%
 4423     \SMglobal\@xa\StoreMacro
 4424     \csname\gmd@detect@\string#1\endcsname{}%
 4425   }%
 4426   \global\@namedef{\gmd@detect@\string#1}{%
 4427     \gnx\ifx\@xa\gnx\csname\gmd@detectname@\string#1\endcsname{}%
 4428     \gnx\macro@pname
 4429     \def\@nx\next{\% this \next will be executed in line 3733.
 4430       \SMglobal\RestoreMacro\% they both are \protected.
 4431       \@xa\gnx\csname\gmd@detect@\string#1\endcsname}
```

```

4434      \c@nx\gmd@charbychar}%
4435      \c@nx\fi}%
4436      }% of \c@nameedef.
4437      }% of \gmd@adef@selfrestore.

```

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 4450 \def\ResumeDefining{\begingroup
4451   \MakePrivateLetters
4452   \gmd@ResumeDfng}
\gmd@ResumeDfng 4454 \def\gmd@ResumeDfng#1{%
4455   \escapechar\m@ne
4456   \SMglobal\RestoreMacro*{\gmd@detect@\string#1}%
4457   \SMglobal\RestoreMacro*{\gmd@detect@\string#1*}%
4458   \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 $_2$ macro's name, usually `\macro@pname`, built with `\stringing` every char in lines 3551, 3571 and 3583. #3 is used only to test if the entry should be specially formatted.

```

\index@macro 4470 \newcommand*\index@macro[3][\verb+ +]{%
4471   \@ifundefined{\gmd/idxcl/#3}{%
4472     {%
4473       \ifundefined{\gmd/defentry/#3}{%
4474         {%
4475           \ifundefined{\gmd/usgentry/#3}{%
4476             {%
4477               \edef\kind@fentry{\CommonEntryCmd}}%
4478             {%
4479               \def\kind@fentry{UsgEntry}}%
4480               \un@usgentryze{#3}}%
4481             }%
4482             {%
4483               \def\kind@fentry{DefEntry}}%
4484               \un@defentryze{#3}}%
4485             }%
4486             {%
4487               \if@pageindex\@pageinindexfalse\fi% should it be here or there?
4488                 Definitely here because we'll wish to switch the switch with a declaration.
4489               \if@pageinindex
4490                 \edef\gmu@tempa{\gmdindexpagecs{\HLPrefix}{\kind@fentry}{%
4491                   \EntryPrefix}}%
4492               \else
4493                 \edef\gmu@tempa{\gmdindexrefcs{\HLPrefix}{\kind@fentry}{%
4494                   \EntryPrefix}}%
4495               \fi
4496               \edef\gmu@tempa{\IndexPrefix#2\actualchar}%
4497               \quotetechar\bslash_\verb*#1\quoted@eschar#2#1% The last macro in
4498               this line usually means the first two, but in some cases it's redefined
4499               to be empty (when we use \index@macro to index not a cs).

```

```

4499      \encapchar\gmu@tempa}%
4500      \@xa\special@index\@xa{\gmu@tempa}%
4501      We give the indexing macro the
4502      argument expanded so that hyperref may see the explicit encapchar
4503      in order not to add its own encapsulation of \hyperpage when the
4504      (default) hyperindex=true option is in force. (After this setting the
4505      \edefs in the above may be changed to \defs.)
4512      \}{}}% closing of gmd/iexcl/ test.
4513      }

\un@defentryze 4517 \def\un@defentryze#1{%
4518   \@xa\g@relaxen\csname_gmd/defentry/#1\endcsname
4519   \ifx\gmd@detectors\empty
4520     \g@relaxen\last@defmark
4521   \fi}% the last macro (assuming \fi is not a macro :-) is only used by \changes. If
4522   we are in the scope of automatic detection of definitions, we want to be able
4523   not to use \Define but write \changes after a definition and get proper en-
4524   try. Note that in case of automatic detection of definitions \last@defmark's
4525   value keeps until the next definition.

\un@usgentryze 4528 \def\un@usgentryze#1{%
4529   \@xa\g@relaxen\csname_gmd/usgentry/#1\endcsname}
4531 \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@pageinclistindex 4536 \newif\if@pageinclistindex
\quoted@eschar 4538 \newcommand*\quoted@eschar{\quotec\bslash}%
4539   we'll redefine it when in-
4540   dexing an environment.

```

Let's initialize \IndexPrefix

```
\IndexPrefix 4542 \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{\<package name>!} for instance and it will work as main level index entry and \def\HLPrefix{\<package name>} as a prefix in hypertargets in the codelines. They are redefined by \DocInclude e.g.

```

4551 \if@linesnotnum\@pageindextrue\fi
4552 \AtBeginDocument{%
4553   \if@pageindex
\gmdindexrefs 4554   \def\gmdindexrefs#1#2#3#4{\csname#2\endcsname{\hyperpage{%
4555   #4}}}}%
4556   in the page case we gobble the third argument that is supposed
4557   to be the entry prefix.
4558   \let\gmdindexpagecs=\gmdindexrefs
4559
\gmdindexrefs 4560   \else
4561     \def\gmdindexrefs#1#2#3#4{\gmiflink[clnum.#4]{%
4562       \csname#2\endcsname{#4}}}}%
4563   \def\gmdindexpagecs#1#2#3#4{\hyperlink{page.#4}{%
4564     \csname#2\endcsname{\gmd@revprefix{#3}#4}}}}%
4565
\gmd@revprefix 4566   \def\gmd@revprefix#1{%
4567     \def\gmu@tempa{#1}%
4568     \ifx\gmu@tempa\empty\p.\,\fi}
4569
\HLPrefix 4570   \providecommand*\HLPrefix{}%
4571   it'll be the hypertargets names' prefix in

```

multi-docs. Moreover, it showed that if it was empty, hyperref saw duplicates of the hyper destinations, which was perfectly understandable (`codelinenum.123` made by `\refstepcounter` and `codelinenum.123` made by `\gmp hypertarget`). But since v0.98 it is not a problem anymore because during the automatic `\hypertarget` the lines are labeled `c1num.<number>`. When `\HLPrefix` was defined as dot, `MakeIndex` rejected the entries as ‘illegal page number’.

4582 \fi}

The definition is postponed till `\begin{document}` because of the `\PageIndex` declaration (added for doc-compatibility), see line 7410.

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 `linenumber` 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 4597 \def\DefEntry#1{\underline{#1}}
\UsgEntry 4598 \def\UsgEntry#1{\textit{#1}}
```

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

```
\CommonEntryCmd 4600 \def\CommonEntryCmd{\relax}
```

In line 4477 it’s `\edef` 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 641.

Now let’s `\def` the macros declaring a `cs` to be indexed special way. Each declaration puts the `12ed` name of the macro given it as the argument into proper macro to be `\ifx`ed in lines 4473 and 4475 respectively.

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

```
\DefIndex 4616 \outer\def\DefIndex{\begingroup
 4617   \MakePrivateLetters
 4618   \@ifstar{\MakePrivateOthers\Code@DefIndexStar}{%
    \Code@DefIndex}}
```

```
\Code@DefIndex 4623 \long\def\Code@DefIndex#1{\endgroup%
 4624   \escapechar\m@ne% because we will compare the macro's name with a string
   without the backslash.
 4626   \@defentryze{#1}{#1}}
```

```
\Code@DefIndexStar 4630 \long\def\Code@DefIndexStar#1{%
 4631   \endgroup
 4632   \addtoestoindex{#1}%
 4633   \@defentryze{#1}{#1}}
```

```
\gmd@justadot 4635 \def\gmd@justadot{.}
```

```
\@defentryze 4637 \long\def\@defentryze#1#2{%
 4638   \xa\glet\csname\gmd@defentry\string#1\endcsname%
   \gmd@justadot% The
   LATEX \namedef macro could not be used since it's not 'long'.
\last@defmark 4641 \xdef\last@defmark{\string#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 \\).
 4646 \@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.

```
\@usgentryze 4650 \long\def\@usgentryze#1{%
  4651   \@xa\let\csname_gmd/usgentry\string#1\endcsname\gmd@justadot}
    Initialize \envirs@toindex
  4654 \emptyify\envirs@toindex
```

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

```
\CodeUsgIndex 4657 \outer\def\CodeUsgIndex{\begingroup
  4658   \MakePrivateLetters
  4659   \@ifstarl{\MakePrivateOthers\Code@UsgIndexStar}{%
    \Code@UsgIndex}}
```

The * possibility is for marking environments etc.

```
\Code@UsgIndex 4662 \long\def\Code@UsgIndex#1{\endgroup{%
  4663   \escapechar\m@ne
  4664   \global\@usgentryze{\#1}}}
```

```
\Code@UsgIndexStar 4667 \long\def\Code@UsgIndexStar#1{%
  4668   \endgroup
  4669   \addto@estoindex{\#1}%
  4670   \@usgentryze{\#1}}
```

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 4674 \outer\def\CodeCommonIndex{\begingroup
  4675   \MakePrivateLetters
  4676   \@ifstarl{\MakePrivateOthers\Code@CommonIndexStar}{%
    \Code@CommonIndex}}
```

```
\Code@CommonIndex 4679 \long\def\Code@CommonIndex#1{\endgroup}
```

```
\Code@CommonIndexStar 4682 \long\def\Code@CommonIndexStar#1{%
  4683   \endgroup\addto@estoindex{\#1}}
```

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

```
\text@indexmacro 4688 \long\def\text@indexmacro#1{%
  4689   {\escapechar\m@ne\xdef\macro@pname{\xiistring#1}}%
  4690   \@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.
  4694   \if\verb@im@firstpar\macro@pname
    \def\im@firstpar{[$]}%
  4696   \else\def\im@firstpar{}%
  4697   \fi
  4698   {\do@properindex% see line 5036.
    \@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 4704 \def\quote@mname{%
```

```

\macro@iname 4705 \def\macro@iname{}%
4706 \quote@charbychar}

\quote@charbychar 4709 \def\quote@charbychar#1{%
4710 \if\relax#1% finish quoting when you meet \relax or
4711 \else
4712 \quote@char#1%
4713 \xdef\macro@iname{\macro@iname\gmd@maybequote#1}%
4714 \afterfi\quote@charbychar
4715 \fi}

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 4721 \def\TextUsgIndex{\begingroup
4722 \MakePrivateLetters
4723 \@ifstarl{\MakePrivateOthers\Text@UsgIndexStar}{%
4724 \Text@UsgIndex}

\Text@UsgIndex 4726 \long\def\Text@UsgIndex#1{%
4727 \endgroup\@usgentryze#1%
4728 \text@indexmacro#1}

\Text@UsgIndexStar 4731 \long\def\Text@UsgIndexStar#1{\endgroup\@usgentryze{#1}%
4732 \text@indexenvir{#1}}

\text@indexenvir 4734 \long\def\text@indexenvir#1{%
4735 \edef\macro@pname{\xiistring#1}%
4736 \if\bslash@\xa@\firstofmany\macro@pname\@nil% if \stringed #1 begins with a backslash, we will gobble it to make MakeIndex not see it.
4737 \edef\gmu@tempa{\xa@gobble\macro@pname}%
4738 \tempswattrue
4739 \else
4740 \let\gmu@tempa\macro@pname
4741 \tempswafalse
4742 \fi
4743 \xa\quote@name\gmu@tempa\relax% we process \stringed #1 char by char
4744 and quote MakeIndex controls. \relax is the iterating macro's stopper. The
4745 quoted \stringed #1 shall be the meaning of \macro@iname.
4746 \if@tempswa
4747 \def\quoted@eschar{\quotechar\bslash}%
4748 \else\@empty\quoted@eschar\fi% we won't print any backslash before
4749 an environment's name, but we will before a cs's name.
4750 \do@properindex% see line 5036.
4751 \index@macro\macro@iname\macro@pname}

\quoted@eschar 4752 \def\quoted@eschar{\quotechar\bslash}%
4753 \else\@empty\quoted@eschar\fi% we won't print any backslash before
4754 an environment's name, but we will before a cs's name.
4755 \do@properindex% see line 5036.
4756 \index@macro\macro@iname\macro@pname}

\TextCommonIndex 4757 \def\TextCommonIndex{\begingroup
4758 \MakePrivateLetters
4759 \@ifstarl{\MakePrivateOthers\Text@CommonIndexStar}{%
4760 \Text@CommonIndex}

\Text@CommonIndex 4761 \long\def\Text@CommonIndex#1{\endgroup
4762 \text@indexmacro#1}

\Text@CommonIndexStar 4763 \long\def\Text@CommonIndexStar#1{\endgroup
4764 \text@indexenvir{#1}}

```

As you see in the lines 4484 and 4480, 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 4773 \outer\def\CodeMarginize{\begingroup
4774   \MakePrivateLetters
4775   \@ifstarl
4776     {\MakePrivateOthers\egCode@MarginizeEnvir}
4777     {\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\catcodeing group.

```

\egCode@MarginizeMacro 4783 \long\def\egCode@MarginizeMacro#1{\endgroup
4784   \Code@MarginizeMacro#1}
\Code@MarginizeMacro 4787 \long\def\Code@MarginizeMacro#1{{\escapechar\m@ne
4788   \oxa\glet\csname_gmd/2marpar/\string#1\endcsname\gmd@justadot
4790   {}}
\egCode@MarginizeEnvir 4793 \long\def\egCode@MarginizeEnvir#1{\endgroup
4794   \Code@MarginizeEnvir{#1}}
\Code@MarginizeEnvir 4797 \long\def\Code@MarginizeEnvir#1{\addtoestomarginpar{#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 4803 \def\mark@envir{%
4804   \ifx\envirs@tomarginpar\@empty
4805   \else
4806     \let\do\Text@Marginize
4807     \envirs@tomarginpar%
4808     \g@emptyify\envirs@tomarginpar%
4809   \fi
4810   \ifx\envirs@toindex\@empty
4811   \else
4812     \gmd@doindexingtext
4813     \envirs@toindex
4814     \g@emptyify\envirs@toindex%
4815   \fi}
\gmd@doindexingtext 4817 \def\gmd@doindexingtext{%
4818   \def\do##1{\% 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 4207.
4819   \if\bslash\@firstofmany##1\@nil% if ##1 begins with a backslash, we
4820     will gobble it for MakeIndex not see it.
4821   \edef\gmd@resa{\@gobble##1}%
4822   \tempswatrue
4823   \else
4824   \edef\gmd@resa##1\@tempswafalse
4825   \fi
4826   \oxa\quote@mname\gmd@resa\relax% see line 4745 & subs. for commentary.
4827   \if@tempswa
4828     \def\quoted@eschar{\quotechar\bslash}%
4829     \else\@emptyify\quoted@eschar\fi
\quoted@eschar 4833 \else\@emptyify\quoted@eschar\fi
4834

```

```

4835      \index@macro\macro@iname{##1}}}%  

4836 }

```

One very important thing: initialisation of the list macros:

```

4840 \@emptyify\envirs@tomarginpar  

4841 \@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 4848 \outer\def\Define{\begingroup  
4849   \MakePrivateLetters
```

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

```

4854   \@ifstarl{\MakePrivateOthers\Code@DefEnvir}{\Code@DefMacro}  

\CodeUsage 4856 \outer\def\CodeUsage{\begingroup  
4857   \MakePrivateLetters  
4858   \@ifstarl{\MakePrivateOthers\Code@UsgEnvir}{\Code@UsgMacro}}

```

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

```
\Code@DefMacro 4861 \long\def\Code@DefMacro#1{%
4862   \Code@DefIndex#1% we use the internal macro; it'll close the group.
4863   \Code@MarginizeMacro#1}
```

```
\Code@UsgMacro 4866 \long\def\Code@UsgMacro#1{%
4867   \Code@UsgIndex#1% here also the internal macro; it'll close the group
4868   \Code@MarginizeMacro#1}
```

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

```
\codeline@wrindex 4873 \def\codeline@wrindex#1{\if@filesw
4874   \immediate\write\@indexfile
4875   {\string\indexentry{#1}%
4876    {\HLPrefix\number\c@codelinenum}}\fi}
\codeline@glossary 4880 \def\codeline@glossary#1{%
4881   It doesn't need to establish a group since it is always called in a group.
4882   \if@pageinclistindex
4883     \edef\gmu@tempa{\gmdindexpagecs{\HLPrefix}{relax}{%
4884       \EntryPrefix}}%
4885     \else
4886       \edef\gmu@tempa{\gmdindexrefcs{\HLPrefix}{relax}{%
4887         \EntryPrefix}}%
4888       relax stands for the formatting command. But we
4889       don't want to do anything special with the change history entries.
4890     \fi
4891     \protected@edef\gmu@tempa{%
4892       \codeline@wrindex{\glossaryfile}%
4893       {\string\glossaryentry{#1\encapchar\gmu@tempa}%
4894        {\HLPrefix\number\c@codelinenum}}}}%
4895     \gmu@tempa
4896   }
```

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

```
4900 \AtBeginDocument{%
```

```

4901  \if@pageindex
4902    \let\special@index=\index
4903    \let\gmd@glossary\glossary
4904  \else
4906    \let\special@index=\codeline@wrindex
4907    \let\gmd@glossary\codeline@glossary
4909  \fi}%
4910  postponed till \begin{document} with respect of doc-like declarations.

```

And in case we don't want to index:

```

\gag@index 4913 \def\gag@index{\let\index=\@gobble
4915   \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:

```
4920 \StoreMacros{\index\codeline@wrindex}
```

```
\ungag@index 4922 \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 4938 \long\def\Code@DefEnvir#1{%
4939   \endgroup
4940   \addto@estomarginpar{#1}%
4941   \addto@estoindex{#1}%
4942   \@defentryze{#1}{o}}

```

```

\Code@UsgEnvir 4945 \long\def\Code@UsgEnvir#1{%
4946   \endgroup
4947   \addto@estomarginpar{#1}%
4948   \addto@estoindex{#1}%
4949   \@usgentryze{#1}}

```

```

\addto@estomarginpar 4952 \long\def\addto@estomarginpar#1{%
4953   \edef\gmu@tempa{\@nx\do{\xiistring#1}}% we \string the argument to al-
4954   low it to be a control sequence.
4955   \@xa\addtomacro\@xa\envirs@tomarginpar\@xa{\gmu@tempa}}

```

```

\addto@estoindex 4958 \long\def\addto@estoindex#1{%
4959   \edef\gmu@tempa{\@nx\do{\xiistring#1}}
4960   \@xa\addtomacro\@xa\envirs@toindex\@xa{\gmu@tempa}}

```

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 4967 \def\TextUsage{\begingroup
4969   \MakePrivateLetters
4970   \@ifstar{ \MakePrivateOthers\Text@UsgEnvir }{ \Text@UsgMacro } }

```

```

\Text@UsgMacro 4973 \long\def\Text@UsgMacro#1{%
4974   \endgroup\@tt\xiistring#1}%

```

```

4975  \Text@Marginize#1%
4976  \begingroup\Code@UsgIndex#1% we declare the kind of formatting of the entry.
4977  \text@indexmacro#1}

\Text@UsgEnvir 4980 \long\def\Text@UsgEnvir#1{%
4981  \endgroup{\tt\xiistring#1}%
4982  \Text@Marginize{#1}%
4983  \@usgentryze{#1}% we declare the ‘usage’ kind of formatting of the entry and
        index the sequence #1.
4985  \text@indexenvir{#1}}

```

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 4991 \def\TextMarginize{\begingroup
4992  \MakePrivateLetters
4993  \@ifstarl{\MakePrivateOthers\egText@Marginize}{%
        \egText@Marginize}}
\egText@Marginize 4996 \long\def\egText@Marginize#1{\endgroup
4997  \Text@Marginize#1}

```

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

```

5001 \if@marginparsused
5002  \reversemarginpar
5003  \marginparpush\z@
5004  \marginparwidth8pc\relax

```

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

```

\gmdmarginpar 5009 \long\def\gmdmarginpar#1{%
5010  \marginpar{\raggedleft\strut
5011  \hskipoptplus1ooptminus1oopt%
5012  #1}}%
5014 \else
\gmdmarginpar 5015 \long\def\gmdmarginpar#1{%
5016 \fi
\Text@Marginize 5018 \long\def\Text@Marginize#1{%
5019  \gmdmarginpar{\marginpartt\xiistring#1}}

```

Note that the above macro will just gobble its argument 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.)

```
5026 \let\marginpartt\tt
```

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 4907. 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 5036 \def\do@properindex{%
5037  \if@printalllinenos\else
```

```

5038   \@pageincludindextrue
5039   \let\special@index=\index
5040   \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* 5060 \newenvironment*{macrocode*}{%
5061   \if@codeskipput\else\par\addvspace\CodeTopsep%
5062     \@codeskipputgtrue\fi
5063   \QueerEOL}%
5064   {\par\addvspace\CodeTopsep\CodeEscapeChar\\}

```

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

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

```

macrocode 5071 \newenvironment*{macrocode}{%
5072   \if@codeskipput\else\par\addvspace\CodeTopsep%
5073     \@codeskipputgtrue\fi
5074   \QueerEOL}%
5075   {\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*”⁸.

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 5093 \let\oldmc\macrocode
5094 \let\endoldmc\endmacrocode
oldmc* 5096 \n@melet{\oldmc*}{macrocode*}
5097 \n@melet{\endoldmc*}{endmacrocode*}

```

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

```

5101 \addtomacro\oldmc{\@oldmacrocode@launch}%
5102 \xa\addtomacro\csname\oldmc*\endcsname{%
5103   \@oldmacrocode@launch}
5106 \def\@oldmacrocode@launch{%
5107   \empty\gmd@textEOL% to disable it in \gmd@docstripdirective launched
5108   within the code.
5109   \gmd@ctallsetup
5110   \glet\stored@code@delim\code@delim
5111   \makeother\^\^B\CodeDelim\^\^B%

```

⁸ Richard Strauss after Oscar Wilde, *Salomé*.

```

5112  \ttverbatim_{\gmd@DoTeXCodeSpace}%
5113  \@makeother\| because \ttverbatim doesn't do that.
5114  \MakePrivateLetters% see line 3506.
5116  \docstrips@percent_{\@makeother\>}%

```

sine qua non of the automatic delimiting is replacing possible *₁₂in the environment's name with *₁₁. Not to complicate assume * 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 *) will be the same in the verbatim text.

```

5123  \@xa\gmd@currenvxistar\currenvir*\relax
5124  \@oldmacrocode}
5126 \foone{\catcode`*_{11} }
5127 {\def\gm@xistar{*}}
\gm@xistar 5129 \def\gmd@currenvxistar#1#2\relax{%
5130   \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 *₁₁or else nothing will be added. (Note that a * not at the end of env. name would cause a disaster.)

```

5140 \foone{%
5141 \catcode`[=_1\catcode`]=_2
5142 \catcode`\{=\active_{\@makeother\}
5143 \@makeother\^\^B
5144 \catcode`/_=o_{\catcode`\\=\active
5145 \catcode`&=_4\catcode`*_11
5146 \catcode`\%=\active_{\obeyspaces}\&%
5147 [& here the \foone's second pseudo-argument begins
@oldmacrocode 5149 /def/@oldmacrocode[&
5150 /bgroup/let_=relax& to avoid writing /@nx four times.
5151 /xdef/oldmc@def [&
5152 /def/@nx/oldmc@end####1/@nx%_{\end}/@nx\end&
5153 /@nx{@currenvir} [&
5154 #####1^\^B/@nx/end[/@currenvir]/@nx/gmd@oldmcfinis]]&
5155 /egroup& now \oldmc@edef is defined to have one parameter delimited with
      \end{{current env.'s name}}
5157 /oldmc@def&
5158 /oldmc@end]&
5159 ]
5161 \def\gmd@oldmcfinis{%
5162   \@xa\CodeDelim\stored@code@delim
5163   \gmd@mchook}% see line 7165
5165 \def\OldMacrocodes{%
5167   \let\macrocode\oldmc
5168   \n@melet{macrocode*}{oldmc*}}

```

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

```

5176 \foone{\catcode`\%\active}
5177 {\def\docstrips@percent{\catcode`\%\active
5178   \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 \ttverbatim all the < are 'other').

```
\gmd@codecheckifds 5186 \def\gmd@codecheckifds#1#2{%
  note that #1 is just to gobble \gmd@charbychar
  token.

  5189 \if@dsdir\@dsdirfalse
  5190   \if\@nx<\@nx#2\afterfifi\gmd@docstripdirective
  5191   \else\afterfifi{\xiipercent#1#2}%
  5192   \fi
  5193 \else\afterfi{\xiipercent#1#2}%
  5194 \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 5201 \newenvironment{macro}{%
  5202   \tempskipa=\MacroTopsep
  5203   \if@codeskipput\advance\tempskipa by-\CodeTopsep\fi
  5204   \par\addvspace{\tempskipa}\@codeskipputtrue
  5205   \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.)
  5209 \gmd@ifoneton\Hybrid@DefMacro\Hybrid@DefEnvir}%
  5211 {\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:

```
5224 \namedef{macro*}{\let\gmd@ifoneton\@secondoftwo\macro}
5226 \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 5244 \def\gmd@ifsingle#1#2\@nil{%
  \gmu@tempa 5245 \def\gmu@tempa{#2}%
  5246 \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 \expandaftered).

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@ifonetoken 5259 \def\gmd@ifonetoken#1#2#3{%
\gmu@tempb 5260   \def\gmu@tempb{#3}% We hide #3 from TeX in case it's \if... or
                  so. \gmu@tempa is used in \gmd@ifsingle.
5262   \gmd@ifsingle#3\@nil
5263     \afterfi{\@xa#1\gmu@tempb}%
5264   \else
5265     \edef\gmu@tempa{\@xa\string\gmu@tempb}%
5266     \afterfi{\@xa#2\@xa{\gmu@tempa}}%
5267   \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 5272 \long\def\Hybrid@DefMacro#1{%
5273   \Code@DefIndex{#1}% this macro closes the group opened by \macro.
5274   \Text@MarginizeNext{#1}}
\Hybrid@DefEnvir 5276 \long\def\Hybrid@DefEnvir#1{%
5277   \Code@DefIndexStar{#1}% this macro also closes the group begun by \macro.
5279   \Text@MarginizeNext{#1}}
\Text@MarginizeNext 5281 \long\def\Text@MarginizeNext#1{%
5282   \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 5288 \long\def\gmd@evpaddonce#1{%
5289   \global\advance\gmd@oncenum\@ne
5290   \@xa\long\@xa\edef%
5291     \csname\gmd/evp/NeuroOncer\the\gmd@oncenum\endcsname{%
5292       \@nx\g@relaxen
5293       \csname\gmd/evp/NeuroOncer\the\gmd@oncenum\endcsname}% Why
                  does it work despite it shouldn't? Because when the cs got
                  with \csname... \endcsname is undefined, it's equivalent \relax
                  and therefore unexpandable. That's why it passes \edef and is able
                  to be assigned.
5298   \@xa\addtomacro\csname\gmd/evp/NeuroOncer\the\gmd@oncenum\%
                  \endcsname{#1}%
5299   \@xa\addto@hook\@xa\everypar\@xa{%
5300     \csname\gmd/evp/NeuroOncer\the\gmd@oncenum\endcsname}%
5301 }
```

\gmd@oncenum 5303 \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 TeXnical obstacle to do so. Therefore we define the environment, because of æ sthetic and psychological reasons.

```

5314 \Oxa\let\@xa\environment\csname_\macro*\endcsname
5315 \Oxa\let\@xa\endenvironment\csname_\endmacro*\endcsname

```

Index exclude list

We want some cses not to be indexed, e.g., the L^AT_EX internals and T_EX 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 \relax).⁹

```

\DoNotIndex 5330 \def\DoNotIndex{\bgroup\MakePrivateLetters\DoNot@Index}
\DoNot@Index 5338 \long\def\DoNot@Index#1{\egroup% we close the group,
5339   \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.
5342 \dont@index#1.}

\dont@index 5345 \long\def\dont@index#1{%
\gmu@tempa 5346 \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.
5349 \if\gmu@tempa.% a dot finishes expelling
5350 \else
5351   \if\gmu@tempa,% The list this macro is put before may contain commas and
      that's O.K., we just continue the work.
      \afterfifi\dont@index
5354 \else% what is else shall off the Index be expelled.
      {\escapechar\m@ne
       \xdef\gmu@tempa{\string#1}%
5355 \Oxa\let%
5356 \csname_\gmd/iexcl/\gmu@tempa\endcsname=\gmd@iedir% In the default
5357 case explained e.g. by the macro's name, the last macro's meaning is
5358 such that the test in line 4471 will turn false and the subject cs shall not
      be indexed. We \let not \def to spare TEX's memory.
      \afterfifi\dont@index
5363 \fi
5364 \fi}
5365 \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.

```

5374 \DoNotIndex{\DoNotIndex\%} the index entries of these two cses would be re-
      jected by MakeIndex anyway.

5377 \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 5381 \gdef\DefaultIndexExclusions{%
5382   \DoNotIndex{\O\O\par \O\beginparpenalty \O\empty\%}
5383   \DoNotIndex{\O\flushglue \O\gobble \O\input\%}
5384   \DoNotIndex{\O\makefnmark \O\makeother \O\maketitle\%}
5385   \DoNotIndex{\O\namedef \O\one \O\spaces \O\tempa\%}
5386   \DoNotIndex{\O\tempb \O\tempswafalse \O\tempswatrue\%}
5387   \DoNotIndex{\O\thanks \O\thefnmark \O\topnum\%}
5388   \DoNotIndex{\O\O\O\elt \O\forloop \O\fortmp \O\gtmpa
      \O\totalleftmargin\%}

```

⁹ This idea comes from Marcin Woliński.

```

5389 \DoNotIndex{\" \/\@ifundefined \@nil \@verbatim \vobeyspaces}%
5390 \DoNotIndex{\| \~\ active \advance \aftergroup \begingroup
5391   \bgroup}%
5392 \DoNotIndex{\mathcal \csname \def \documentstyle \dospecials
5393   \edef}%
5394 \DoNotIndex{\egroup}%
5395 \DoNotIndex{\else \endcsname \endgroup \endinput \endtrivlist}%
5396 \DoNotIndex{\expandafter \fi \fnsymbol \futurelet \gdef \global}%
5397 \DoNotIndex{\hbox \hss \if \if@inlabel \if@tempswa
5398   \if@twocolumn}%
5399 \DoNotIndex{\ifcase}%
5400 \DoNotIndex{\ifcat \iffalse \ifx \ignorespaces \index \input
5401   \item}%
5402 \DoNotIndex{\jobname \kern \leavevmode \leftskip \let \llap
5403   \lower}%
5404 \DoNotIndex{\m@ne \next \newpage \nobreak \noexpand
5405   \nonfrenchspacing}%
5406 \DoNotIndex{\obeylines \or \protect \raggedleft \rightskip \rm
5407   \sc}%
5408 \DoNotIndex{\setbox \setcounter \small \space \string \strut}%
5409 \DoNotIndex{\strutbox}%
5410 \DoNotIndex{\thefootnote \thispagestyle \topmargin \trivlist
5411   \tt}%
5412 \DoNotIndex{\twocolumn \typeout \vss \vtop \xdef \z@}%
5413 \DoNotIndex{\, \@bsphack \@esphack \@noligs \vobeyspaces
5414   \xverbatim}%
5415 \DoNotIndex{\` \catcode \end \escapechar \frenchspacing
5416   \glossary}%
5417 \DoNotIndex{\hangindent \hfil \hfill \hskip \hspace \ht \it
5418   \langle}%
5419 \DoNotIndex{\leaders \long \makelabel \marginpar \markboth
5420   \mathcode}%
5421 \DoNotIndex{\mathsurround \mbox} \% \newcount \newdimen \newskip
5422 \DoNotIndex{\nopagebreak}%
5423 \DoNotIndex{\parfillskip \parindent \parskip \penalty \raise
5424   \rangle}%
5425 \DoNotIndex{\section \setlength \TeX \topsep \underline \unskip}%
5426 \DoNotIndex{\vskip \vspace \widetilde \\ \% \@date \@defpar}%
5427 \DoNotIndex{\[ \]}% see line 5374.
5428 \DoNotIndex{\count @ \ifnum \loop \today \uppercase \uccode}%
5429 \DoNotIndex{\baselineskip \begin \tw@}%
5430 \DoNotIndex{\a \b \c \d \e \f \g \h \i \j \k \l \m \n \o \p \q}%
5431 \DoNotIndex{\r \s \t \u \v \w \x \y \z \A \B \C \D \E \F \G \H}%
5432 \DoNotIndex{\I \J \K \L \M \N \O \P \Q \R \S \T \U \V \W \X \Y \Z}%
5433 \DoNotIndex{\_1 \_2 \_3 \_4 \_5 \_6 \_7 \_8 \_9 \_o}%
5434 \DoNotIndex{\! \# \$ \& ' \(\) . . ; < = > ? \_}%
5435 \DoNotIndex{\+ seems to be
5436   so rarely used that it may be advisable to index it.}%
5437 \DoNotIndex{\discretionary \immediate \makeatletter
5438   \makeatother}%
5439 \DoNotIndex{\meaning \newenvironment \par \relax
5440   \renewenvironment}%
5441 \DoNotIndex{\repeat \scriptsize \selectfont \the \undefined}%
5442 \DoNotIndex{\arabic \do \makeindex \null \number \show \write

```

```

      \@ehc}%
5427 \DoNotIndex{\@author \@ehc \@ifstar \@sanitize \@title}%
5428 \DoNotIndex{\if@minipage \if@restonecol \ifeof \ifmmode}%
5429 \DoNotIndex{\lccode \% \newtoks
5430   \onecolumn \openin \p@ \SelfDocumenting}%
5431 \DoNotIndex{\setwidt\h \resetonecoltrue \resetonecolfalse
5432   \bf}%
5433 \DoNotIndex{\clearpage \closein \lowercase \inlabelfalse}%
5434 \DoNotIndex{\selectfont \mathcode \newmathalphabet \rmdefault}%
\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 `\newcommand` and `\(e/g/x)defs`, as the most common, in my opinion excluded should be.

And some my exclusions:

```

5447 \DoNotIndex{\@input \auxout \currentlabel \dblarg}%
5448 \DoNotIndex{\@ifdefinable \ifnextchar \ifpackageloaded}%
5449 \DoNotIndex{\@indexfile \let@token \sptoken \^}%
the latter comes
from cses like \^\M, see sec. 668.
5451 \DoNotIndex{\addto@hook \addvspace}%
5452 \DoNotIndex{\CurrentOption}%
5453 \DoNotIndex{\empty \empty \firstofone}%
5454 \DoNotIndex{\font \fontdimen \hangindent \hangafter}%
5455 \DoNotIndex{\hyperpage \hyperlink \hypertarget}%
5456 \DoNotIndex{\ifdim \ifhmode \iftrue \ifvmode \medskipamount}%
5457 \DoNotIndex{\message}%
5458 \DoNotIndex{\NeedsTeXFormat \newcommand \newif}%
5459 \DoNotIndex{\newlabel}%
5460 \DoNotIndex{\of}%
5462 \DoNotIndex{\phantom \ProcessOptions \protected@edef}%
5463 \DoNotIndex{\protected@xdef \protected@write}%
5464 \DoNotIndex{\ProvidesPackage \providecommand}%
5465 \DoNotIndex{\raggedright}%
5466 \DoNotIndex{\raisebox \refstepcounter \ref \rlap}%
5467 \DoNotIndex{\reserved@a \reserved@b \reserved@c \reserved@d}%
5468 \DoNotIndex{\stepcounter \subsection \textit \textsf \thepage
  \tiny}%
5469 \DoNotIndex{\copyright \footnote \label \LaTeX}%
5472 \DoNotIndex{\@eha \endparent \if@endpe \endpefalse
  \endpetrue}%
5473 \DoNotIndex{\@evenfoot \oddfoot \firstoftwo \secondoftwo}%
5474 \DoNotIndex{\@for \gobbletwo \idxitem \ifclassloaded}%
5475 \DoNotIndex{\ignorefalse \ignoretrue \ifignore}%
5476 \DoNotIndex{\@input @input}%
5477 \DoNotIndex{\@latex@error \mainaux \nameuse}%
5478 \DoNotIndex{\@nomath \oddfoot}%
% \onlypreamble should be indexed
  IMO.
5480 \DoNotIndex{\outerparskip \partaux \partlist \plus}%
5481 \DoNotIndex{\sverb \sxverbatim}%

```

```

5482  \DoNotIndex{\@tempcnta \@tempcntb \@tempskipa \@tempskipb}%
      I think the layout parameters even the kernel, should not be excluded:
      % \@topsep \@topsepadd \abovedisplayskip \clubpenalty etc.
5486  \DoNotIndex{@writeckpt}%
5487  \DoNotIndex{bfseries chapter part section subsection}%
5488  \DoNotIndex{subsubsection}%
5489  \DoNotIndex{char check@mathfonts closeout}%
5490  \DoNotIndex{fontsize footnotemark footnotetext
                 \footnotesize}%
5491  \DoNotIndex{g@addto@macro hfilneg Huge huge}%
5492  \DoNotIndex{hyphenchar if@partsw IfFileExists}%
5493  \DoNotIndex{include includeonly indexspace}%
5494  \DoNotIndex{itshape language LARGE Large large}%
5495  \DoNotIndex{lastbox lastskip m@th makeglossary}%
5496  \DoNotIndex{maketitle math@fontsf false math@fontstrue
                 \mathsf}%
5497  \DoNotIndex{MessageBreak noindent normalfont normalsize}%
5498  \DoNotIndex{on@line openout outer}%
5499  \DoNotIndex{parbox part rmfamily rule sbox}%
5500  \DoNotIndex{sf@size sffamily skip}%
5501  \DoNotIndex{textsc textup toks@ ttfamily vbox}%
% \DoNotIndex{\begin*} maybe in the future, if the idea gets popular...
5507  \DoNotIndex{hspace* newcommand* newenvironment*
                 providecommand*}%
5508  \DoNotIndex{renewenvironment* section* chapter*}%
5509  }% of \DefaultIndexExclusions.

```

I put all the expellings into a macro because I want them to be optional.

```
5512 \end{MakePrivateLetters}
```

And we execute it due to the (lack of) counter-corresponding option:

```

5516 \if@indexallmacros\else
5517   \DefaultIndexExclusions
5518 \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 5524 \def\DoIndex{\bgroup\MakePrivateLetters\Do@Index}
\Do@Index 5531 \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, she may use the \DefaultIndexExclusions declaration himself. This declaration oCSR, but anyway let's provide the counterpart. It oCSR, too.

```

UndoDefaultIndexExclusions 5540 \def\UndoDefaultIndexExclusions{%
5541   \StoreMacro\DoNotIndex
5543   \let\DoNotIndex\DoIndex
5545   \DefaultIndexExclusions
5547   \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
\index@prologue 5559 \long\def\IndexPrologue#1{\@bsphack\def\index@prologue{#1}%
                                         \@esphack}
\indexdiv 5562 \def\indexdiv{\@ifundefined{chapter}{\section*}{\chapter*}}
\index@prologue 5566 \@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
 5573   \ifx\HLPrefix\@empty
    5576     The\_numbers\_preceded\_with\_`p.'\_are\_page\_numbers.
 5577     \else The\_numbers\_with\_no\_prefix\_are\_page\_numbers.
 5578   \fi\fi
 5579   \ifx\IndexLinksBlack\relax\else
 5580     All\_the\_numbers\_are\_hyperlinks.
 5583   \fi
 5584   \gmd@dip@hook% this hook is intended to let a user add something without
                 redefining the entire prologue, see below.
5586 }{}}

```

During the preparation of this package for publishing I needed only to add something at the end of the default index prologue. So

```

\AtDIPilogue
\AtDIPilogue 5591 \@emptyify\gmd@dip@hook
 5592 \long\def\AtDIPilogue#1{\g@addto@macro\gmd@dip@hook{#1}}

```

The Author(s) of doc assume `multicol` is known not to everybody. My assumption is the other so

```
5597 \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 5606 \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 prologue
             and some index entries on the same page.
\c@IndexColumns 5609 \newcount\c@IndexColumns\c@IndexColumns=3
\theindex 5610 \renewenvironment{theindex}
 5611   {\begin{multicols}\c@IndexColumns[\index@prologue][\IndexMin]%
             \IndexLinksBlack
 5612             \IndexParms\let\item@\idxitem\ignorespaces}%
 5613   {\end{multicols}}
 5614 }
```

```

\IndexLinksBlack 5616 \def\IndexLinksBlack{\hypersetup{linkcolor=black}}% To make Adobe Reader
                  work faster.

\IndexParms 5619 \@ifundefined{IndexParms}
5620   {\def\IndexParms{%
5621     \parindent\z@
5622     \columnsep15pt
5623     \parskip\z@plus1pt
5624     \rightskip\z@plus1pt
5625     \mathsurround\z@
5626     \parfillskip=-15pt plus1fil}% doc defines this parameter rigid but
5627       that's because of the stretchable space (more precisely, a \dotfill) be-
      between the item and the entries. But in gmdoc we define no such special
      delimiters, so we add an infinite stretch.

5632   \small
5633   \def\@idxitem{\par\hangindent30pt}%
5634   \def\subitem{\@idxitem\hspace*{15pt}}%
5635   \def\subsubitem{\@idxitem\hspace*{25pt}}%
5636   \def\indexspace{\par\vspace{10pt plus2pt minus3pt}}%
5637   \ifx\EntryPrefix\empty\else\raggedright\fi% long (actually, a quite
      short but nonempty entry prefix) made space stretches so terribly large
      in the justified paragraphs that we should make \raggedright rather.
5641   \ifnum\c@IndexColumns>\tw@ \raggedright\fi% the numbers in nar-
      row columns look better when they are \raggedright in my opinion.
5643   }{}}

\PrintIndex 5645 \def\PrintIndex{%
5646   we ensure the standard meaning of the line end character not
5647   to cause a disaster.
5648   \@ifQueerEOL{\StraightEOL\printindex\QueerEOL}%
5649   {\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

```

\foone{\makeother\<\makeother\>
5721   \glet\sgtleftxii=<}
5722 {
\gmd@docstripdirective{%
5723   \begingroup\let\do=\makeother
5724   \do\*\do\/\do\+\do\-\do\,\do\&\do\|\do\!\do\(\do\)\do\>\do\<%
5725   \ifnextchar<{}{%
5726     \let\do=\makeother\dospecials
5727     \gmd@docstripverb}
5728   {\gmd@docstripinner}}%}

\gmd@docstripinner{%
5733   \begingroup
5734   \def\gmd@modulehashone{%
5735     \Module{\#1}\space
5736     \@afternarrgfalse\@aftercodegtrue\@codeskipputgfalse}%
5737 }

```

5739 \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 TeX code scanner. Then launch this big TeX 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.

```
5762 }
5764 \foone{\@makeother\<\@makeother\>
5765   \glet\sgtleftxii=<
5766   \catcode`^\^M=\active}%
5767 {
\gmd@docstripverb 5768 \def\gmd@docstripverb<#1^\^M{%
5769   \endgroup%
\gmd@modulehashone 5770 \def\gmd@modulehashone{%
5771   \ModuleVerb{#1}\@afternarrgfalse\@aftercodegtrue%
5772   \@codeskipputgfalse}%
5773 \gmd@docstripshook%
5774 \gmd@textEOL\gmd@modulehashone^\^M}%
5775 }

(–Verbatim ;-) from doc:
\Module 5778 \providecommand*\Module[1]{{\mod@math@codes$\langle\mathsf{#1}\%%
\mathsf{#1}\rangle$}}
\ModuleVerb 5780 \providecommand*\ModuleVerb[1]{{\mod@math@codes$\langle\mathsf{#1}\%%
\mathsf{#1}\rangle$}}
\mod@math@codes 5782 \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 ommit 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 5828 \def\gmd@DefineChanges{%
\changes 5829   \outer\long\def\changes{\@bsphack\begingroup\@sanitize
5830     \catcode`\\z@\catcode`\_10\MakePercentIgnore
5831     \MakePrivateLetters\StraightEOL
5832     \MakeGlossaryControls
5833     \changes@}}
\changes 5835 \newcommand\changes[4][]{\PackageWarningNoLine{gmdoc}{%
5836   ^^JThe \bslash\changes command used on@line
5837   ^^Jwith no \string\RecordChanges space declared.
5838   ^^JI shall not warn you again about it}%
\changes 5840 \renewcommand\changes[4][]{%
5841 }
\MakeGlossaryControls 5843 \def\MakeGlossaryControls{%
5844   \edef\actualchar{\string=}\edef\quotechar{\string!}%
5845   \edef\levelchar{\string>}\edef\encapchar{\string\{}% 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@ 5851 \newcommand\changes@[4][\generalname]{%
5854   \if@RecentChange{#3}% if the date is later than the one stored in \c@Changes-
      % StartDate,
5856   \atempswafalse
5857   \ifx\generalname#1% then we check whether a cs-entry is given in the op-
      tional first argument or is it unchanged.
5859   \ifx\last@defmark\relax\else% if no particular cs is specified in #1, we
      check whether \last@defmark contains something and if so, we put
      it into \gmu@tempb scratch macro.
5862   \atempswatrue
5863   \edef\gmu@tempb{%
      it's a bug fix: while typesetting traditional .dtxes,
      \% \last@defmark came out with \ at the beginning (which re-
      sulted with \\<name> in the change log) but while typesetting the
      'new' way, it occurred without the bslash. So we gobble the bslash
      if it's present and two lines below we handle the exception of
      \% \last@defmark = {} (what would happen if a definition of \\
      was marked in new way gmdocing).
      \if\bslash\last@defmark\else\last@defmark\fii%
5872   \ifx\last@defmark\bslash\let\gmu@tempb\last@defmark\fii%
5873   \n@melet{gmd@glossCSTest}{gmd/isaCS/\last@defmark}%
5874   \fi
5875   \else% the first argument isx not \generalname i.e., a particular cs is specified
      by it (if some day one wishes to \changes \generalname, she should
      type \changes[\generalname]...)
5879   \atempswatrue
5880   {\escapechar\m@ne
5881     \xdef\gmu@tempb{\string#1}}%
```

```

5882      \if\bslash\@xa\@firstofmany\string#1\relax\@nil% we check
5883          whether #1 is a cs...
5884          \def\gmd@glossCStest{1}%
5885          ... and tell the glossary if so.
5886          \fi
5887          \fi
5888          \@ifundefined{gmd@glossCStest}{\def\gmd@glossCStest{o}}{}%
5889          \protected@edef\gmu@tempa{\@nx\gmd@glossary}%
5890          \if\relax\GeneralName\relax\else
5891              \GeneralName% it's for the \DocInclude case to precede every \changes
5892                  of the same file with the file name, cf. line 6334.
5893          \fi
5894          \#2\levelchar%
5895          \if@tempswa% If the macro \last@defmark doesn't contain any cs name
5896              (i.e., is empty) nor #1 specifies a cs, the current changes entry was
5897              done at top-level. In this case we precede it by \generalname.
5898          \gmu@tempb
5899          \actualchar\bslash_verb*%
5900          \if\verbatimchar\gmu@tempb$\else\verbatimchar\fi
5901          \if\gmd@glossCStest\quotechar\bslash\fi\gmu@tempb
5902          \if\verbatimchar\gmu@tempb$\else\verbatimchar\fi
5903          \else
5904              \space\actualchar\generalname
5905          \fi
5906          :\levelchar%
5907          #4%
5908          }%
5909          \gmu@tempa
5910          \grelaxen\gmd@glossCStest
5911          \fi% of \if@recentchange
5912          \endgroup\@esphack}

```

Let's initialize \last@defmark and \GeneralName.

```

5913      \relaxen\last@defmark
5914      \emptyify\GeneralName
5915      \def\ChangesGeneral{\grelaxen\last@defmark}% If automatic detection of def-
5916          definitions is on, the default entry of \changes is the meaning of \last@defmark,
5917          the last detected definiendum that is. The declaration defined here serves to
5918          start a scope of 'general' \changes' entries.
5919      \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 o 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.

```

5920      \c@ChangesStartDate
5921      \newcount\c@ChangesStartDate
5922      \if@RecentChange
5923          \def\if@RecentChange#1{%
5924              \gmd@setChDate#1\@nil\@tempcnta
5925              \ifnum\@tempcnta>\c@ChangesStartDate}
5926      \def\gmd@setChDate#1/#2/#3\@nil#4{%
5927          the last parameter will be a \count
5928          register.
5929          #4=#1\relax

```

```

5946 \multiply#4\by\@M
5947 \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_ (\@te-
      % mpcnta) and some higher (not a scratch) one.
5951 \multiply\count8\by100%
5952 \advance#4\by\count8\count8=\z@
5953 \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 5959 \def\ChangesStart#1#2{%
  5960   \gmd@setChDate#2\@nil\c@ChangesStartDate
  5961   \typeout{^^JPackage\gmdoc_info:^^JChanges' start_date#1%
             memorized
  5962     as\string<\the\c@ChangesStartDate\string>\on@line.^^J}
  5963   \advance\c@ChangesStartDate\m@ne% we shall show the changes at the speci-
             fied day and later.
  5964   \ifnum\c@ChangesStartDate>19820900%10 see below.
  5965     \edef\gmu@tempa{%
  5966       @nx\g@addto@macro@nx\glossary@prologue{%
  5967         The_changes
  5968         \if\relax\GeneralName\relax\else_of_\GeneralName\space\fi
  5969         earlier_than
  5970         #1\if\relax#1\relax#2\else(#2)\fi\space_are_not_
             shown.}%
  5971   \gmu@tempa
  5972 }
  5973 }
```

(Explanation to line 5967.) 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: he 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 5967 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 macroname (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 6016 \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.

¹⁰ DEK writes in *TEX, The Program* of September 1982 as the date of *TEX* Version 0.

```
\RecordChanges 6028 \def\RecordChanges{\makeglossary\gmd@DefineChanges
6029   \relax\endgroup}
```

“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 6041 \newdimen\GlossaryMin           \GlossaryMin      = 8pt
\c@GlossaryColumns 6043 \newcount\c@GlossaryColumns \c@GlossaryColumns = 2
```

“The environment `theglossary` is defined in the same manner as the `theindex` environment.”

```
theglossary 6049 \newenvironment{theglossary}{%
6050   \begin{multicols}{\c@GlossaryColumns}
6051     [\glossary@prologue] [\GlossaryMin]%
6052     \GlossaryParms\IndexLinksBlack
6053     \let\item\@idxitem\ignorespaces}%
6054   {\end{multicols}}
```

Here is the `MakeIndex` style definition:

```
6060 </package>
6061 <+gmglo> preamble
6062 <+gmglo> "\n\begin{theglossary}\n
6063 <+gmglo> \\makeatletter\n"
6064 <+gmglo> postamble
6065 <+gmglo> "\n\n\end{theglossary}\n"
6066 <+gmglo> keyword"\glossaryentry"
6067 <+gmglo> actual='
6068 <+gmglo> quote'!
6069 <+gmglo> level '>
6070 <*package>
```

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 6089 \long\def\GlossaryPrologue#1{\@bsphack
6090   \def\glossary@prologue{#1}%
6091   \@esphack}
```

“Now we test whether the default is already defined by another package file. If not we define it.”

```
\glossary@prologue 6096 \@ifundefined{glossary@prologue}
6097   {\def\glossary@prologue{\indexdiv{{ChangeHistory}}\%
6098   \markboth{{ChangeHistory}}{{ChangeHistory}}\%
6099   }}%
```

“Unless the user specifies otherwise, we set the change history using the same parameters as for the index.”

```
6103 \AtBeginDocument{%
6104   \@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 6116 \def\PrintChanges{%
6117   \@ifQueerEOL
6118   { \StraightEOL \input{\jobname.gls} \QueerEOL }%
6119   { \input{\jobname.gls} }%
6120   \g@emptyify\PrintChanges }
```

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 6157 \newcount\check@sum
\CheckSum 6159 \def\CheckSum#1{\@bsphack\global\check@sum#1\relax\@esphack}
CheckSum 6161 \newcounter{CheckSum}
\step@checksum 6164 \newcommand*\step@checksum{\stepcounter{CheckSum}}
```

And we’ll use it in the line 3538 (`\stepcounter` is `\global`). See also the `\chschange` declaration, l. 6245.

However, the check sum mechanism in `gmdoc` behaves slightly different than in `doc` which is nicely visible while `gmdoc`ing `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.)¹¹

“But `\Finale` will be called at the very end of a file. This is exactly the point were 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.

```
6191 \AtEndInput{\check@checksum}
```

Based on the lines 723–741 of `doc.dtx`.

```
\check@checksum 6194 \def\check@checksum{\relax
6195   \ifnum\check@sum=\z@
```

¹¹ 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`.

```

6196 \edef\gmu@tempa{\% why \edef—see line 6224
6197   \@nx\typeout{*****\gmd@inputname\space has no Checksum
6198     * The input file \gmd@inputname\space has no Checksum
6199     stated.\^J%
6200     * The current checksum is \the\c@CheckSum.\^J%
6201     \gmd@chschangeline% a check sum changes history entry, see below.
6202     * (package\gmdoc\info.)\^J%
6203     *****\^J\}%
6204 
6205 \else
6206 \ifnum\check@sum=\c@CheckSum
6207   \edef\gmu@tempa{%
6208     \@nx\typeout{*****\gmd@inputname:\_Checksum\_passed.\^J%
6209       * The input file \gmd@inputname:\_Checksum\_passed.\^J%
6210       \gmd@chschangeline
6211       * (package\gmdoc\info.)\^J%
6212       *****\^J\}%
6213 
6214   \edef\gmu@tempa{%
6215     \@nx\typeout{*****!*!*!*!*!*!*!*!*!*!*!*!\^J%
6216       *! The input file \gmd@inputname:\^J%
6217       *! CheckSum stated: \the\check@sum\space<>\_my
6218       count: \the\c@CheckSum.\^J%
6219       \gmd@chschangeline
6220       *! (package\gmdoc\info.)\^J%
6221       *****!*!*!*!*!*!*!*!*!*!\^J\}%
6222 
6223 \fi
6224 \gmu@tempa
6225 \AtEndDocument{\gmu@tempa}% we print the checksum notification
6226   on the terminal immediately and at end of TeXing not to have to scroll the
6227   output far nor search the log.
6228 \global\check@sum\z@}

```

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 6233 \def\gmd@chschangeline{%
6234   \xiipercent\space\string\chschange
6235   {\@ifndef{fileversion}{v???\{\fileversion\}}%
6236   {\the\year/\the\month/\the\day}%
6237   {\the\c@CheckSum}\^J%
6238   \xiipercent\space\string\chschange
6239   {\@ifndef{fileversion}{v???\{\fileversion\}}%
6240   {\@xa\@gobbletwo\the\year/\the\month/\the\day}%
6241   { \% with two digit year in case you use \ChangesStart.
6242     \the\c@CheckSum}\^J\}

```

And here the meaning of such a line is defined:

```

\chschange 6245 \newcommand*\chschange[3]{%
6246   \csname\changes\endcsname{\#1}{\#2}{\CheckSum\#3}\%\csname...
6247   % \changes is \outer.
6248   \CheckSum\#3\}

```

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 6270 \newcommand*\ltxPageLayout{%
```

“Increase the text width slightly so that width the standard fonts 72 columns of code may appear in a macrocode environment.”

```
6274 \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 gmdocc.cls) we replace the original \addtolengths with \setlengths.

```
6284 \setlength\marginparwidth{95pt}%
6285 \setlength\oddsidemargin{82pt}%
6286 \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 titlepages 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 6318 \newcommand*\DocInclude{\bgroup\@makeother\_ \Doc@Include}%
           First, we
           make _ ‘other’ in order to allow it in the filenames.

\Doc@Include 6321 \newcommand*{\Doc@Include}[2] [] {%
           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.

           6326 \egroup% having the arguments read, we close the group opened by the previous
           macro for _12.
```

```
\HLPrefix 6328 \gdef\HLPrefix{\filesep}%
6329 \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 6334 \def\GeneralName{\#2\actualchar\pk{\#2}\_}%
           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`.

```

6341 \edef\gmd@jobname{\jobname}%
6342 \edef\gmd@filename{\% we want the filename all ‘other’, just as in \jobname.
6343   \cxa\cxa\cxa\gobble\cxa\string\csname#2\endcsname}%
6344 \ifx\gmd@jobname\gmd@filename
6345   \def\gmd@auxext{auxx}%
6346 \else
6347   \def\gmd@auxext{aux}%
6348 \fi
6349 \relax
6350 \clearpage
6351 \gmd@docincludeaux
6352 \def\currentfile{gmtdoc-IncludeFileNotFound.ooo}%
6353 \let\fullcurrentfile\currentfile
6354 \IfFileExists{#1#2.fdd}{\edef\currentfile{#2.fdd}}{\% it's not .fdd,
6355   \IfFileExists{#1#2.dtx}{\edef\currentfile{#2.dtx}}{\% it's not
6356     either,
6357     \IfFileExists{#1#2.sty}{\edef\currentfile{#2.sty}}{\% it's not .sty,
6358       \IfFileExists{#1#2.cls}{\edef\currentfile{#2.cls}}{\% it's not
6359         .cls,
6360         \IfFileExists{#1#2.tex}{\edef\currentfile{#2.tex}}{\% it's not
6361           .tex,
6362           \IfFileExists{#1#2.fd}{\edef\currentfile{#2.fd}}{\% so it
6363             must be .fd or error.
6364             \PackageError{gmtdoc}{\string\DocInclude\space_file
6365               #1#2.fdd/dtx/sty/cls/tex/fd_not_found.}}}}}}%
6366 \edef\fullcurrentfile{#1\currentfile}%
6367 \ifnum@\auxout=\@partaux
6368   \@latexerr{\string\DocInclude\space_cannot_be_nested}\@eha
6369 \else@\docinclude{#1}#2\fi% Why is #2 delimited with _ not braced as
6370   we are used to, one may ask.
6371 \def\@docinclude#1#2{\% To match the macro’s parameter string, is an answer.
6372   But why is \@docinclude defined so? Originally, in ltxdoc it takes one argument
6373   and it’s delimited with a space probably in resemblance to the true
6374   \input (\@input in LATEX).
6375 \clearpage
6376 \if@files_w\gmd@writemauxinpaux{#2.\gmd@auxext}\fi% this strange macro
6377   with a long name is another thing to allow _ in the filenames (see line 6449).
6378 \tempswattrue
6379 \if@partsw\@tempswafalse\edef\gmu@tempb{#2}%
6380   \@for\gmu@tempa:=\partlist\do{\ifx\gmu@tempa\gmu@tempb%
6381     \tempswattrue\fi}%
6382 \fi
6383 \if@tempswa\let\@auxout\@partaux
6384   \if@files_w
6385     \immediate\openout\@partaux#2.\gmd@auxext\relax% Yes, only #2.
6386     It’s to create and process the partial .aux(x) files always in the main
6387     document’s (driver’s) directory.
6388     \immediate\write\@partaux{\relax}%
6389   \fi
6400 \fi

```

"We need to save (and later restore) various index-related commands which might be changed by the included file."

```

6410  \StoringAndRelaxingDo\gmd@doIndexRelated
6411  \if@ltxDocInclude\part{\currentfile}%
6412    In the ltxdoc-like setup we make
6413      a part title page with only the filename and the file's \maketitle will
6414      typeset an article-like title.
6415  \else\let\maketitle=\InclMaketitle
6416  \fi% In the default setup we redefine \maketitle to typeset a common chapter
6417      or part heading.
6418  \if@ltxDocInclude\xdef@filekey\fi
6419  \GetFileInfo{\currentfile}%
6420    it's my (GM) addition with the account of
6421      using file info in the included files' title/heading etc.
6422  \incl@DocInput{\fullcurrentfile}%
6423    originally just \currentfile.
6424  \if@ltxDocInclude\else\xdef@filekey\fi%
6425    in the default case we add
6426      new file to the file key after the input because in this case it's the files
6427      own \maketitle what launches the sectioning command that increases
6428      the counter.

```

And here is the moment to restore the index-related commands.

```

6427  \RestoringDo\gmd@doIndexRelated
6428  \clearpage
6429  \gmd@writeckpt{\#1#2}%
6430  \if@files_w\immediate\closeout\@partaux\fi
6431  \else\@nameuse{cp@\#1#2}%
6432  \fi
6433  \let\@auxout\@mainaux}%
6434  end of \@docinclude.

```

(Two is a sufficient number of iterations to define a macro for.)

```

\xdef@filekey 6439 \def\xdef@filekey{{\relaxen\ttfamily}%
6440   This assignment is very trickly crafted:
6441   it makes all \ttfamily's present in the \filekey's expansion unexpandable
6442   not only the one added in this step.
6443   \xdef\filekey{\filekey,\thefilediv={\ttfamily}%
6444     \currentfile}}}

```

To allow `_` in the filenames we must assure `_` will be `_12` while reading the filename. Therefore define

```

\gmd@writemauxinpaux 6449 \def\gmd@writemauxinpaux#1{%
6450   this name comes from 'write outto main .aux to
6451   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 localized 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 `gmutils` where they are a bit more explained.)

```

6461  \immediate\write\@mainaux{%
6462    \bgroup\string\@makeother\string\_
6463    \string\firstofone{\egroup
6464    \string\@input{\#1}}}}

```

We also slightly modify a `\LaTeX` kernel macro `\@writeckpt` to allow `_` in the file name.

```

\gmd@writeckpt 6471 \def\gmd@writeckpt#1{%
6472   \immediate\write\@partaux{%

```

```

6473   \string\bgroup\string\@makeother\string\%
6474   \string\firstofone\@charl\string\egroup}
6475   \@writeckpt{\#1}%
6476   \immediate\write\@partaux{\@charrb}}
\gmd@doIndexRelated 6478 \def\gmd@doIndexRelated{%
6479   \do\tableofcontents\do\makeindex\do\EnableCrossrefs
6480   \do\PrintIndex\do\printindex\do\RecordChanges\do%
6481   \PrintChanges
6482   \do\theglossary\do\endtheglossary}
6484 \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

```

\alph 6488 \def\alph{\@alph{\csname c@\#1\endcsname}}
\@alph 6489 \def\@alph#1{%
6490   \ifcase#1\or\@a\or\@b\or\@c\or\@d\or\@e\or\@f\or\@g\or\@h\or\@i\or
6491   \j\or\@k\or\@l\or\@m\or\@n\or\@o\or\@p\or\@q\or\@r\or\@s\or
6492   \t\or\@u\or\@v\or\@w\or\@x\or\@y\or\@z\or\@A\or\@B\or\@C\or
6493   \D\or\@E\or\@F\or\@G\or\@H\or\@I\or\@J\or\@K\or\@L\or\@M\or
6494   \N\or\@O\or\@P\or\@Q\or\@R\or\@S\or\@T\or\@U\or\@V\or\@W\or
6495   \X\or\@Y\or\@Z\else\@ctrerr\fi}

```

A macro that initialises things for `\DocInclude`.

```
\gmd@docincludeaux 6498 \def\gmd@docincludeaux{%
```

We set the things for including the files only once.

```
6500 \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.

```

6504 \ifx\filediv\relax
6505   \ifx\filedivname\relax% (nor \filediv neither \filedivname is defined
6506     by the user)
6507     \@ifundefined{chapter}{%
6508       \SetFileDiv{part}}%
6509     {\SetFileDiv{chapter}}%
6510   \else% (\filedivname is defined by the user, \filediv is not)
6511     \SetFileDiv{\filedivname}% why not? Inside is \edef so it'll work.
6512   \fi
6513 \else% (\filediv is defined by the user
6514   \ifx\filedivname\relax% and \filedivname is not)
6515     \PackageError{gmdoc}{You've redefined \string\filediv\space
6516     without redefining \string\filedivname.}{Please redefine
6517     the
6518     two macros accordingly. You may use \string\SetFileDiv{%
6519       name
6520       without\bslash}.}%
6521   \fi
6522 \fi
6523 \def\thefilediv{\alph{\filedivname}}% The files will be numbered with
6524 letters, lowercase first.
6525 \xa\let\csname\the\filedivname\endcsname=\the\filediv% This line lets
6526 \the<chapter> etc. equal \the\filediv.

```

```

\filesep 6539 \def\filesep{\thefilediv-}% File separator (identifier) for the index.
6540 \let\filekey=\@gobble
6541 \g@addto@macro@index@prologue{%
6542   \gdef\@oddfoot{\parbox{\textwidth}{\strut\footnotesize
6543     \raggedright\bfseries\thefilekey}}% The footer for the
6544     pages of index.
6545   \glet\@evenfoot\@oddfoot}% anyway, it's intended to be oneside.
6546 \g@addto@macro@glossary@prologue{%
6547   \gdef\@oddfoot{\strut\ChangeHistory\hfill\thepage}}% The footer for
6548     the changes history.
6549   \glet\@evenfoot\@oddfoot}%
6550 \gdef\@oddfoot{%
6551   \thefilekey The footer of the file pages will be its name and, if there is
6552   a file info, also the date and version.
6553   \relax
6554   \xa\ifx\csname_ver@\currentfile\endcsname\relax
6555     \thefilekey{\ttfamily\currentfile}%
6556   \else
6557     \GetFileInfo{\currentfile}%
6558     \thefilekey{\ttfamily\filename}%
6559     Date:\filedate\%
6560     Version:\fileversion
6561   \fi
6562   \hfill\thepage}%
6563 \glet\@evenfoot\@oddfoot% see line 6545.
6564 \xa\def\csname\filedivname\name\endcsname{File}}% we redefine the name
6565     of the proper division to 'File'.
6566 \ifx\filediv\section
6567   \let\division=\subsection
6568   \let\subdivision=\subsubsection
6569   \let\subsubdivision=\paragraph
6570
6571 \fi}%
6572 \fi}%
6573 \fi}%
6574 \fi}%
6575 \fi}%
6576 \fi}%
6577 \fi}%
6578 \fi}%
6579 \fi}%

```

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.

6579 \fi} end of \gmd@docincludiaux.

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 6584 \def\SetFileDiv#1{%
6585   \edef\filedivname{#1}%
6586   \xa\let\@xa\filediv\csname#1\endcsname}

```

```

\SelfInclude 6590 \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 she will prefer what I offer, gmdocc.cls e.g.), so we put those preparations into a declaration.

```

\if@ltxDocInclude 6603 \newif\if@ltxDocInclude
\ltxLookSetup 6605 \newcommand*\ltxLookSetup{%
6606   \SetFileDiv{part}%
6607   \ltxPageLayout
6608   \ltxDocIncludetrue
6609 }

```

```

6611 \onlypreamble\ltxLookSetup
The default is that we \DocInclude the files due to the original gmdoc input settings.
6615 \let\incl@DocInput=\DocInput
6617 \emptyify\currentfile% for the pages outside the \DocInclude's scope. In force
for all includes.

```

If you want to \Doc/SelfInclude doc-likes:

```
\olddocIncludes 6637 \newcommand*\olddocIncludes{%
6638   \let\incl@DocInput=\OldDocInput}
```

And, if you have set the previous and want to set it back:

```
\gmdocIncludes 6641 \newcommand*\gmdocIncludes{%
6642   \let\incl@DocInput=\DocInput
6643   \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.

```

6661 \AtBeginInput{%
\maketitle 6662   \providecommand*\maketitle{\par
6663     \begingroup\def\thefootnote{\fnsymbol{footnote}}%
6664     \setcounter{footnote}\z@%
6665     \def\@makefnmark{\hbox{to\z@\{$\m@th^{\@thefnmark} $\hss}}%
6666     \long\def\@makefntext##1{\parindent\em\noindent
6667       \hbox{to1.8em{$\hss$\m@th^{\@thefnmark}##1}}%
6668     \if@twocolumn\twocolumn[\@maketitle]%
6669     \else\newpage\global\@topnum\z@\@maketitle\fi

```

“For special formatting requirements (such as in tugboat), we use pagestyle titlepage for this; this is later defined to be plain, unless already defined, as, for example, by ltugboat.sty.”

```
6674   \thispagestyle{titlepage}\@thanks\endgroup
```

“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:”

```

6678   \setcounter{footnote}\z@
6679   \gdef\@date{\today}\g@emptyify\@thanks%
6680   \g@emptyify\@author\g@emptyify\@title%
6681 }%
```

“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 tugboat can define this macro in advance. However, if no such definition exists, we use pagestyle plain for title pages.”

```
6688 \@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`.

```

6693 \providecommand*\@maketitle{%
6694   \newpage\null\vskip.2em\relax%
6695   \begin{center}%
6696     \titlesetup
6697     \let\footnote\thanks
6698     {\LARGE\@title\par}%
6699     \vskip.15em%
6700     {\large\lineskip.5em%
6701       \begin{tabular}[t]{c}%
6702         \strut\author
6703       \end{tabular}\par}%
6704     \vskip.1em%
6705     {\large\@date}%
6706   \end{center}%
6707   \par\vskip.15em\relax}%

```

We'd better restore the primary meanings of the macros making a title. ($\text{\LaTeX}_2\epsilon$ source, File F: `ltsect.dtx` Date: 1996/12/20 Version v1.0z, lines 3.5.7.9–12.14–17.)

```

\title 6711 \providecommand*\title[1]{\gdef\@title{\#1}}
\author 6712 \providecommand*\author[1]{\gdef\@author{\#1}}
\date 6713 \providecommand*\date[1]{\gdef\@date{\#1}}
\thanks 6714 \providecommand*\thanks[1]{\footnotemark
  \protected@xdef\@thanks{\@thanks
  \protect\footnotetext[\the\c@footnote]{\#1}}%
  }%
6717 }%
\and 6718 \providecommand*\and{%
  \begin{tabular}{c}
  \end{tabular}%
  \hspace{1em}\@plus.17fil%
  \begin{tabular}[t]{c}%
  \end{tabular}%
} And finally, let's initialize
\titlesetup 6719 \titlesetup if it is not yet.
6720 \providecommand*\titlesetup{}%
6721 }% 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 initialize the file division macros.

```

6738 \@relaxen\filediv
6739 \@relaxen\filedivname
6740 \@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 6750 \def\InclMaketitle{%
\and 6753   {\def\and{,} we make \and just a comma.
6754     {\let\thanks=\@gobble% for the toc version of the heading we discard \thanks.

```

```

6756     \protected@xdef\incl@titletotoc{\@title\if@fshda\protect%
6757         \space
6758     (\@author)\fi}% we add the author iff the 'files have different authors'
6759     % (\@fshda)
6760 }%
6761 \def\thanks##1{\footnotemark
6762     \protected@xdef\@thanks{\@thanks% to keep the previous \thanks if
6763         there were any.
6764     \protect\footnotetext[\the\c@footnote]{##1}}% for some mys-
6765         terious reasons so defined \thanks do typeset the footnote mark
6766         and text but they don't hyperlink it properly. A hyperref bug?
6767     \g@empty\@thanks
6768     \protected@xdef\incl@filedivtitle{%
6769         [{\incl@titletotoc}]}% braces to allow [ and ] in the title to toc.
6770     \protect\@title
6771         {\smallerr% this macro is provided by the gutils package after the rel-
6772             size package.
6773             \if@fshda\relax[0.15em]\protect\@author
6774                 \if\relax\@date\relax\else,\fi
6775             \else
6776                 \if\relax\@date\relax\else\relax[0.15em]\fi
6777             \fi
6778         }

```

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@fshda switch defined in line 6809.

If we wish to print the author's name (\if@fshda), 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.

```

6790         \protect\@date}}% end of \incl@filedivtitle's brace (2nd or 3rd
6791             argument).
6792     }% 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).

```

6793     }% end of the comma-\and's group.
6794     \gxa\filediv\incl@filedivtitle
6795     \@thanks
6796     \g@relaxen\@author\g@relaxen\@title\g@relaxen\@date
6797     \g@empty\@thanks
6800 }% 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@fshda 6809 \newif\if@fshda
            (its name comes from files have different authors).
\PrintFilesAuthors 6813 \newcommand*\PrintFilesAuthors{\@fshdatrue}
                    And the counterpart, if you change your mind:
\SkipFilesAuthors 6815 \newcommand*\SkipFilesAuthors{\@fshdafalse}

```

The file's date and version information

Define \filedate and friends from info in the \ProvidesPackage etc. commands.

```
\GetFileInfo 6823 \def\GetFileInfo#1{%
  \filename 6824 \def\filename{\#1}%
  \gmu@tempb 6825 \def\gmu@tempb##1##2##3\relax##4\relax{%
    \filedate 6826 \def\filedate{\##1}%
    \fileversion 6827 \def\fileversion{\##2}%
    \fileinfo 6828 \def\fileinfo{\##3}%
  6829 \edef\gmu@tempa{\csname ver@\#1\endcsname}%
  6830 \xa\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 - \TeX provides a special primitive for that action but we remain strictly \TeX nical and do it with writing to a file and inputting that file.

```
\FileInfo 6841 \newcommand*\FileInfo{%
  6842   \bgroup
  6843   \gmd@ctallsetup
  6844   \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).
  6848   \let\do\@makeother
  6849   \do\do\{\do\}\do\^\^M\do\\%
  6850   \gmd@fileinfo}

  6853 \foone{%
  6854   \catcode`!\z@
  6855   \catcode`(\@ne
  6856   \catcode`)\tw@
  6857   \let\do\@makeother
  6858   \do\% we make space 'other' to keep it for scanning the code where it may be
             leading.
  6860   \do\{\do\}\do\^\^M\do\\%
  6861 (%
\gmd@fileinfo 6862 !def!gmd@fileinfo#1Provides#2{\#3}#4[\#5]#6^\^M%
  6863 (!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.
  6866 !gmd@writeFI(#2)(#3)(#5)%
  6867 !gmd@FIrescan(#1Provides#2{\#3}#4[\#5]#6)% this macro will close the group.
  6872 )%
  6873 )

\gmd@writeFI 6875 \def\gmd@writeFI#1#2#3{%
  6877   \immediate\write\@auxout{%
  6878     \global\@nx\@namedef{%
  6879       ver@\#2.\ifP@\firstofmany#1\@nil\sty\else\cls\fi\{\#3\}}}
  6881 \foone\obeylines{%
\gmd@FIrescan 6882   \def\gmd@FIrescan#1{%
  6887   {\newlinechar`\^\^M\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 2_c, file L v1.1g, l. 102.

```
\ProvideFileInfo 6895 \def\ProvideFileInfo#1{%
 6896   \begingroup
 6897     \catcode`\_10\catcode\endlinechar_10%
 6898     \@makeother`/\@makeother`&%
 6899     \kernel@ifnextchar[{\gmd@providefii{#1}}{\gmd@providefii{#1}[]}%
 6900   }
\gmd@providefii 6904 \def\gmd@providefii#1[#2]{%
 6905   (we don't write the file info to .log)
 6906   \xa\xdef\csname_ver@#1\endcsname{#2}%
 6907   \endgroup}
```

And a self-reference abbreviation (intended for providing file info for the driver):

```
\ProvideSelfInfo 6911 \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 6915 \newcommand*\filenote[This_file_has_version_number_\fileversion{%
 6916   }_dated_\filedate{}.]
```

And exactly as \thanks:

```
\thfileinfo 6917 \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:

```
6928 \foone{\obeylines}%
6929 {%
\IndexInput 6930 \def\IndexInput#1{%
 6931   \StoreMacro\code@delim%
 6932   \CodeDelim\^\^Z%
\gmd@iihook 6935 \def\gmd@iihook{%
 6936   this hook is \edefed!
 6937   \code@delim\relax\@nx\let\@nx\EOFMark\relax}%
 6938   \DocInput{#1}\RestoreMacro\code@delim}%
 6939 }
```

How does it work? We assume in the input file is no explicit *char1*. 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 \skipgmlonly declaration's scope it gobbles its contents.

```
gmlonly 6955 \newenvironment{gmlonly}{\StraightEOL}{}
\skipgmlonly 6957 \newcommand\skipgmlonly[1][]{%
 6958   \def\gmu@tempa{%
 6959     \def\gmd@skipgmltext{%
 6960       \g@emptyify\gmd@skipgmltext
```

```

6962      #1%
6963      }% not to count the lines of the substituting text but only of the text omitted
6965      \gmu@tempa
6966      \@xa\AtBeginInput\@xa{\gmu@tempa}%
6967      \renewenvironment{gmlonely}{%
6968          \StraightEOL
6969          \@fileswfalse% to forbid writing to .toc, .idx etc.
6970          \setboxo=\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.

```

6977 \foone{\catcode`\_=_8}% we ensure the standard \catcode of _.
6978 {
6979     \newcommand*\catother{${}_{12}}%

```

Similarly, if we need to say some char is of category 13 ('active'), we'll write 13, got by \catactive

```

\catactive 6982 \newcommand*\catactive{${}_{13}}%
             and a letter, 11

```

```

\catletter 6984 \newcommand*\catletter{${}_{11}}%.
6985 }

```

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 6990 \newenvironment*{copyrnote}{%
6991     \StraightEOL\everypar{\hangindent3em\relax\hangafter1}%
6992     \par\addvspace\medskipamount\parindent\z@\obeylines}%
6993     \@codeskipputgfalse\stanz{a}

```

I renew the quotation environment to make the fact of quoting visible.

```

\gmd@quotationname 6997 \StoreEnvironment{quotation}
quotation 6998 \def\gmd@quotationname{quotation}
6999 \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.

```

7006 \ifx\@currenvir\gmd@quotationname
7007 \afterfi{\par``\ignorespaces}%
7008 \else\afterfi{\storedcsname{quotation}}%
7009 \fi%
7010 {\ifx\@currenvir\gmd@quotationname
7011 \afterfi{\ifhmode\unskip\fi'\'\par}%
7012 \else\afterfi{\storedcsname{endquotation}}%
7013 \fi}

```

For some mysterious reasons `\noindent` doesn't work with the first (narrative) paragraph after the code so let's work it around:

```

\gmdnoindent 7018 \def\gmdnoindent{%
7019     \ifvmode\leavevmode\hskip-\parindent\ignorespaces
7020     \fi}%
7021 \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 inline 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 7032 \newcommand*\inverb{%
7033   \@ifstar{%
\gmu@tempa 7034     \def\gmu@tempa{{\tt\xiipercent}}%
7035     \@emtify\gmu@tempb% here and in the paralell points of the other case and
7036       \%nlpercent I considered an \ifhmode test but it's not possible to be
7037       in vertical mode while in an inline comment. If there happens vertical
7038       mode, the commentary begins to be 'outline' (main text).
7039     \gmd@inverb}%
7040   {\@emtify\gmu@tempa
\gmu@tempb 7041     \def\gmu@tempb{\gboxedspace}%
7042     \gmd@inverb}%
7043
\gboxedspace 7044 \newcommand*\gboxedspace{\hbox{\normalfont{[ ]}}}
7045
\gmd@nlperc 7046 \newcommand*\gmd@nlperc[1] [] {%
7047   \ifhmode\unskip\fi
7048   \discretionary{\hbox{\gmu@tempa}}%(pre-break). I always put a \hbox here
7049   to make this discretionary score the \hyphenpenalty not \exhyphenpenalty
7050   (The TeXbook p. 96) since the latter may be 10,000 in Polish typesetting.
7051   {\tt\xiipercent\gboxedspace}%(post-break)
7052   {\gmu@tempb}%(no-break).
7053   \penalty1000\hskip0pt\relax}
7054
\gmd@inverb 7055 \newcommand*\gmd@inverb[1] [] {%
7056   \gmd@nlperc
7057   \ifmmode\hbox\else\leavevmode\null\fi
7058   \bgroup
7059   \ttverbatim
\breakablevisspace 7060   \def\breakablevisspace{%
7061     \discretionary{\visiblespace}{\xiipercent\gboxedspace}{%
7062       \visiblespace}}%
7063
\breakbslash 7064 \def\breakbslash{%
7065   \discretionary{}{\xiipercent\gboxedspace\bslash}{\bslash}}%
7066
\breakbrace 7067 \def\breakbrace{%
7068   \discretionary{}{\xiipercent\gboxedspace\brace}{\brace}}%
7069   \gma@verb@eol
7070   \sverb@chbsl% It's always with visible spaces.
7071
7072
7073
7074
7075
7076
7077
7078 }
7079
\nlpercent 7080 \newcommand*\nlpercent{%
7081   \@ifstar{\def\gmu@tempa{{\tt\xiipercent}}%
7082     \@emtify\gmu@tempb
7083     \gmd@nlperc}%
7084   {\@emtify\gmu@tempa
\gmu@tempb 7085     \def\gmu@tempb{\gboxedspace}%
7086     \gmd@nlperc}%
7087
\incs 7088 \newcommand*\incs{%
7089   \ifstar{\def\gmu@tempa{{\tt\xiipercent}}%
7090     \gmu@tempa}%
7091
7092
7093
7094
7095
7096
7097
7098
7099
7100
7101
7102
7103
7104
7105
7106
7107
7108
7109
7110
7111
7112
7113
7114
7115
7116
7117
7118
7119
7120
7121
7122
7123
7124
7125
7126
7127
7128
7129
7130
7131
7132
7133
7134
7135
7136
7137
7138
7139
7140
7141
7142
7143
7144
7145
7146
7147
7148
7149
7150
7151
7152
7153
7154
7155
7156
7157
7158
7159
7160
7161
7162
7163
7164
7165
7166
7167
7168
7169
7170
7171
7172
7173
7174
7175
7176
7177
7178
7179
7180
7181
7182
7183
7184
7185
7186
7187
7188
7189
7190
7191
7192
7193
7194
7195
7196
7197
7198
7199
7200
7201
7202
7203
7204
7205
7206
7207
7208
7209
7210
7211
7212
7213
7214
7215
7216
7217
7218
7219
7220
7221
7222
7223
7224
7225
7226
7227
7228
7229
7230
7231
7232
7233
7234
7235
7236
7237
7238
7239
7240
7241
7242
7243
7244
7245
7246
7247
7248
7249
7250
7251
7252
7253
7254
7255
7256
7257
7258
7259
7260
7261
7262
7263
7264
7265
7266
7267
7268
7269
7270
7271
7272
7273
7274
7275
7276
7277
7278
7279
7280
7281
7282
7283
7284
7285
7286
7287
7288
7289
7290
7291
7292
7293
7294
7295
7296
7297
7298
7299
7300
7301
7302
7303
7304
7305
7306
7307
7308
7309
7310
7311
7312
7313
7314
7315
7316
7317
7318
7319
7320
7321
7322
7323
7324
7325
7326
7327
7328
7329
7330
7331
7332
7333
7334
7335
7336
7337
7338
7339
7340
7341
7342
7343
7344
7345
7346
7347
7348
7349
7350
7351
7352
7353
7354
7355
7356
7357
7358
7359
7360
7361
7362
7363
7364
7365
7366
7367
7368
7369
7370
7371
7372
7373
7374
7375
7376
7377
7378
7379
7380
7381
7382
7383
7384
7385
7386
7387
7388
7389
7390
7391
7392
7393
7394
7395
7396
7397
7398
7399
7399
7400
7401
7402
7403
7404
7405
7406
7407
7408
7409
7410
7411
7412
7413
7414
7415
7416
7417
7418
7419
7420
7421
7422
7423
7424
7425
7426
7427
7428
7429
7430
7431
7432
7433
7434
7435
7436
7437
7438
7439
7440
7441
7442
7443
7444
7445
7446
7447
7448
7449
7450
7451
7452
7453
7454
7455
7456
7457
7458
7459
7459
7460
7461
7462
7463
7464
7465
7466
7467
7468
7469
7469
7470
7471
7472
7473
7474
7475
7476
7477
7478
7479
7479
7480
7481
7482
7483
7484
7485
7486
7487
7488
7489
7489
7490
7491
7492
7493
7494
7495
7496
7497
7498
7499
7499
7500
7501
7502
7503
7504
7505
7506
7507
7508
7509
7509
7510
7511
7512
7513
7514
7515
7516
7517
7518
7519
7519
7520
7521
7522
7523
7524
7525
7526
7527
7528
7529
7529
7530
7531
7532
7533
7534
7535
7536
7537
7538
7539
7539
7540
7541
7542
7543
7544
7545
7546
7547
7548
7549
7549
7550
7551
7552
7553
7554
7555
7556
7557
7558
7559
7559
7560
7561
7562
7563
7564
7565
7566
7567
7568
7569
7569
7570
7571
7572
7573
7574
7575
7576
7577
7578
7578
7579
7579
7580
7581
7582
7583
7584
7585
7586
7587
7588
7589
7589
7590
7591
7592
7593
7594
7595
7596
7597
7598
7598
7599
7599
75100
75101
75102
75103
75104
75105
75106
75107
75108
75109
75109
75110
75111
75112
75113
75114
75115
75116
75117
75118
75119
75119
75120
75121
75122
75123
75124
75125
75126
75127
75128
75129
75129
75130
75131
75132
75133
75134
75135
75136
75137
75138
75139
75139
75140
75141
75142
75143
75144
75145
75146
75147
75148
75149
75149
75150
75151
75152
75153
75154
75155
75156
75157
75158
75159
75159
75160
75161
75162
75163
75164
75165
75166
75167
75168
75169
75169
75170
75171
75172
75173
75174
75175
75176
75177
75178
75178
75179
75179
75180
75181
75182
75183
75184
75185
75186
75187
75188
75189
75189
75190
75191
75192
75193
75194
75195
75196
75197
75198
75198
75199
75199
75200
75201
75202
75203
75204
75205
75206
75207
75208
75209
75209
75210
75211
75212
75213
75214
75215
75216
75217
75218
75219
75219
75220
75221
75222
75223
75224
75225
75226
75227
75228
75229
75229
75230
75231
75232
75233
75234
75235
75236
75237
75238
75239
75239
75240
75241
75242
75243
75244
75245
75246
75247
75248
75249
75249
75250
75251
75252
75253
75254
75255
75256
75257
75258
75259
75259
75260
75261
75262
75263
75264
75265
75266
75267
75268
75269
75269
75270
75271
75272
75273
75274
75275
75276
75277
75278
75278
75279
75279
75280
75281
75282
75283
75284
75285
75286
75287
75288
75289
75289
75290
75291
75292
75293
75294
75295
75296
75297
75297
75298
75298
75299
75299
75300
75301
75302
75303
75304
75305
75306
75307
75308
75309
75309
75310
75311
75312
75313
75314
75315
75316
75317
75318
75319
75319
75320
75321
75322
75323
75324
75325
75326
75327
75328
75329
75329
75330
75331
75332
75333
75334
75335
75336
75337
75338
75339
75339
75340
75341
75342
75343
75344
75345
75346
75347
75348
75349
75349
75350
75351
75352
75353
75354
75355
75356
75357
75358
75359
75359
75360
75361
75362
75363
75364
75365
75366
75367
75368
75369
75369
75370
75371
75372
75373
75374
75375
75376
75377
75378
75379
75379
75380
75381
75382
75383
75384
75385
75386
75387
75388
75389
75389
75390
75391
75392
75393
75394
75395
75396
75397
75397
75398
75398
75399
75399
75400
75401
75402
75403
75404
75405
75406
75407
75408
75409
75409
75410
75411
75412
75413
75414
75415
75416
75417
75418
75419
75419
75420
75421
75422
75423
75424
75425
75426
75427
75428
75429
75429
75430
75431
75432
75433
75434
75435
75436
75437
75438
75439
75439
75440
75441
75442
75443
75444
75445
75446
75447
75448
75449
75449
75450
75451
75452
75453
75454
75455
75456
75457
75458
75459
75459
75460
75461
75462
75463
75464
75465
75466
75467
75468
75469
75469
75470
75471
75472
75473
75474
75475
75476
75477
75478
75479
75479
75480
75481
75482
75483
75484
75485
75486
75487
75488
75489
75489
75490
75491
75492
75493
75494
75495
75496
75497
75497
75498
75498
75499
75499
75500
75501
75502
75503
75504
75505
75506
75507
75508
75509
75509
75510
75511
75512
75513
75514
75515
75516
75517
75518
75519
75519
75520
75521
75522
75523
75524
75525
75526
75527
75528
75529
75529
75530
75531
75532
75533
75534
75535
75536
75537
75538
75539
75539
75540
75541
75542
75543
75544
75545
75546
75547
75548
75549
75549
75550
75551
75552
75553
75554
75555
75556
75557
75558
75559
75559
75560
75561
75562
75563
75564
75565
75566
75567
75568
75569
75569
75570
75571
75572
75573
75574
75575
75576
75577
75578
75579
75579
75580
75581
75582
75583
75584
75585
75586
75587
75588
75589
75589
75590
75591
75592
75593
75594
75595
75596
75597
75597
75598
75598
75599
75599
75600
75601
75602
75603
75604
75605
75606
75607
75608
75609
75609
75610
75611
75612
75613
75614
75615
75616
75617
75618
75619
75619
75620
75621
75622
75623
75624
75625
75626
75627
75628
75629
75629
75630
75631
75632
75633
75634
75635
75636
75637
75638
75639
75639
75640
75641
75642
75643
75644
75645
75646
75647
75648
75649
75649
75650
75651
75652
75653
75654
75655
75656
75657
75658
75659
75659
75660
75661
75662
75663
75664
75665
75666
75667
75668
75669
75669
75670
75671
75672
75673
75674
75675
75676
75677
75678
75679
75679
75680
75681
75682
75683
75684
75685
75686
75687
75688
75689
75689
75690
75691
75692
75693
75694
75695
75696
75697
75697
75698
75698
75699
75699
75700
75701
75702
75703
75704
75705
75706
75707
75708
75709
75709
75710
75711
75712
75713
75714
75715
75716
75717
75718
75719
75719
75720
75721
75722
75723
75724
75725
75726
75727
75728
75729
75729
75730
75731
75732
75733
75734
75735
75736
75737
75738
75739
75739
75740
75741
75742
75743
75744
75745
75746
75747
75748
75749
75749
75750
75751
75752
75753
75754
75755
75756
75757
75758
75759
75759
75760
75761
75762
75763
75764
75765
75766
75767
75768
75769
75769
75770
75771
75772
75773
75774
75775
75776
75777
75778
75779
75779
75780
75781
75782
75783
75784
75785
75786
75787
75788
75789
75789
75790
75791
75792
75793
75794
75795
75796
75797
75797
75798
75798
75799
75799
75800
75801
75802
75803
75804
75805
75806
75807
75808
75809
75809
75810
75811
75812
75813
75814
75815
75816
75817
75818
75819
75819
75820
75821
75822
75823
75824
75825
75826
75827
75828
75829
75829
75830
75831
75832
75833
75834
75835
75836
75837
75838
75839
75839
75840
75841
75842
75843
75844
75845
75846
75847
75848
75849
75849
75850
75851
75852
75853
75854
75855
75856
75857
75858
75859
75859
75860
75861
75862
75863
75864
75865
75866
75867
75868
75869
75869
75870
75871
75872
75873
75874
75875
75876
75877
75878
75879
75879
75880
75881
75882
75883
75884
75885
75886
75887
75888
75889
75889
75890
75891
75892
75893
75894
75895
75896
75897
75897
75898
75898
75899
75899
75900
75901
75902
75903
75904
75905
75906
75907
75908
75909
75909
75910
75911
75912
75913
75914
75915
75916
75917
75918
75919
75919
75920
75921
75922
75923
75924
75925
75926
75927
75928
75929
75929
75930
75931
75932
75933
75934
75935
75936
75937
75938
75939
75939
75940
75941
75942
75943
75944
75945
75946
75947
75948
75949
75949
75950
75951
75952
75953
75954
75955
75956
75957
75958
75959
75959
75960
75961
75962
75963
75964
75965
75966
75967
75968
75969
75969
75970
75971
75972
75973
75974
75975
75976
75977
75978
75979
75979
75980
75981
75982
75983
75984
75985
75986
75987
75988
75989
75989
75990
75991
75992
75993
75994
75995
75996
75997
75997
75998
75998
75999
75999
76000
76001
76002
76003
76004
76005
76006
76007
76008
76009
76009
76010
76011
76012
76013
76014
76015
76016
76017
76018
76019
76019
76020
76021
76022
76023
76024
76025
76026
76027
76028
76029
76029
76030
76031
76032
76033
76034
76035
76036
76037
76038
76039
76039
76040
76041
76042
76043
76044
76045
76046
76047
76048
76049
76049
76050
76051
76052
76053
76054
76055
76056
76057
76058
76059
76059
76060
76061
76062
76063
76064
76065
76066
76067
76068
76069
76069
76070
76071
76072
76073
76074
76075
76076
76077
76078
76079
76079
76080
76081
76082
76083
76084
76085
76086
76087
76088
76089
76089
76090
76091
76092
76093
76094
76095
76096
76097
76097
76098
76098
76099
76099
76100
76101
76102
76103
76104
76105
76106
76107
76108
76109
76109
76110
76111
76112
76113
76114
76115
76116
76117
76118
76119
76119
76120
76121
76122
76123
76124
76125
76126
76127
76128
76129
76129
76130
76131
76132
76133
76134
76135
76136
76137
76138
76139
76139
76140
76141
76142
76143
76144
76145
76146
76147
76148
76149
76149
76150
76151
76152
76153
76154
76155
76156
76157
76158
76159
76159
76160
76161
76162
76163
76164
76165
76166
76167
76168
76169
76169
76170
76171
76172
76173
76174
76175
76176
76177
76178
76179
76179
76180
76181
76182
76183
76184
76185
76186
76187
76188
76189
76189
76190
76191
76192
76193
76194
76195
76196
76197
76198
76198
76199
76199
76200
76201
76202
76203
76204
76205
76206
76207
76208
76209
76209
76210
76211
76
```

```

7091   \emptyify\gmu@tempb
7092   \gmd@nlperc\cs}%
7093 { \emptyify\gmu@tempa
7094   \def\gmu@tempb{\gmbboxedspace}%
7095   \gmd@nlperc\cs}}
\gmu@tempb
\inenv 7097 \def\inenv{\inccs[]}% an in-line \env

```

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.

TODO: make the space flexible (most probably it requires using sth. else than `\discretionary`).

An optional hyphen for cses in the inline comment:

```

7115 \ifundefined{+}{}{\typeout{^^Jgmdoc.sty: redefining \bslash+.}}
\+ 7116 \def\+{\discre{\normalfont}{\tt\xiipercent\gmbboxedspace}{}}}
\ds 7120 \providecommand*\ds{DocStrip}

```

A shorthand for \CS:

```

\CS 7123 \pdef\CS{%
7124   \acro\CS\%
7125   \ifnextcat\aa{}% we put a space if the next token is 11. It's the next best
      thing to checking whether the cs consisting of letters is followed by a space.

\CSS 7129 \pdef\CSS{\CS{}es\ifnextcat\aa{}% for pluralis.
\CSes 7131 \pdef\CSes{\CS{}es\ifnextcat\aa{}% 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 7140 \newcommand*\CDAnd{\CodeDelim\&}
\CDPerc 7142 \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 7150 \let\division=\section
\subdivision 7153 \let\subdivision=\subsection
\subsubdivision 7156 \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 7162 \newcounter{gmd@mc}

Note it is after the macrocode group

\gmd@mchook 7165 \def\gmd@mchook{\stepcounter{gmd@mc}%
7166   \gmd@mcdiag
7167   \ifcsname\gmd@mchook\the\c@gmd@mc\endcsname
7168   \afterfi{\csname\gmd@mchook\the\c@gmd@mc\endcsname}%
7169   \fi}

\AfterMacrocode 7171 \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).

```

7181 \emptyify\gmd@mcdiag
\mcdiagOn 7183 \def\mcdiagOn{\def\gmd@mcdiag{%
7184   \typeout{^^J\bslash_end{\@currenvir}_No.\the\c@gmd@mc
7185   \space\on@line,\_cln.\the\c@codelenum.}}}
\mcdiagOff 7187 \def\mcdiagOff{\emptyify\gmd@mcdiag}

```

An environment to display the meaning of macro parameters: its items are automatically numbered as #1, #2 etc.

```

enumargs 7191 \newenvironment*{enumargs}[1][1]{%
7197   \if@aftercode\edef\gmu@tempa{\the\leftskip}%
7198     \edef\gmu@tempb{\the\hangindent}\fi
7199   \enumerate
7200   \if@aftercode
7201     \leftskip=\glueexpr\gmu@tempa+\gmu@tempb\relax
7202   \fi
7203   \namedef{label}{enumctr}{%
7204     \env{\if@aftercode\code@delim\space\fi
7205       \gmd@ea@bwrap
7206       \#\ifcase#1\relax\or\or\#\or\or\#\#\#\fi
7207       \csname\the\@enumctr\endcsname
7208       \gmd@ea@ewrap}}%
7209   \let\mand\item
7210   \provide\gmd@ea@wraps{%
7211     \emptyify\gmd@ea@ewrap
7212     \emptyify\gmd@ea@bwrap}%
7213   \gmd@ea@wraps
7214   \def\opt{%
7215     \def\gmd@ea@bwrap{} \def\gmd@ea@ewrap{} }%
7216   \item
7217     \gmd@ea@wraps}%
7218 }
7219 {\endenumerate}

```

The starred version is intended for lists of arguments some of which are optional: to align them in line.

```

enumargs* 7223 \newenvironment*{enumargs*}{%
\gmd@ea@wraps 7224 \def\gmd@ea@wraps{%
\gmd@ea@bwrap 7225 \def\gmd@ea@bwrap{}\def\gmd@ea@ewrap{} }%
\gmd@ea@ewrap 7226 \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 2407.

At the very beginning of a doc-prepared file we meet a nice command \CharacterTable. My T_EX Guru says it’s a bit old fashioned these days so let’s just make it notify the user:

```
\CharacterTable 7249 \def\CharacterTable{\begingroup
7250   \@makeother{\@\makeother\}%
7251   \Character@Table}

7253 \foone{%
7254   \catcode`[\=1\catcode`\]=2%
7255   \@makeother{\@\makeother\}%
7256   [
\Character@Table 7257 \def\Character@Table#1{#2}[\endgroup
7258   \message[^^J^J gmdoc.sty package:^^J
7259   ===== The input file contains the \bslash CharacterTable.^^J
7260   ===== If you really need to check the correctness of the
7261   chars,^^J
7262   ===== please notify the author of gmdoc.sty at the email
7263   address^^J
7264   ]]
```

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 7630.

Let us now look over other original doc’s control sequences and let’s ‘domesticate’ them if they are not yet.

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 7284 \outer\def\DescribeMacro{%
7285   \begingroup\MakePrivateLetters
7286   \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 \MakePrivateOthers re\catcodeing shall not be done to it.

```
\DescribeEnv 7291 \outer\def\DescribeEnv{%
7292   \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 7297 \outer\def\Describe{%
7298   \begingroup\MakePrivateLetters
7299   \@ifstarl{\MakePrivateOthers\Describe@Env}{\Describe@Macro}}
```

The below two definitions are adjusted ~s of \Text@UsgMacro and \Text@UsgEnvir.

```
\Describe@Macro 7305 \long\def\Describe@Macro#1{%
7306   \endgroup
7307   \strut\Text@Marginize#1%
7308   \@usgentryze#1% we declare kind of formatting the entry
7309   \text@indexmacro#1\ignorespaces}

\Describe@Env 7312 \def\Describe@Env#1{%
7313   \endgroup
7314   \strut\Text@Marginize{#1}%
7315   \@usgentryze{#1}% we declare the 'usage' kind of formatting the entry and in-
    index the sequence #1.
7317   \text@indexenvir{#1}\ignorespaces}
```

Note that here the environments' names are typeset in \tt 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 T_EX code in \tt font nad this tradition sustained should be. If one wants to change the tradition, let him 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:

```
7332 \let\MacroFont\tt
```

We have provided \CodeIndent in line 2227. And it corresponds with doc's \MacroIndent so

```
\MacroIndent 7340 \let\MacroIndent\CodeIndent
```

And similarly the other skips:

```
7342 \let\MacrocodeTopsep\CodeTopsep
```

\MacroTopsep Note that \MacroTopsep is defined in gmdoc and has the same rôle as in doc.

```
\SpecialEscapechar 7346 \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 linenumber among others, so it's not appropriate to put two things into one macro. Thus let's give the user a notice if she 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.

```
7360 \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 7368 \def\noeffect@info#1{\@ifundefined{#1}{}{\PackageInfo{gmdoc}{%
  The \bslash#1 macro is not supported by this package
  and therefore has no effect but this notification.}}}
```

```

7371     If you think it should have, please contact the
7372         maintainer^^J
indicated in the package's legal note.^^J}}}

```

The four macros formatting the macro and environment names, namely

```
\PrintDescribeMacro
\PrintMacroName
\PrintDescribeEnv
\PrintEnvName
```

\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 (æsthetical) 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.

```

7385 \AtEndInput{%
7386   \noeffect@info{\PrintDescribeMacro}%
7387   \noeffect@info{\PrintMacroName}%
7388   \noeffect@info{\PrintDescribeEnv}%
7389   \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
```

```

7394 \def\CodelineNumbered{\AtBeginDocument{\gag@index}%
7395 \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 he'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

```

7407 \let\CodelineIndex\@pageindexfalse
7408 \onlypreamble\CodelineIndex
7410 \let\PageIndex\@pageindextrue
7412 \onlypreamble\PageIndex

```

The next two declarations I find useful and smart:

```
\DisableCrossrefs
\EnableCrossrefs
\DisableCrossrefs
```

```

7416 \def\DisableCrossrefs{\@bsphack\gag@index\@esphack}
7418 \def\EnableCrossrefs{\@bsphack\ungag@index
7419   \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
\StopEventually
```

```

7427 \newcommand*\AlsoImplementation{\@bsphack%
7428   \long\def\StopEventually##1{\gdef\Finale{##1}}% we define \Finale
just to expand to the argument of \StopEventually not to add anything
to the end input hook because \Finale should only be executed 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 virtually numerous) input(s).
7439   \@esphack}
7441 \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 7448 \def\OnlyDescription{\@bsphack\long\def\StopEventually##1{%
\StopEventually
```

“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”

```
7452 ##1\endinput}\@esphack}
```

“If no \StopEventually command is given we silently ignore a \Finale issued.”

```
7457 \@relaxen\Finale
```

The \meta macro is so beautifully crafted in doc that I couldn’t resist copying it into gutils. It’s also available in Knuthian (*The $\text{\TeX}book$ format’s*) disguise \(<\langle the argument \rangle>).

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:

```
7469 \let\SpecialMainIndex=\DefIndex
```

```
\SpecialMainEnvIndex 7472 \def\SpecialMainEnvIndex{\csname\CodeDefIndex\endcsname*}%
we don't
type \DefIndex explicitly here because it's \outer, remember?
```

```
\SpecialIndex 7477 \let\SpecialIndex=\CodeCommonIndex
```

```
\SpecialUsageIndex 7479 \let\SpecialUsageIndex=\TextUsgIndex
```

```
\SpecialEnvIndex 7481 \def\SpecialEnvIndex{\csname\TextUsgIndex\endcsname*}
```

```
\SortIndex 7483 \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.

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 \verbatimchar 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 7503 \def\verbatimchar{\&}
```

One more a very neat macro provided by doc. I copy it verbatim and put into gutils, too. (\DeclareRobustCommand doesn’t issue an error if its argument has been defined, it only informs about redefining.)

```
* 7512 \pdef\*{\leavevmode\lower.8ex\hbox{$\backslash$\widetilde{\_}\,$}}}
```

```
\IndexPrologue \IndexPrologue is defined in line 5559. And other doc index commands too.
```

```
7519 \@ifundefined{main}{}{\let\DefEntry=\main}
```

```
7521 \@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 7529 \newcounter{StandardModuleDepth}
For now let's just mark the macro for further development
\DocstyleParms 7534 \noeffect@info{DocstyleParms}
For possible further development or to notify the user once and forever:
\Don'tCheckModules 7539 \Cemptify\Don'tCheckModules\neffect@info{Don'tCheckModules}
\CheckModules 7540 \Cemptify\CheckModules\neffect@info{CheckModules}
\Module The \Module macro is provided exactly as in doc.
\AltMacroFont 7544 \Cemptify\AltMacroFont\neffect@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."
\MakePercentIgnore 7550 \def\MakePercentIgnore{\catcode`\%\relax}
\MakePercentComment 7551 \def\MakePercentComment{\catcode`\%\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 humble¹² 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 oldstyle input commands change the
inner macros so that also \MakeShortVerb works as in doc (cf. line 7592).

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 7589 \def\OldDocInput{%
7591   \AtBeginInputOnce{\StraightEOL
7592     \let\@MakeShortVerb=\old@MakeShortVerb
7594   \OldMacrocodes}%
7595   \bgroup\@makeother\_ \% it's to allow _ in the filenames. The next macro will
    close the group.
7597   \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 nad what a bliss when the 'Successfully formated
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 adjust-
ments too because some crucial differences came out. I'd like to write a word about them
now.

¹² What a *false* modesty! ;-)

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

7630 `%\AVerySpecialMacro\% delete the first % when...`

was got with

```
\CodeDelim\.
% \AVerySpecialMacro % delete the first % when.\unskip|..|\CDPerc
```

One more difference is that my shortverb chars expand to their 12 versions in the math mode while in doc remain shortverb, so I added a declaration `\OldMakeShortVerb` etc.

Moreover, it's TEXing doc what inspired adding the `\StraightEOL` and `\QueerEOL` declarations.

Polishing, development and bugs

- `\MakePrivateLetters` theoretically may interfere with `\activeating` some chars to allow linebreaks. 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 [2072–2077](#). 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?¹³
- Do you use `\listoftables` and/or `\listoffigures` in your documentations? If so, I should 'EOL-straighten' them like `\tableofcontents`, I suppose (cf. line [2503](#)).
- 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.

¹³ 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.

(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>*', she should redefine the \EOFMark cs or end the file with a comment ending with \NoEOF macro defined below¹⁴:

```
7724 \foone{\catcode`^=M\active_}{%
\@NoEOF 7725 \def\@NoEOF#1^=M{%
7726     \relaxen\EOFMark\endinput}%
\@EOF 7727 \def\@EOF#1^=M{\endinput}%
\@EOF 7729 \def\@EOF{\queerEOL\@NoEOF}%
\EOF 7730 \def\@EOF{\queerEOL\@EOF}
```

As you probably see, \NoEOF have the 'immediate' \endinput effect: the file ends even in the middle of a line, the stuff after \NoEOF will be gobbled unlike with a bare \endinput.

```
7741 \endinput
7743 </package>
```

¹⁴ Thanks to Bernd Raichle at BachoTeX 2006 Session where he presented \inputing a file inside \edef.

b. The `gmdocc` Class For `gmdoc` Driver Files¹

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

© 2006, 2007 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>

for the details of that license.

LPPL status: "author-maintained".

```
41 \NeedsTeXFormat{LaTeX2e}
42 \ProvidesClass{gmdocc}
43 [2008/08/30 vo.80 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 &a. 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.

`nochanges` Simili modo, the `nochanges` option is provided to turn creating the change history off.

¹ This file has version number vo.80 dated 2008/08/30.

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 gmdoc.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 homonimic with the class'es name: mwrep for mwrep and mwbk for mwbk. For the symmetry there's also mwart option (equivalent to the default setting).

mwrep

mwbk

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 xtextra) 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.

\EOFMark

The \EOFMark in this class typesets like this (of course, you can redefine it as you wish):



The Code

¹³⁹ \RequirePackage{xkeyval}

A shorthands for options processing (I know xkeyval to little to redefine the default prefix and family).

\gm@DOX

¹⁴⁴ \newcommand*\gm@DOX{\DeclareOptionX[gmcc]<>}

\gm@EOX

¹⁴⁵ \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

¹⁵⁴ \newif\ifgmcc@mwcls

Note that the following option defines \gmcc@class#1.

class

¹⁵⁷ \gm@DOX{class}{% the default will be Marcin Woliński class (mwcls) analogous to article, see line [263](#).

\gmcc@CLASS

¹⁵⁹ \def\gmcc@CLASS{\#1}%

¹⁶⁰ \@for\gmcc@resa:=mwart,mwrep,mwbk\do{\%

¹⁶¹ \ifx\gmcc@CLASS\gmcc@resa\gmcc@mwclstrue\fi}%

¹⁶² }

mwart

¹⁶⁴ \gm@DOX{mwart}{\gmcc@class{mwart}}% The mwart class may also be declared explicitly.

```

mwrep  167 \gm@DOX{mwrep}{\gmcc@class{mwrep}}% If you need chapters, this option chooses
       an MW class that corresponds to report,
mwbk   171 \gm@DOX{mwbk}{\gmcc@class{mwbk}}% and this MW class corresponds to book.
article 174 \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 182 \gm@DOX{outeroff}{\let\outer\relax}% This option allows \outer-prefixed
       macros to be gmdoc-processed with all the bells and whistles.
\if@debug 186 \newif\if@debug
debug   188 \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 193 \gm@DOX{noindex}{%
194   \PassOptionsToPackage{noindex}{gmdoc}}% This option turns the writing
       outto .idx file off.
\if@gmccnochanges 198 \newif\if@gmccnochanges
nochanges 200 \gm@DOX{nochanges}{\@gmccnochangestrue}% This option turns the writing outto
       .glo file off.
gmeometric 204 \gm@DOX{gmeometric}{}% The gmeometric package causes the \geometry macro
       provided by geometry package is not restricted to the preamble.

```

Since version 0.99g of gmdoc the bundle goes \LaTeX and that means geometry should be loaded with dvipdfm option and the \pdfoutput counter has to be declared and that's what gmeometric does by default if with \LaTeX . And gmeometric has passed enough practical test. Therefore the gmeometric option becomes obsolete and the package is loaded always instead of original geometry.

As already mentioned, since version 0.99g the gmdoc bundle goes \LaTeX . That means that if \LaTeX is detected, we may load the fontspec package and the other two of basic three \LaTeX -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 223 \newif\ifgmcc@oldfonts
224 \gmcc@oldfontstrue
sysfonts 225 \gm@DOX{sysfonts}{\gmcc@oldfontsfalse}

```

Now we define a key-val option that sets the version of marginpar typewriter font definition (relevant only with the sysfonts option). 0 for OpenType LMTT LC visible for the system (not on my computer), 1 for LMTT LC specially on my computer, any else number to avoid an error if you don't have OpenType LMTT LC installed (and leave the default gmdoc's definition of \marginpartt; all the versions allow the user to define marginpar typewriter himself).

```

mptt 234 \gm@DOX{mptt}[17]{\def\mpttversion{\#1}}% the default value (17) works if the
\mpttversion      user puts the mptt option with no value. In that case leaving the default gm-
       doc's definition of marginpar typewriter and letting the user to redefine it her-
       self seemed to me most natural.

```

```

\gmcc@setfont 240 \def\gmcc@setfont#1{%
241   \gmcc@oldfontsfalse% note that if we are not in \LaTeX, this switch will be turned
       true in line 312
243   \AtBeginDocument{%

```

```

244  \@ifXeTeX{%
245      \defaultfontfeatures{Numbers={OldStyle,Proportional}}%
246      \setmainfont[Mapping=tex-text]{#1}%
247      \setsansfont[Mapping=tex-text,Scale=MatchLowercase]{Latin Modern Sans}%
248      \setmonofont[Scale=MatchLowercase]{Latin Modern Mono}%
249      \let\sl\it\let\textsl\textit
250  }{}%
251 }

minion 253 \gm@DOX{minion}{\gmcc@setfont{Minion Pro}}
pagella 254 \gm@DOX{pagella}{\gmcc@setfont{TeX Gyre Pagella}}%
\gmcc@PAGEALLA 255 \def\gmcc@PAGEALLA{1}%
256 }

fontspec 259 \gm@DOX{fontspec}{\PassOptionsToPackage{#1}{fontspec}}
263 \gm@EOX{class=mwart}%
We set the default basic class to be mwart.
266 \gm@EOX{mptt=o}%
We default to set the marginpar typewriter font to OpenType
    LMTT LC.

270 \DeclareOptionX*{\PassOptionsToPackage{\CurrentOption}{gmdoc}}
272 \ProcessOptionsX[\gmcc]<>

278 \IfFileExists{\gmcc@CLASS.cls}{}{\gmcc@mwclsfalse}%
As announced,
we do the ontological test to any mwcls.

290 \fi
291 \ifgmcc@mwcls
292     \XKV@ifundefined{XeTeXdefaultencoding}{}{%
293         \XeTeXdefaultencoding "cp1250"}%
mwcls are encoding-sensitive because
    MW uses Polish diacritics in the commentaries.
295     \LoadClass[fleqn,oneside,noindentfirst,11pt,withmarginpar,
296     sfheadings]{\gmcc@CLASS}%
297     \XKV@ifundefined{XeTeXdefaultencoding}{}{%
298         \XeTeXdefaultencoding "utf-8"}%
299 \else
300     \LoadClass[fleqn,11pt]{article}%
Otherwise the standard article is loaded.
302 \fi

307 \RequirePackage{gmutils}[2008/08/30]%
we load it early to provide \@ifXeTeX.

310 \ifgmcc@mwcls\afterfi\ParanoidPostsec\fi
312 \@ifXeTeX{}{\gmcc@oldfontstrue}
315 \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 `\[...]`:

```

\[[|\verbatim\stuff|]\].
323 \ifgmcc@oldfonts
324     \IfFileExists{lmodern.sty}%
We also examine the ontological status of this
        package
326     \RequirePackage{lmodern}%
and if it shows to be satisfactory (the package
        shows to be), we load it and set the proper font encoding.

```

```

329     \RequirePackage[T1]{fontenc}%
330 }
```

A couple of diacritics I met while gmdocing these files and The Source etc. Somewhy the accents didn't want to work at my X_ET_X settings so below I define them for X_ET_X as respective chars.

```

\grave 334 \def\grave{\`a}%
\acute 335 \def\acute{\^c}%
\acute 336 \def\acute{\^e}%
\idiaeres 337 \def\idiaeres{"\i}%
\nacute 338 \def\nacute{\^n}%
\circum 339 \def\circum{\^o}%
\oumlaut 340 \def\oumlaut{\\"o}%
\uumlaut 341 \def\uumlaut{\\"u}%
342 \else% this case happens only with XETX.
343   \let\do\relax
344   \do\Finv\do\Game\do\beth\do\gimel\do\daleth% these five caused the 'al-
345   ready defined' error.
346   \let\@zf@euenctrue\zf@euencfalse
347     \XeTeXthree%
\grave 352 \def\grave{\char"ooEo}%
\acute 353 \def\acute{\char"o107}% Note the space to be sure the number ends here.
\acute 355 \def\acute{\char"ooE9}%
\idiaeres 356 \def\idiaeres{\char"ooEF}%
\nacute 357 \def\nacute{\char"o144}%
\oumlaut 358 \def\oumlaut{\char"ooF6}%
\uumlaut 359 \def\uumlaut{\char"ooFC}%
\circum 360 \def\circum{\char"ooF4}%
361 \AtBeginDocument{%
\ae 362   \def\ae{\char"ooE6}%
363   \def\l{\char"o142}%
\oe 364   \def\oe{\char"o153}%
365 }
366 \fi
```

Now we set the page layout.

```

\gmddocMargins 369 \RequirePackage{gmeometric}
370 \def\gmddocMargins{%
371   \geometry{top=77pt,\height=687pt,\lines=53 lines but the lines option seems
372   not to work 2007/11/15 with TeX Live 2007 and XETX 0.996-patch1
373   left=4cm,right=2.2cm}}
375 \gmddocMargins
378 \if@debug% For debugging we load also the trace package that was very helpful to
379   me.
380   \RequirePackage{trace}%
381   \errorcontextlines=100% And we set an error info parameter.
382 \fi
384 \newcommand*\ifdtraceon{\if@debug\afterfi\traceon\fi}
385 \newcommand*\ifdtraceoff{\if@debug\traceoff\fi}

We load the core package:
388 \RequirePackage{gmdoc}
390 \ifgmcc@oldfonts
```

```

391  \@ifpackageloaded{lmodern}{% The Latin Modern font family provides a light
      condensed typewriter font that seems to be the most suitable for the margin-
      par CS marking.
\marginpartt 394  \def\marginpartt{\normalfont\fontseries{lc}\ttfamily}{}%
395  \else
\marginpartt 407  \def\marginpartt{\fontspec{LMTypewriter10-LightCondensed}}%
417  \fi
419  \ifnum1=0\csname\gmcc@PAGELLA\endcsname\relax
420    \RequirePackage{pxfonts,tgpagella,qpxmath}%
421  \fi
425  \raggedbottom
427  \setcounter{secnumdepth}{0}% We wish only the parts and chapters to be num-
      bered.
\thesection 430  \renewcommand*\thesection{\arabic{section}}% isn't it redundant at the above
      setting?
433  \@ifnotmw{}{%
434    \@ifclassloaded{mwart}{% We set the indentation of Contents:
435      \SetTOCIndents{{}{\quad}{\quad}{\quad}{\quad}}%
        {\quad}{\quad}{\quad}}% for mwart
      ...
436    \SetTOCIndents{{}{\bf g.\enspace}{\quad}{\quad}{\quad}{\quad}}%
        {\quad}{\quad}{\quad}}% and for the two other
      mwclss.
437  \pagestyle{outer}}% We set the page numbers to be printed in the outer and
      bottom corner of the page.
\titlesetup 440  \def\titlesetup{\bfseries\sffamily}% We set the title(s) to be boldface and
      sans serif.
443  \if@gmccnochanges\let\RecordChanges\relax\fi% If the nochanges option is
      on, we discard writing outto the .glo file.
446  \RecordChanges% We turn the writing the \changes outto the .glo file if not the
      above.
450  \dekclubs*% We declare the club sign | to be a shorthand for \verb|.
454  \edverbs% to redefine \[ so that it puts a shortverb in a \hbox.
455  \smartunder% and we declare the _ char to behave as usual in the math mode and
      outside math to be just an uderscore.
458  \exhyphenpenalty\hyphenpenalty%'cause mwcls set it =10000 due to Polish cus-
      toms.
\EOFMark 463  \RequirePackage{amssymb}
464  \def\EOFMark{\rightline{\ensuremath{\square}}}
466  \DoNotIndex{\@nx\@xa}%
467  }
469  \endinput

```

c. The gutils Package¹

Written by Grzegorz Murzynowski,
natror at o2 dot pl

© 2005, 2006, 2007, 2008 by Grzegorz 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".

Many thanks to my TeX Guru Marcin Woliński for his Texnical support.

```
83 \NeedsTeXFormat{LaTeX2e}
84 \ProvidesPackage{gutils}
85 [2008/10/03 vo.94 some rather Texnical macros, some of them
   tricky (GM)]
```

Intro

The gutils.sty package provides some macros that are analogous to the standard LATEX 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)TeX-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 ommitted.

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
```

```
157 \ifx\XeTeXversion\relax
158   \let\XeTeXversion\@undefined% If someone earlier used \ifundefined{%
   % XeTeXversion} to test whether the engine is XETEX, then \XeTeXversion
   is defined in the sense of ε-TEx tests. In that case we \let it to something
   really undefined. Well, we might keep sticking to \ifundefined, but it's
   a macro and it eats its arguments, freezing their catcodes, which is not what
   we want in line 3380.
```

¹ This file has version number vo.94 dated 2008/10/03.

```

165 \fi
167 \ifdefined\XeTeXversion
168 \XeTeXinencoding(utf-8)% we use Unicode dashes later in this file.
169 \fi% and if we are not in XeTeX, we skip them thanks to XeTeX-test.

```

A couple of abbreviations

```

\@xa 175 \let\@xa\expandafter
\@nx 176 \let\@nx\noexpand
\@xau 178 \def\@xau{\@xa\unexpanded\@xa}
\pdef 182 \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 189 \long\def\gobble#1{}
\@gobble 191 \let\@gobble\gobble
\gobbletwo 192 \let\gobbletwo\@gobbletwo
\provide 196 \long\pdef\provide#1{%
 197   \ifdefined#1%
 198     \ifx\relax#1\afterfifi{\def#1}%
 199     \else\afterfifi{\gmu@gobdef}%
 200     \fi
 201   \else\afterfi{\def#1}%
 202   \fi}
\gmu@gobdef 205 \long\def\gmu@gobdef#1#{}%
206   \def\gmu@tempa{}% it's a junk macro assignment to absorb possible prefixes.
208   \gobble}
\pprovide 211 \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 224 \long\def\@nameedef#1#2{%
 225   \@xa\edef\csname#1\endcsname{#2}}

```

\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.

```

236 \long\def\firstofone#1{#1}

```

The next command, \foone, is intended as two-argument for shortening of the \begingroup... \firstofone{\endgroup...} hack.

```

\foone 241 \long\def\foone#1{\begingroup#1\egfirstofone}
243 \long\def\egfirstofone#1{\endgroup#1}
\fooatletter 245 \long\def\fooatletter{\foone\makeatletter}

```

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 259 \pdef\newgif#1{%
260   {\escapechar\m@ne
261     \global\let#1\iffalse
262     \@gif#1\iftrue
263     \@gif#1\iffalse
264   }}}
```

'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 270 \def\@gif#1#2{%
271   \protected\@xa\gdef\csname\@xa\@gobbletwo\string#1%
272   g% the letter g for '\global'.
273   \@xa\@gobbletwo\string#2\endcsname
274   {\global\let#1#2} }

276 \pdef\newif#1{%
277   We not only make \newif \protected but also make it to define
278   \protected assignments so that premature expansion doesn't affect \if...
279   % \fi nesting.
280   \count@\escapechar\escapechar\m@ne
281   \let#1\iffalse
282   \@if#1\iftrue
283   \@if#1\iffalse
284   \escapechar\count@}

\@if 289 \def\@if#1#2{%
290   \protected\@xa\def\csname\@xa\@gobbletwo\string#1%
291   \@xa\@gobbletwo\string#2\endcsname
292   {\let#1#2} }

\hidden@iffalse 295 \pdef\hidden@iffalse{\iffalse}
\hidden@iftrue 296 \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 317 \pdef\grefstepcounter#1{%
318   {\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:

```
\hhrefstepcounter
332 \pdef\hhrefstepcounter#1{%
333   \ifhmode\leavevmode\fi\grefstepcounter{#1}}
```

By the way I read some lines from *The TeXbook* 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
340 \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
350 \def\glet{\global\let}
```

\LaTeX 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
358 \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
363 \long\def\addto@macro#1#2{%
364   \toks@\@xa{#1#2}%
365   \edef#1{\the\toks@}%
366 }% (\toks@ is a scratch register, namely \tokso.)
```

And for use in the very document,

```
\addtomacro
370 \let\addtomacro=\addto@macro
```

2008/08/09 I need to prepend something not add at the end—so

```
\prependtomacro
373 \long\def\prependtomacro#1#2{%
375   \edef#1{\unexpanded{#2}\@xa\unexpanded\@xa{#1}}}
```

Note that `\prependtomacro` can be prefixed.

```
\addtotoks
379 \long\def\addtotoks#1#2{%
380   #1=\@xa{\the#1#2}}
@emptyify
383 \newcommand*\@emptyify[1]{\let#1=\@empty}
@emptyify
384 \@ifdefinable\emptyify{\let\emptyify\@emptyify}
```

Note the two following commands are in fact one-argument.

```
\g@emptyify
388 \newcommand*\g@emptyify{\global\@emptyify}
@gemtpify
389 \@ifdefinable\gemtpify{\let\gemtpify\g@emptyify}
@relaxen
392 \newcommand\@relaxen[1]{\let#1=\relax}
@relaxen
393 \@ifdefinable\relaxen{\let\relaxen\@relaxen}
```

Note the two following commands are in fact one-argument.

```
\g@relaxen
397 \newcommand*\g@relaxen{\global\@relaxen}
@grelaxen
398 \@ifdefinable\grelaxen{\let\grelaxen\g@relaxen}
```

From the ancient xparse of T_EX Live 2007

The code of this section is rewritten contents of the xparse package version 0.17 dated 1999/09/10, the version available in T_EX Live 2007-13, in Ubuntu packages at least. It's a stub 'im Erwartung' (Schönberg) for the L^AT_EX3 bundle and it does what I want, namely defines \DeclareDocumentCommand. I rewrote the 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).

Well, I add the \protected prefix to the first macro.

```
413 \unless\ifdefined\@temptokenb
414 \newtoks\@temptokenb
415 \fi
416 \xparsed@args
417 \newtoks\xparsed@args
418 \long\def\DeclareDocumentCommand#1#2#3{%
    % #1 command to be defined,
    % #2 arguments specification,
    % #3 definition body.
419     \atempcpta\z@
420     \toks@{}%
421     \atemptokena\toks@
422     \atemptokenb\toks@
423     \atdc#2X% X is the guardian of parsing.
424     \protected\edef#1{\% The \protected prefix is my (GM) addition.
        \onx\atdc@
        {\the\toks@}%
        \xa\onx\csname\string#1\endcsname
        \onx#1%
    }%
425     \long\xa\def\csname\string#1\xa\endcsname
426     \the\attemptokena{#3}
427
428 \long\def\DeclareDocumentEnvironment#1#2#3#4{%
429     \xa\DeclareDocumentCommand\csname#1\endcsname{#2}{%
430         \xparsed@args\toks@
431         #3}%
432         \atxa\let\csname_end#1\endcsname\@parsed@endenv
433         \long\xa\def\csname_end\string\\#1\xa\endcsname
434         \the\attemptokena{#4}
435
436 \def\@parsed@endenv{%
437     \xa\@parsed@endenv@{\the\xparsed@args}
438
439 \def\@parsed@endenv@#1{%
440     \csname_end\string#1\endcsname}
441
442 \def\@ddc@#1#2#3{%
443     \ifx\protect\@typeset@protect
444     \xa\@firstofone
445     \else
446     \protect#3\xa\@gobble
447     \fi
448     {\toks@{#2}#1\the\toks@}}
449
450 \def\@ddc#1{%
451     \ifx#1X%
```

```

463   \else
464     \ifx#1m%
465       \addto@hook\@temptokenb_m%
466     \else
467       \toks@\@xa{%
468         \the\@xa\toks@
469         \csname_\@ddc@\the\@temptokenb\@xa\endcsname
470         \csname_\@ddc@#1\endcsname}%
471       \@temptokenb{}%
472     \fi
473     \advance\@tempcnta\@ne
474     \@temptokena\@xa{%
475       \the\@xa\@temptokena\@xa##\the\@tempcnta}%
476     \@xa
477     \@ddc
478   \fi}

\@ddc@s 480 \long\def\@ddc@s#1\toks@{%
481   \@ifstar
482   {\addto@hook\toks@\BooleanTrue#1\toks@}%
483   {\addto@hook\toks@\BooleanFalse#1\toks@}%

\@ddc@m 485 \long\def\@ddc@m#1\toks@#2{%
486   \addto@hook\toks@{\{#2\}}#1\toks@}%

\@ddc@o 488 \long\def\@ddc@o#1\toks@{%
489   \@ifnextchar[%]
490   {\@ddc@o@{#1}}
491   {\addto@hook\toks@\NoValue#1\toks@}%

\@ddc@o@ 493 \long\def\@ddc@o@#1[#2]{%
494   \addto@hook\toks@{\{#2\}}#1\toks@}

\@ddc 496 \def\@ddc#1{%
497   \ifx#1X%
498   \perhaps@grab@ms
499   \else
500   \ifx#1m%
501     \addto@hook\@temptokenb_m%
502   \else
503     \toks@\@xa{%
504       \the\@xa\toks@
505       \csname_\@ddc@x\the\@temptokenb\@xa\endcsname
506       \csname_\@ddc@#1\endcsname}%
507     \@temptokenb{}%
508     \ifx#10%
509       \let\next@ddc\grab@default
510     \else
511     \ifx#1C%
512       \let\next@ddc\grab@default
513     \fi
514     \fi
515     \fi
516     \advance\@tempcnta\@ne
517     \@temptokena\@xa{%
518       \the\@xa\@temptokena\@xa##\the\@tempcnta}%

```

```

519   \@xa
520   \next@ddc
521   \fi
522 }%
524 \let\next@ddc\@ddc
525 \def\grab@default#1{%
526   \toks@\@xa{%
527     \the\toks@
528     {#1}}%
529   \let\next@ddc\@ddc
530   \@ddc
531 }
532 \long\def\@ddc@0#1#2\toks@{%
533   \@ifnextchar[%
534   {\@ddc@o{#2}}%
535   {\addto@hook\toks@{{#1}}#2\toks@}}
536
537 \long\def\@ddc@c#1\toks@{%
538   \ifnextchar(%
539   {\@ddc@c@#1}%
540   {\PackageError{xparse}{Missing~coordinate~argument}%
541    {A~value~of~(o,o)~is~assumed}%
542    \addto@hook\toks@{{oo}}#1\toks@}%
543   }%
544 }
545 \long\def\@ddc@c@#1(#2,#3){%
546   \addto@hook\toks@{{{{#2}}{#3}}}}#1\toks@}
547
548 \long\def\@ddc@c#1#2\toks@{%
549   \ifnextchar(%
550   {\@ddc@c@#2}%
551   {\addto@hook\toks@{{#1}}#2\toks@}}%
552
553 \let\perhaps@grab@ms\relax
554 \def\grab@ms{%
555   \toks@\@xa{%
556     \the\@xa\toks@
557     \csname\@ddc@x\the\@temptokenb\endcsname
558   }%
559 }
560 \let\@ddc@m\undefined
561
562 \long\def\@ddc@xm#1\toks@#2{%
563   \addto@hook\toks@{{#2}}#1\toks@}
564
565 \long\def\@ddc@xmm#1\toks@#2#3{%
566   \addto@hook\toks@{{#2}{#3}}#1\toks@}
567
568 \long\def\@ddc@xmmm#1\toks@#2#3#4{%
569   \addto@hook\toks@{{#2}{#3}{#4}}#1\toks@}
570
571 \long\def\@ddc@xmmmm#1\toks@#2#3#4#5{%
572   \addto@hook\toks@{{#2}{#3}{#4}{#5}}#1\toks@}
573
574 \long\def\@ddc@xmmmmm#1\toks@#2#3#4#5#6{%
575   \addto@hook\toks@{{#2}{#3}{#4}{#5}{#6}}#1\toks@}
576
577 \long\def\@ddc@xmmmmmm#1\toks@#2#3#4#5#6#7{%
578   \addto@hook\toks@{{#2}{#3}{#4}{#5}{#6}{#7}}#1\toks@}
579

```

```

\@ddc@xxxxxxxxxx 581 \long\def\@ddc@xxxxxxxxxx#1\toks@#2#3#4#5#6#7#8{%
582   \addto@hook\toks@{\{#2}\{#3}\{#4}\{#5}\{#6}\{#7}\{#8}\}#1\toks@}

\@ddc@xxxxxxxxxx 584 \long\def\@ddc@xxxxxxxxxx#1\toks@#2#3#4#5#6#7#8#9{%
585   \addto@hook\toks@{\{#2}\{#3}\{#4}\{#5}\{#6}\{#7}\{#8}\{#9}\}#1\toks@}

\@ddc@xxxxxxxxxx 587 \long\def\@ddc@xxxxxxxxxx\the\toks@#1#2#3#4#5#6#7#8#9{%
588   \addto@hook\toks@{\{#1}\{#2}\{#3}\{#4}\{#5}\{#6}\{#7}\{#8}\{#9}\}\the%
      \toks@}

590 \let\@ddc@x\relax

DeclareDocumentEnvironment 592 \long\def\DeclareDocumentEnvironment#1#2#3#4{%
593   \csname\@xa\DeclareDocumentCommand\endcsname{#2}{%
594     #3}%
595   \csname\@namedef{end#1}{#4}%
596 }
597 \let\@parsed@endenv\undefined
598 \let\@parsed@endenv@\undefined
599 \def\IfSomethingTF#1{\def\something@in{#1}\If@SomethingTF}
600 \def\IfSomethingT#1#2#3{\def\something@in{#1}%
601   \If@SomethingTF{#2}{#3}\@empty}
602 \def\IfSomethingF#1#2#3{\def\something@in{#1}%
603   \If@SomethingTF{#2}\@empty{#3}}
604 \def\If@SomethingTF#1{%
605   \def\something@tmp{#1}%
606   \ifx\something@tmp\something@in
607     \@xa\@secondofthree
608   \else
609     \@xa\def\@xa\something@tmpb\@xa{#1}%
610     \ifx\something@tmp\something@tmpb
611       \@xa\@xa\@xa\@thirdofthree
612     \else
613       \@xa\@xa\@xa\@firstofone
614     \fi
615   \fi
616   \fi
617   \{@xa\If@SomethingTF\@xa{#1}}%
618 }
619 }

@secondofthree 621 \long\def\@secondofthree#1#2#3{#2}
@thirdofthree 622 \long\def\@thirdofthree#1#2#3{#3}
>NoValue 623 \def\NoValue{-NoValue-}
>NoValueInIt 624 \def\NoValueInIt{\NoValue}
>IfNoValueTF 625 \def\IfNoValueTF{\IfSomethingTF\NoValue}
>IfNoValueT 626 \def\IfNoValueT{\IfSomethingT\NoValue}
>IfNoValueF 627 \def\IfNoValueF{\IfSomethingF\NoValue}
>IfValueTF 628 \def\IfValueTF#1#2#3{\IfNoValueTF{#1}{#3}{#2}}
629 \let\IfValueT\IfNoValueF
630 \let\IfValueF\IfNoValueT
>BooleanFalse 631 \def\BooleanFalse{TF}
>BooleanTrue 632 \def\BooleanTrue{TT}
>IfBooleanTF 633 \def\IfBooleanTF#1{%
634   \if#1%
635     \@xa\@firstoftwo
636   \else

```

```

637   \xa\@secondoftwo
638   \fi
639 }
\IfBooleanT 641 \def\IfBooleanT#1#2{%
642   \IfBooleanTF{#1}{#2}\empty
643 }
\IfBooleanF 645 \def\IfBooleanF#1{%
646   \IfBooleanTF{#1}\empty
647 }

```

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^AT_EX 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 1649.

```

\ampulexdef 668 \DeclareDocumentCommand\ampulexdef{O{}mO{}O{}mmmm}{%
  % [#1] definition prefix, empty by default,
  % #2 macro to be redefined,
  % [#3] \def parameters string, empty by default,
  % [#4] definition body parameters to be taken in a one-step expansion of the
        redefined macro, empty by default,
  % #5 start token(s),
  % #6 end token(s)
  % #7 replacement.

```

For the example of usage see 1649.

```

\gmu@tempa 683 \def\gmu@tempa{#5}%
\gmu@tempb 684 \def\gmu@tempb{#6}%
\gmu@tempc 685 \def\gmu@tempc{#7}%
  we wrap the start, end and replacement tokens in macros
  to avoid unbalanced \ifs.
\edef\gmu@tempd{%
  687 \long\def\@nx\gmu@tempd
  688 #####1\all@other\gmu@tempa
  689 #####2\all@other\gmu@tempb
  690 #####3@nx\gmu@tempd{%
  691   @ifempty{#####3}{\hidden@iffalse}{\hidden@iftrue}}}}
\gmu@tempd%
  694 it defines \gmu@tempc to produce an open \if depending on whether
  the start and end token(s) are found in the meaning of #2.

\edef\gmu@tempe{%
  698 \@nx\gmu@tempd\all@other#2%
  699 \all@other\gmu@tempa
  700 \all@other\gmu@tempb\@nx\gmu@tempd
  701 }%
  702 }%

```

```

704 \gmu@tempe% we apply the checker and it produces an open \if.
705 \edef\gmu@tempd{%
706   \long\def\@nx\gmu@tempd
707   #####1\@xa\unexpanded\@xa{\gmu@tempa}%
708   #####2\@xa\unexpanded\@xa{\gmu@tempb}%
709   #####3\@nx\ampulexdef{%
710     we define a temporary macro with the parameters
711     delimited with the 'start' and 'end' parameters of \ampulexdef.
712     \@nx\unexpanded{#####1}%
713     \@nx\@xa\@nx\unexpanded
714     \@nx\@xa{\@nx\gmu@tempc}%
715     we replace the part of the redefined macro's
716     meaning with the replacement text.
717     \@nx\unexpanded{#####3}}}%
718 \gmu@tempd
719 \edef\gmu@tempf{\#4}%
720 \edef\gmu@tempe{%
721   #1\def\@nx\#2\#3{%
722     \@xa\@xa\@xa\gmu@tempd\@xa\#2\gmu@tempf\ampulexdef}}%
723 \gmu@tempe
724 \fi}
725
726 \ampulexhash \def\ampulexhash{####}%
727 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 737 \def\gmshowlists{\showboxdepth=1000\showboxbreadth=1000%
\showlists}

\nameshow 741 \newcommand\nameshow[1]{\@xa\show\csname#1\endcsname}
\nameshowthe 742 \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 753 \def\xiistring#1{%
754   \if\@nx#1\xiispace
755     \xiispace
756   \else
757     \string#1%
758   \fi}

```

`\@ifnextcat`, `\@ifnextac`

As you guess, we `\def \@ifnextcat à la \@ifnextchar`, see $\text{\LaTeX}_2\epsilon$ 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 a macro. Because an assignment is involved, you can’t test whether the next token is an active char.

```

\@ifnextcat 775 \long\def\@ifnextcat#1#2#3{%
 779   \def\reserved@d{#1}%
 780   \def\reserved@a{#2}%
 781   \def\reserved@b{#3}%
 782   \futurelet\@let@token\@ifncat}

\@ifncat 785 \def\@ifncat{%
 786   \ifx\@let@token\@sptoken
 787     \let\reserved@c\@xifncat
 788   \else
 789     \ifcat\reserved@d\@nx\@let@token
 790       \let\reserved@c\reserved@a
 791     \else
 792       \let\reserved@c\reserved@b
 793     \fi
 794   \fi
 795   \reserved@c}

797 {\def\:{\let\@sptoken=\:\: \% this makes \@sptoken a space token.
800 \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.

```

\@ifnextif 811 \long\pdef\@ifnextif#1#2#3{%
 815   \def\reserved@d{#1}%
 816   \def\reserved@a{#2}%
 817   \def\reserved@b{#3}%
 818   \futurelet\@let@token\@ifnif}

\@ifnif 821 \def\@ifnif{%
 822   \ifx\@let@token\@sptoken
 823     \let\reserved@c\@xifnif
 824   \else
 825     \if\reserved@d\@nx\@let@token
 826       \let\reserved@c\reserved@a
 827     \else
 828       \let\reserved@c\reserved@b
 829     \fi
 830   \fi
 831   \reserved@c}

834 {\def\:{\let\@sptoken=\:\: \% this makes \@sptoken a space token.
836 \def\:{\@xifnif}\@xa\gdef\:{\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 847 \long\pdef\@ifnextac#1#2{%
 848   \@ifnextcat\bgroup{#2}{\gm@ifnac{#1}{#2}}}

\gm@ifnac 850 \long\def\gm@ifnac#1#2#3{%

```

```
851  \ifcat\@nx~\@nx#3\afterfi{#1#3}\else\afterfi{#2#3}\fi}
```

Yes, it won't work for an active char \let to {₁, 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?)

Now, define a test that checks whether the next token is a genuine space, 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 =.

```
864 \let\gmu@reserveda\*%
/* 865 \def\*{%
866   \let\*\gmu@reserveda
867   \let\gm@letspace=\%
868 \*%
@ifnextspace 871 \def\@ifnextspace#1#2{%
872   \let\gmu@reserveda\*%
/* 873   \def\*{%
874     \let\*\gmu@reserveda
875     \ifx\@let@token\gm@letspace\afterfi{#1}%
876     \else\afterfi{#2}%
877     \fi}%
878   \futurelet\@let@token\*}
```

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.

```
@ifnextMac 887 \foone\obeylines{%
888   \long\pdef\@ifnextMac#1#2{%
889     \@ifnextchar^{\M{#1}{#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.

```
\afterfi 914 \long\def\afterfi#1#2\fi{\fi#1}
```

And two more of that family:

```
\afterfifi 916 \long\def\afterfifi#1#2\fi#3\fi{\fi\fi#1}
\afteriffifi 918 \long\def\afteriffifi#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.

```
\afterifffffifi 922 \long\def\afterifffffifi#1#2\fi#3\fi#4\fi{\fi#1}
```

```

\afteriffifi
  \long\def\afteriffifi#1#2\fi#3\fi#4\fi{\fi\fi#1}
\afterfifi
  \long\def\afterfifi#1#2\fi#3\fi#4\fi{\fi\fi\fi#1}

```

\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
  \long\def\gm@ifundefined#1%
    \ifcsname#1\endcsname% defined
      \xa\ifx\csname\endcsname\relax% but as \relax
        \afterfifi\@firstoftwo%
      \else% defined and not \relax
        \afterfifi\@secondoftwo%
      \fi
    \else% not defined
      \afterfi\@firstoftwo%
    \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^AT_EX'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 \long\def\@begnamedgroup#1{%
  \@ignorefalse% not to ignore blanks after group
  \begingroup\@endpefalse
  \edef\@currenvir{#1}% We could do recatcoding through \string but all the
  % name 'other' could affect a thousand packages so we don't do that and we'll
  % recatcode in a testing macro, see line 1029.
  \edef\@currenvline{\on@line}%
  \csname\#1\endcsname}% if the argument is a command's name (an environment'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* 998 \def\begin{\@ifstar{\@begnamedgroup}{%
\begin 999     \@begnamedgroup@ifcs}}
@begnamedgroup@ifcs 1002 \def\@begnamedgroup@ifcs#1{%
1003     \ifcsname#1\endcsname\afterfi{\@begnamedgroup{#1}}%
1004     \else\afterfi{\@latex@error{Environment #1 undefined}\@eha}%
1005     \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^AT_EX 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, 1, 2 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 1029 \long\def\@ifenvir#1#2#3{%
1031     \edef\gmu@reserveda{\@xa\string\csname\currenvir\endcsname}%
1032     \edef\gmu@reservedb{\@xa\string\csname#1\endcsname}%
1033     \ifx\gmu@reserveda\gmu@reservedb\afterfi{#2}%
1034     \else\afterfi{#3}%
1035     \fi}
@checkend 1037 \def\@checkend#1{\@ifenvir{#1}{}{\@badend{#1}}}
```

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.

From relsize

As file `relsize.sty`, v3.1 dated July 4, 2003 states, L^AT_EX 2_E version of these macros was written by Donald Arseneau asnd@triumf.ca and Matt Swift swift@bu.edu after the L^AT_EX 2.09 `smaller.sty` style file written by Bernie Cosell cosell@WILMA.BBN.COM.

I take only the basic, non-math mode commands with the assumption that there are the predefined font sizes.

```
\relsize
@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 \normalsize to \LARGE size. Negative n selects smaller fonts. \smaller == \relsize{-1}; \larger == \relsize{1}. \smallerr(my addition) == \relsize{-2}; \largerr guess yourself.
@smallerr  (Since \DeclareRobustCommand doesn't issue an error if its argument has been defined and it only informs about redefining, loading relsize remains allowed.)
@largerr
@relsize 1077 \pdef\relsize#1{%
```

```

1078 \ifmmode\@nomath\relsize\else
1079   \begingroup
1080     \tempcnta% assign number representing current font size
1081     \ifx\currsize\normalsize\else% funny order is to have most
1082     ...
1083     \ifx\currsize\small\else% ...likely sizes checked first
1084       \ifx\currsize\footnotesize\else
1085         \ifx\currsize\large\else
1086           \ifx\currsize\Large\else
1087             \ifx\currsize\scriptsize\else
1088               \ifx\currsize\tiny\else
1089                 \ifx\currsize\huge\else
1090                   \ifx\currsize\Huge\else
1091                     4\rs@unknown@warning% unknown state: \normalsize as
1092                     starting point
1093                     \fi\fi\fi\fi\fi\fi\fi

```

Change the number by the given increment:

```
1094   \advance\tempcnta#1\relax
```

watch out for size underflow:

```

1096   \ifnum\tempcnta<\z@\rs@size@warning{small}{\string\tiny}%
1097     \tempcnta\z@\fi
1098   \endgroup
1099   \ifcase\tempcnta% set new size based on altered number
1100     \tiny\or\scriptsize\or\footnotesize\or\small\or%
1101     \normalsize\or
1102     \large\or\Large\or\LARGE\or\huge\or\Huge\else
1103     \rs@size@warning{large}{\string\Huge}\Huge
1104   \fi\fi% end of \relsize.

\rs@size@warning 1104 \providecommand*\rs@size@warning[2]{\PackageWarning{gmutils}%
1105   (relsize)}{%
1106   \ifx\requested\too#1.\MessageBreak Using #2 instead}
\rs@unknown@warning 1107 \providecommand*\rs@unknown@warning{\PackageWarning{gmutils}%
1108   (relsize)}{Current font size
1109   is unknown! (Why?!?)\MessageBreak Assuming \string\normalsize}


```

And a handful of shorthands:

```

\larger 1112 \DeclareRobustCommand*\larger[1][\one]{\relsize{+\#1}}
\smaller 1113 \DeclareRobustCommand*\smaller[1][\one]{\relsize{-\#1}}
\textlarger 1114 \DeclareRobustCommand*\textlarger[2][\one]{{\relsize{+\#1}\#2}}
\textsmaller 1115 \DeclareRobustCommand*\textsmaller[2][\one]{{\relsize{-\#1}\#2}}
\largerr 1116 \pdef\largerr{\relsize{+2}}
\smallerr 1117 \pdef\smallerr{\relsize{-2}}

```

Some ‘other’ 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 `\chardef`ed.

```
1127 \foone{\catcode`\_=8}%
```

```

\subs 1128 {\let\subs=_}
       1130 \foone{\@makeother\_}%
\xiunder 1131 {\def\xiunder{_}}
           1133 \ifdefined\XeTeXversion
\xiunder 1134   \def\xiunder{\char"005F}%
           1135   \let\_xiunder
\xiunder 1136 \fi
           1138 \foone{\catcode`_=1\@makeother\{%
           1139   \catcode`\_=_\@makeother\}}%
           1140 [%]
\xiilbrace 1141 \def\xiilbrace[{}]
\xiirbrace 1142 \def\xiirbrace[]%
           1143 ]% of \firstofone

```

Note that \LaTeX 's \char1b and \char1b are of catcode 11 ('letter'), cf. The \LaTeX 2_ε Source file k, lines 129–130.

Now, let's define such a smart $_$ (underscore) which will be usual $_8$ in the math mode and $_2$ ('other') outside math.

```

\xiunder 1154 \foone{\catcode`\_=\active}
           1155 {%
\xiunder 1156   \newcommand*\xiunder{%
           1157     \catcode`\_=\active
           1158     \def_{\ifmmode\subs\else\_}\fi}}% We define it as \_ not just as \xiunder
           because some font encodings don't have _ at the \char`\_ position.
\xiibackslash 1164 \foone{\catcode`\!=0
           1165   \@makeother\`}
\xiibackslash 1166 {!newcommand!*xiibackslash{}}
\xiibackslash 1170 \let\xiibackslash=\xiibackslash
\xipercent 1174 \foone{\@makeother\%}
\xipercent 1175 {\def\xipercent{\%}}
\xiand 1178 \foone{\@makeother\&}%
\xiand 1179 {\def\xiand{\&}}
\xiispace 1181 \foone{\@makeother\_\_}%
\xiispace 1182 {\def\xiispace{\_}}

```

We introduce \visible from Will Robertson's xltextra if available. It's not sufficient $\text{\ifpackageloaded{xltextra}}$ since \xxt@visible is defined only unless no-verb option is set. 2008/08/06 I recognized the difference between \xiispace which has to be plain 'other' char (used in \xiistring) and something visible to be printed in any font.

```

\xiispace 1191 \AtBeginDocument{%
           1192   \ifdefined\xxt@visible
             \let\xiispace\xxt@visible
           1193   \else
             \let\xiispace\xiispace
           1195   \fi}

```

Metasymbols

I fancy also another Knuthian trick for typesetting *<metasymbols>* in *The TeXbook*. 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 1217 \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."

```
1225   \ensuremath\langle  
1226   \ifmmode\@xa\@nfss@text\fi  
1227   {  
1228     \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.

```
1236   #1\%  
1238 } \ensuremath\rangle  
1239 }
```

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.

```
\meta@font@select 1247 \def\meta@font@select{\it}
```

The below `\meta`'s drag² is a version of *The TeXbook*'s one.

```
\langle...> 1253 \def\langle#1{\meta{#1}}
```

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 linebreak just before nor just after themselves. The `\discretionary` TeX primitive has three arguments: #1 'before break', #2 'after break', #3 'without break', remember?

```
\discre 1264 \def\discre#1#2#3{\leavevmode\kernosp%  
1265   \discretionary{#1}{#2}{#3}\penalty10000\hskip\relax}  
\discret 1266 \def\discret#1{\leavevmode\kernosp%  
1267   \discretionary{#1}{#1}{#1}\penalty10000\hskip\relax}
```

A tiny little macro that acts like `\-` outside the math mode and has its original meaning inside math.

```
1271 \def\:{\ifmmode\afterfi{\mskip\medmuskip}\else\afterfi{\discret{  
}}\fi}
```

```
\vs 1273 \newcommand*{\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 `re\catcode`ing has no effect).

```
\printspaces 1279 \def\printspaces#1{{\let~=\vs\let\ =\vs\gm@pswords#1\@nil}}  
\gm@pswords 1281 \def\gm@pswords#1#2\@nil{%
```

1282 \ifx\relax#1\relax\else#1\fi
1283 \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 guardian space: it has been already
by \printspaces.

```
\sfname 1288 \pdef\sfname#1{\textsf{\printspaces{#1}}}
```

```
\gmu@discretionaryslash 1290 \def\gmu@discretionaryslash{\discre{/}{\hbox{}}{}/}% the  
second pseudo-argument nonempty to get \hyphenpenalty  
not \exhyphenpenalty.
```

```
\file 1295 \pdef\file#1{\gmu@printslashes#\gmu@printslashes}
```

```
\gmu@printslashes 1297 \def\gmu@printslashes#1#2\gmu@printslashes{  
1298   \sfname{#1}%  
1299   \ifx\gmu@printslashes#2\gmu@printslashes  
1300   \else  
1301   \textsf{\gmu@discretionaryslash}%  
1302   \afterfi{\gmu@printslashes#2\gmu@printslashes}\fi}
```

it allows the spaces in the filenames (and prints them as `\`).

The below macro I use to format the packages' names.

```
\pk 1310 \pdef\pk#1{\textsf{\textup{#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 1321 \DeclareRobustCommand*{\cs}[2][\bslash]{%  
1322   \def\-\{\discretionary{\rmfamily-}{}{}\}%  
1323   \def\{\char`\{}\def\}{\char`\}\ttfamily\#1#2}}
```

```
\env 1327 \pdef\env#1{\cs[]#1}
```

And for the special sequences like `^~A`:

```
1330 \foone{\makeother\~}  
1331 {\pdef\hathat#1{\cs[^~]#1}}
```

And one for encouraging linebreaks e.g., before long verbatim words.

```
\possfil 1336 \newcommand*\possfil{\hfil\penalty1000\hfilneg}
```

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 1346 \def\cmd#1{\cs{\@xa\cmd@to@cs\string#1}}  
\cmd@to@cs 1348 \def\cmd@to@cs#1#2{\char\number`#2\relax}
```

`\marg{text}` prints `{text}`, 'mandatory argument'.

```

\marg 1352 \def\marg#1{{\ttfamily\char`\\{}\meta{#1}{\ttfamily\char`\\{}}
\oarg{text} prints [text], ‘optional argument’. Also \oarg{text} does that.
\oarg 1357 \def\oarg{\@ifnextchar[\@oargsq\@arg}
\@oarg 1359 \def\@oarg#1{{\ttfamily[]}\meta{#1}{\ttfamily[]}}
\@oargsq 1360 \def\@oargsq[#1]{\@oarg{#1}}
\parg 1361 \parg{te,xt} prints ((te,xt)), ‘picture mode argument’.
\parg 1364 \def\parg{\@ifnextchar(\@pargp\@parg}
\@parg 1366 \def\@parg#1{{\ttfamily()}\meta{#1}{\ttfamily()}}
\@pargp 1367 \def\@pargp(#1){\@parg{#1}}

```

But we can have all three in one command.

```

\arg 1371 \AtBeginDocument{%
\arg 1372   \let\math@arg\arg
\arg 1373   \def\arg{\ifmmode\math@arg\else\afterfi{%
\arg 1374     \@ifnextchar[%
\arg 1375       \@oargsq{\@ifnextchar(%%
\arg 1376         \@pargp\marg})\fi}%
\arg 1377 }

```

Now you can write

```

\arg{mand.\arg} to get {\mand. arg},
\arg[opt.\arg] for [opt. arg] and
\arg(pict.\arg) for (pict. arg).
And $\arg(1+i)=\pi/4$ for  $\arg(1 + i) = \pi/4$ .

```

Storing and restoring the meanings of cses

First a Boolean switch of globalness of assignments and its verifier.

```

\ifgmu@SMglobal 1392 \newif\ifgmu@SMglobal
\SMglobal 1394 \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 1405 \pdef\StoreMacro{%
\begin{group}\makeatletter\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 1529.

```

\egStore@Macro 1411 \long\def\egStore@Macro#1{\endgroup\Store@Macro{#1}}
\egStore@MacroSt 1412 \long\def\egStore@MacroSt#1{\endgroup\Store@MacroSt{#1}}
\Store@Macro 1414 \long\def\Store@Macro#1{%
\begin{group}\escapechar92
\ifgmu@SMglobal\afterfi\global\fi
\@xa\let\csname\gmu/store\string#1\endcsname\fi}
\end{group}

```

² Think of the drags that transform a very nice but rather standard ‘auntie’ (‘Tante’ in Deutsch) into a most adorable Queen ;-).

```

1418  \global\gmu@SMglobalfalse}
\Store@MacroSt 1421 \long\def\Store@MacroSt#1{%
1422  \edef\gmu@smtempa{%
1423  \ifgmu@SMglobal\global\fi
1424  \nx\let\x@\nx\csname/gmu/store\bslash#1\endcsname% we add
       backslash because to ensure compatibility between \StoreMacro
       and \StoreMacro*, that is. to allow writing
       e.g. \StoreMacro{kitten} and then \RestoreMacro*{kitten} to
       restore the meaning of \kitten.
1430  \x@\nx\csname#1\endcsname}
1431  \gmu@smtempa
1432  \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 may be separated with commas but they don't have to.

```

\StoreMacros 1448 \long\pdef\StoreMacros{\begin{group}\makeatletter\Store@Macros}
\Store@Macros 1449 \long\def\Store@Macros#1{\end{group}
1450  \gmu@setsetSMglobal
1451  \let\gml@StoreCS\Store@Macro
1452  \gml@storemacros#1.}

\gmu@setsetSMglobal 1455 \def\gmu@setsetSMglobal{%
1456  \ifgmu@SMglobal
1457  \let\gmu@setSMglobal\gmu@SMglobaltrue
1458  \else
1459  \let\gmu@setSMglobal\gmu@SMglobalfalse
1460  \fi}

```

And the inner iterating macro:

```

\gml@storemacros 1463 \long\def\gml@storemacros#1{%
\gmu@reserveda 1464  \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
                     ing.
1467  \if\gmu@reserveda.% a dot finishes storing.
1468  \global\gmu@SMglobalfalse
1469  \else
1470  \if\gmu@reserveda,% The list this macro is put before may contain commas
                     and that's O.K., we just continue the work.
                     \afterfifi\gml@storemacros
1473  \else% what is else this shall be stored.
1474  \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.
                     \afterfifi{\gmu@setSMglobal\gml@storemacros}%
1477  \fi
1479  \fi}

```

And for the restoring

```

\RestoreMacro 1485 \pdef\RestoreMacro{%

```

```

1486  \begingroup\makeatletter\@ifstar\egRestore@MacroSt%
      \egRestore@Macro}
1488 \long\def\egRestore@Macro#1{\endgroup\Restore@Macro{#1}}
1489 \long\def\egRestore@MacroSt#1{\endgroup\Restore@MacroSt{#1}}
\Restore@Macro 1491 \long\def\Restore@Macro#1{%
  \escapechar92
  \ifgmu@SMglobal\afterfi\global\fi
  \let\@xa\let\@xa#1\csname\gmu/store/string#1\endcsname
  \global\gmu@SMglobalfalse}
\Restore@MacroSt 1497 \long\def\Restore@MacroSt#1{%
  \edef\gmu@smtempa{%
    \ifgmu@SMglobal\global\fi
    \let\@nx\let\@xa\@nx\csname#1\endcsname
    \let\@xa\@nx\csname/gmu/store/bslash#1\endcsname}% cf. the commentary
    in line 1424.
  \gmu@smtempa
  \global\gmu@SMglobalfalse}
\RestoreMacros 1507 \long\pdef\RestoreMacros{\begingroup\makeatletter\Restore@Macros}
\Restore@Macros 1509 \long\def\Restore@Macros#1{\endgroup
  \gmu@setsetSMglobal
  \let\gml@StoreCS\Restore@Macro% we direct the core cs towards restoring
  and call the same iterating macro as in line 1452.
  \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 1520 \def\StoredMacro{\begingroup\makeatletter\Stored@Macro}
\Stored@Macro 1521 \long\def\Stored@Macro#1{\endgroup\Restore@Macro#1#1}

```

To be able to call a stored cs without restoring it.

```

\storedcsname 1524 \def\storedcsname#1{%
  \let\csname\gmu/store/bslash#1\endcsname}
  2008/08/03 we need to store also an environment.

```

```

\StoreEnvironment 1529 \pdef\StoreEnvironment#1{%
  \let\StoreMacro*\{#1\}\let\StoreMacro*\{end#1\}}

```

```

\RestoreEnvironment 1533 \pdef\RestoreEnvironment#1{%
  \let\RestoreMacro*\{#1\}\let\RestoreMacro*\{end#1\}}

```

It happended (see the definition of `\@docininclude` 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 1550 \long\def\StoringAndRelaxingDo{%
  \gmu@SMdo@setscope
  \long\def\do##1{%
    \gmu@SMdo@scope
    \let\csname\gmu/store/string##1\endcsname##1%
    \gmu@SMdo@scope\let##1\relax}}

```

```
\gmu@SMdo@setscope 1557 \def\gmu@SMdo@setscope{%
 1558   \ifgmu@SMglobal\let\gmu@SMdo@scope\global
 1559   \else\let\gmu@SMdo@scope\relax
 1560   \fi
 1561   \global\gmu@SMglobalfalse}
```

And here is the counter-definition for restore.

```
\RestoringDo 1570 \long\def\RestoringDo{%
 1571   \gmu@SMdo@setscope
 1572   \long\def\do##1{%
 1573     \gmu@SMdo@scope
 1574     \@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 1582 \def\n@melet#1#2{%
 1583   \edef\gmu@nl@reserveda{%
 1584     \let\@xa\@nx\csname#1\endcsname
 1585     \@xa\@nx\csname#2\endcsname}%
 1586   \gmu@nl@reserveda}
```

The `\global` prefix doesn’t work with `\n@melet` so we define the alternative.

```
\gn@melet 1590 \def\gn@melet#1#2{%
 1591   \edef\gmu@nl@reserveda{%
 1592     \global\let\@xa\@nx\csname#1\endcsname
 1593     \@xa\@nx\csname#2\endcsname}%
 1594   \gmu@nl@reserveda}
```

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 IMO.

```
\not@onlypreamble 1611 \newcommand\not@onlypreamble[1]{{%
 1612   \def\do##1{\ifx##1##1\else\@nx\do\@nx##1\fi}%
 1613   \xdef\@preamblecmds{\@preamblecmds}}}
 1615 \not@onlypreamble\@preamblecmds
 1616 \not@onlypreamble\@ifpackageloaded
 1617 \not@onlypreamble\@ifclassloaded
 1618 \not@onlypreamble\@ifl@aded
 1619 \not@onlypreamble\@pkgextension
```

And let’s make the message of only preamble command’s forbidden use informative a bit:

```
\gm@notprerr 1624 \def\gm@notprerr{\canbeusedonlyinpreamble(\online)}
 1626 \AtBeginDocument{%
 1627   \def\do##1{\@nx\do\@nx##1}%
 1628   \edef\@preamblecmds{%
 1629     \def\@nx\do##1{%
```

```

1630      \def##1{\@nx\PackageError{gmutils/TeX}%
1631          {\@nx\string##1\@nx\gm@notprerr}\@nx\@eha}}%
1632      \@preamblecmds}

```

A subtle error raises: the L^AT_EX standard \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^AT_EX’s \nocite that checks \ifx\onlypreamble\document. So hoping this is a rare problem, we circumvent in with. 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 1649 \def\gmu@nocite@ampulex{\% we wrap the stuff in a macro to hide an open \if.
                           And not to make the begin-input hook not too large. the first is the parameters
                           string and the second the argument for one-level expansion of \nocite so it
                           has to consist of two times less hashes than the first. Both hash strings are
                           doubled to pass the first \def.
1656 \ampulexdef []\nocite[####1] [{####1}]% note the double brace around
                           % #3.
1658 \ifx
1659 {\onlypreamble\document}%
1660 \iftrue
1662 \AtBeginDocument\gmu@nocite@ampulex

```

Third person pronouns

Is a reader of my documentations ‘she’ or ‘he’ and does it make a difference?

Not to favour any gender in the personal pronouns, define commands that’ll print alternately masculine and feminine pronoun of third person. By ‘any’ I mean not only typically masculine and typically feminine but the entire amazingly rich variety of people’s genders, *including* those who do not describe themselves as ‘man’ or ‘woman’.

One may say two pronouns is far too little to cover this variety but I could point Ursula’s K. LeGuin’s *The Left Hand Of Darkness* as another acceptable answer. In that moody and moderate SF novel the androgynous persons are usually referred to as ‘mister’, ‘sir’ or ‘he’: the meaning of reference is extended. Such an extension also my automatic pronouns do suggest. It’s *not* political correctness, it’s just respect to people’s diversity.

```

\gm@PronounGender 1689 \newcounter{gm@PronounGender}
\gm@atppron 1691 \newcommand*\gm@atppron[2]{%
1692   \stepcounter{gm@PronounGender}% remember \stepcounter is global.
1693   \ifodd\value{gm@PronounGender}\#1\else\#2\fi}
\heshe 1695 \newcommand*\heshe{\gm@atppron{he}{she}}
\hisher 1696 \newcommand*\hisher{\gm@atppron{his}{her}}
\himher 1697 \newcommand*\himher{\gm@atppron{him}{her}}
\hishers 1698 \newcommand*\hishers{\gm@atppron{his}{hers}}
\HeShe 1700 \newcommand*\HeShe{\gm@atppron{He}{She}}
\HisHer 1701 \newcommand*\HisHer{\gm@atppron{His}{Her}}
\HimHer 1702 \newcommand*\HimHer{\gm@atppron{Him}{Her}}
\HisHers 1703 \newcommand*\HisHers{\gm@atppron{His}{Hers}}

```

Improvements to mwcls sectioning commands

That is, ‘Experi-mente’³ 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.

1722 \AtBeginDocument{ % because we don’t know when exactly hyperref is loaded and maybe after this package.

```
NoNumSecs 1724 \@ifpackageloaded{hyperref}{\newcounter{NoNumSecs}%
1725   \setcounter{NoNumSecs}{617} % to make \refing to an unnumbered section
                                visible (and funny?).
\gm@hyperrefstepcounter 1727 \def\gm@hyperrefstepcounter{\refstepcounter{NoNumSecs}%
1728   \pdef\gm@targetheading#1{%
1729     \hypertarget{#1}{#1}}% end of then
\gm@hyperrefstepcounter 1730 {\def\gm@hyperrefstepcounter{}%
1731   \def\gm@targetheading#1{#1}}% end of else
1732 }% of \AtBeginDocument
```

Auxiliary macros for the kernel sectioning macro:

```
bersectionsoutofmainmatter 1735 \def\gm@dontnumbersectionsoutofmainmatter{%
1736   \if@mainmatter\else\HeadingNumberedfalse\fi}
\m@clearpagesduetoopenright 1737 \def\gm@clearpagesduetoopenright{%
1738   \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 gmdoc 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 1751 \long\def\@ifnotmw#1#2{\gm@ifundefined{mw@sectionxx}{#1}{#2}}
```

The kernel of MW’s sectioning commands:

```
\mw@sectionxx 1776 \@ifnotmw{}{%
1777 \def\mw@sectionxx#1#2[#3]{#4}%
1778   \edef\mw@HeadingLevel{\csname #1@level\endcsname
1779     \space}% space delimits level number!
1780   \ifHeadingNumbered
1781     \ifnum\mw@HeadingLevel>\c@secnumdepth%
1782       \HeadingNumberedfalse\fi
line below is in \gm@ifundefined to make it work in classes other than mwbk
1784   \gm@ifundefined{if@mainmatter}{}{%
1785     \gm@dontnumbersectionsoutofmainmatter}
1785   \fi
%   \ifHeadingNumbered
%   \refstepcounter{#1}%
```

³ A. Berg, Wozzeck.

```

%      \protected@edef\HeadingNumber{\csname
%        the#1\endcsname\relax}%
% \else
%   \let\HeadingNumber\empty
% \fi

\HeadingRHeadText 1794 \def\HeadingRHeadText{\#2}%
\HeadingTOCText 1795 \def\HeadingTOCText{\#3}%
\HeadingText 1796 \def\HeadingText{\#4}%
\mw@HeadingType 1797 \def\mw@HeadingType{\#1}%
1798 \if\mw@HeadingBreakBefore
1799   \if@specialpage\else\thispagestyle{closing}\fi
1800   \gm@ifundefined{if@openright}{}{%
1801     \gm@clearpagesduetoopenright}%
1802   \if\mw@HeadingBreakAfter
1803     \thispagestyle{blank}\else
1804     \thispagestyle{opening}\fi
1805   \global\@topnum\z@
1806 \fi% of \if\mw@HeadingBreakBefore

placement of \refstep suggested by me (GM):
1808 \ifHeadingNumbered
1809   \refstepcounter{\#1}%
1810   \protected@edef\HeadingNumber{\csname the#1\endcsname\relax}%
1811 \else
1812   \let\HeadingNumber\empty
1813   \gm@hyperrefstepcounter
1814 \fi% of \ifHeadingNumbered
1816 \if\mw@HeadingRunIn
1817   \mw@runinheading
1818 \else
1819   \if\mw@HeadingWholeWidth
1820     \if@twocolumn
1821       \if\mw@HeadingBreakAfter
1822         \onecolumn
1823         \mw@normalheading
1824         \pagebreak\relax
1825         \if@twoside
1826           \null
1827           \thispagestyle{blank}%
1828           \newpage
1829         \fi% of \if@twoside
1830       \twocolumn
1831     \else
1832       \atopnewpage[\mw@normalheading]%
1833     \fi% of \if\mw@HeadingBreakAfter
1834   \else
1835     \mw@normalheading
1836     \if\mw@HeadingBreakAfter\pagebreak\relax\fi
1837   \fi% of \if@twocolumn
1838 \else
1839   \mw@normalheading
1840   \if\mw@HeadingBreakAfter\pagebreak\relax\fi
1841 \fi% of \if\mw@HeadingWholeWidth

```

```

1842     \fi% of \if\mw@HeadingRunIn
1843 }

```

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 postskeep

```

1867 \relax\n SetSectionFormatting
\SetSectionFormatting 1868 \newcommand*\SetSectionFormatting[5][\empty]%
1869   \ifx\empty#1\relax\else% empty (not \empty!) #1 also launches \else.
1870     \def\mw@HeadingRunIn{\o}\def\mw@HeadingBreakBefore{\o}%
1871     \def\mw@HeadingBreakAfter{\o}\def\mw@HeadingWholeWidth{\o}%
1872     \@ifempty{#1}{}{\mw@processflags#1,\relax}% If #1 is omitted, the flags
1873       are left unchanged. If #1 is given, even as [], the flags are first cleared and
1874       then processed again.
1875   \fi
1876   \gm@ifundefined{#2}{\@namedef{#2}{\mw@section{#2}}}{}
1877   \mw@secdef{#2}{@preskip}{#3}{2\oblig.}%
1878   \mw@secdef{#2}{@head}{#4}{3\oblig.}%
1879   \mw@secdef{#2}{@postskip}{#5}{4\oblig.}%
1880   \ifx\empty#1\relax
1881     \mw@secundef{#2@flags}{1(optional)}%
1882   \else\mw@setflags{#2}%
1883   \fi
1884 \def\mw@secdef#1#2#3#4{%
1885   % #1 the heading name,
1886   % #2 the command distinctor,
1887   % #3 the meaning,
1888   % #4 the number of argument to error message.
1889   \@ifempty{#3}
1890     {\mw@secundef{#1#2}{#4}}
1891     {\@namedef{#1#2}{#3}}}
1892 \def\mw@secundef#1#2{%
1893   \gm@ifundefined{#1}{%
1894     \ClassError{mwcls/gm}{%
1895       command\bslash#1\undefined\MessageBreak
1896       after\bslash\SetSectionFormatting!!!\MessageBreak}{%
1897         Provide the #2 argument of \bslash
1898         SetSectionFormatting.}}}
1899
1900
1901
1902
1903
1904
1905
1906
1907
1908
1909
1910
1911
1912
1913
1914
1915
1916
1917
1918
1919
1920
1921
1922
1923
1924
1925
1926
1927
1928
1929
1930
1931
1932
1933
1934
1935
1936
1937
1938
1939
1940
1941
1942
1943
1944
1945
1946
1947
1948
1949
1950
1951
1952
1953
1954
1955
1956
1957
1958
1959
1960
1961
1962
1963
1964
1965
1966
1967
1968
1969
1970
1971
1972
1973
1974
1975
1976
1977
1978
1979
1980
1981
1982
1983
1984
1985
1986
1987
1988
1989
1990
1991
1992
1993
1994
1995
1996
1997
1998
1999
2000
2001
2002
2003
2004
2005
2006
2007
2008
2009
2010
2011
2012
2013
2014
2015
2016
2017
2018
2019
2020
2021
2022
2023
2024
2025
2026
2027
2028
2029
2030
2031
2032
2033
2034
2035
2036
2037
2038
2039
2040
2041
2042
2043
2044
2045
2046
2047
2048
2049
2050
2051
2052
2053
2054
2055
2056
2057
2058
2059
2060
2061
2062
2063
2064
2065
2066
2067
2068
2069
2070
2071
2072
2073
2074
2075
2076
2077
2078
2079
2080
2081
2082
2083
2084
2085
2086
2087
2088
2089
2090
2091
2092
2093
2094
2095
2096
2097
2098
2099
20100
20101
20102
20103
20104
20105
20106
20107
20108
20109
20110
20111
20112
20113
20114
20115
20116
20117
20118
20119
20120
20121
20122
20123
20124
20125
20126
20127
20128
20129
20130
20131
20132
20133
20134
20135
20136
20137
20138
20139
20140
20141
20142
20143
20144
20145
20146
20147
20148
20149
20150
20151
20152
20153
20154
20155
20156
20157
20158
20159
20160
20161
20162
20163
20164
20165
20166
20167
20168
20169
20170
20171
20172
20173
20174
20175
20176
20177
20178
20179
20180
20181
20182
20183
20184
20185
20186
20187
20188
20189
20190
20191
20192
20193
20194
20195
20196
20197
20198
20199
20200
20201
20202
20203
20204
20205
20206
20207
20208
20209
20210
20211
20212
20213
20214
20215
20216
20217
20218
20219
20220
20221
20222
20223
20224
20225
20226
20227
20228
20229
20230
20231
20232
20233
20234
20235
20236
20237
20238
20239
20240
20241
20242
20243
20244
20245
20246
20247
20248
20249
20250
20251
20252
20253
20254
20255
20256
20257
20258
20259
202510
202511
202512
202513
202514
202515
202516
202517
202518
202519
202520
202521
202522
202523
202524
202525
202526
202527
202528
202529
202530
202531
202532
202533
202534
202535
202536
202537
202538
202539
202540
202541
202542
202543
202544
202545
202546
202547
202548
202549
202550
202551
202552
202553
202554
202555
202556
202557
202558
202559
202560
202561
202562
202563
202564
202565
202566
202567
202568
202569
202570
202571
202572
202573
202574
202575
202576
202577
202578
202579
202580
202581
202582
202583
202584
202585
202586
202587
202588
202589
202590
202591
202592
202593
202594
202595
202596
202597
202598
202599
2025100
2025101
2025102
2025103
2025104
2025105
2025106
2025107
2025108
2025109
2025110
2025111
2025112
2025113
2025114
2025115
2025116
2025117
2025118
2025119
20251100
20251101
20251102
20251103
20251104
20251105
20251106
20251107
20251108
20251109
20251110
20251111
20251112
20251113
20251114
20251115
20251116
20251117
20251118
20251119
202511100
202511101
202511102
202511103
202511104
202511105
202511106
202511107
202511108
202511109
202511110
202511111
202511112
202511113
202511114
202511115
202511116
202511117
202511118
202511119
2025111100
2025111101
2025111102
2025111103
2025111104
2025111105
2025111106
2025111107
2025111108
2025111109
2025111110
2025111111
2025111112
2025111113
2025111114
2025111115
2025111116
2025111117
2025111118
2025111119
20251111100
20251111101
20251111102
20251111103
20251111104
20251111105
20251111106
20251111107
20251111108
20251111109
20251111110
20251111111
20251111112
20251111113
20251111114
20251111115
20251111116
20251111117
20251111118
20251111119
202511111100
202511111101
202511111102
202511111103
202511111104
202511111105
202511111106
202511111107
202511111108
202511111109
202511111110
202511111111
202511111112
202511111113
202511111114
202511111115
202511111116
202511111117
202511111118
202511111119
2025111111100
2025111111101
2025111111102
2025111111103
2025111111104
2025111111105
2025111111106
2025111111107
2025111111108
2025111111109
2025111111110
2025111111111
2025111111112
2025111111113
2025111111114
2025111111115
2025111111116
2025111111117
2025111111118
2025111111119
20251111111100
20251111111101
20251111111102
20251111111103
20251111111104
20251111111105
20251111111106
20251111111107
20251111111108
20251111111109
20251111111110
20251111111111
20251111111112
20251111111113
20251111111114
20251111111115
20251111111116
20251111111117
20251111111118
20251111111119
202511111111100
202511111111101
202511111111102
202511111111103
202511111111104
202511111111105
202511111111106
202511111111107
202511111111108
202511111111109
202511111111110
202511111111111
202511111111112
202511111111113
202511111111114
202511111111115
202511111111116
202511111111117
202511111111118
202511111111119
2025111111111100
2025111111111101
2025111111111102
2025111111111103
2025111111111104
2025111111111105
2025111111111106
2025111111111107
2025111111111108
2025111111111109
2025111111111110
2025111111111111
2025111111111112
2025111111111113
2025111111111114
2025111111111115
2025111111111116
2025111111111117
2025111111111118
2025111111111119
20251111111111100
20251111111111101
20251111111111102
20251111111111103
20251111111111104
20251111111111105
20251111111111106
20251111111111107
20251111111111108
20251111111111109
20251111111111110
20251111111111111
20251111111111112
20251111111111113
20251111111111114
20251111111111115
20251111111111116
20251111111111117
20251111111111118
20251111111111119
202511111111111100
202511111111111101
202511111111111102
202511111111111103
202511111111111104
202511111111111105
202511111111111106
202511111111111107
202511111111111108
202511111111111109
202511111111111110
202511111111111111
202511111111111112
202511111111111113
202511111111111114
202511111111111115
202511111111111116
202511111111111117
202511111111111118
202511111111111119
2025111111111111100
2025111111111111101
2025111111111111102
2025111111111111103
2025111111111111104
2025111111111111105
2025111111111111106
2025111111111111107
2025111111111111108
2025111111111111109
2025111111111111110
2025111111111111111
2025111111111111112
2025111111111111113
2025111111111111114
2025111111111111115
2025111111111111116
2025111111111111117
2025111111111111118
2025111111111111119
20251111111111111100
20251111111111111101
20251111111111111102
20251111111111111103
20251111111111111104
20251111111111111105
20251111111111111106
20251111111111111107
20251111111111111108
20251111111111111109
20251111111111111110
20251111111111111111
20251111111111111112
20251111111111111113
20251111111111111114
20251111111111111115
20251111111111111116
20251111111111111117
20251111111111111118
20251111111111111119
202511111111111111100
202511111111111111101
202511111111111111102
202511111111111111103
202511111111111111104
202511111111111111105
202511111111111111106
202511111111111111107
202511111111111111108
202511111111111111109
202511111111111111110
202511111111111111111
202511111111111111112
202511111111111111113
202511111111111111114
202511111111111111115
202511111111111111116
202511111111111111117
202511111111111111118
202511111111111111119
2025111111111111111100
2025111111111111111101
2025111111111111111102
2025111111111111111103
2025111111111111111104
2025111111111111111105
2025111111111111111106
2025111111111111111107
2025111111111111111108
2025111111111111111109
2025111111111111111110
2025111111111111111111
2025111111111111111112
2025111111111111111113
2025111111111111111114
2025111111111111111115
2025111111111111111116
2025111111111111111117
2025111111111111111118
2025111111111111111119
20251111111111111111100
20251111111111111111101
20251111111111111111102
20251111111111111111103
20251111111111111111104
20251111111111111111105
20251111111111111111106
20251111111111111111107
20251111111111111111108
20251111111111111111109
20251111111111111111110
20251111111111111111111
20251111111111111111112
20251111111111111111113
20251111111111111111114
20251111111111111111115
20251111111111111111116
20251111111111111111117
20251111111111111111118
20251111111111111111119
202511111111111111111100
202511111111111111111101
202511111111111111111102
202511111111111111111103
202511111111111111111104
202511111111111111111105
202511111111111111111106
202511111111111111111107
202511111111111111111108
202511111111111111111109
202511111111111111111110
202511111111111111111111
202511111111111111111112
202511111111111111111113
202511111111111111111114
202511111111111111111115
202511111111111111111116
202511111111111111111117
202511111111111111111118
202511111111111111111119
2025111111111111111111100
2025111111111111111111101
2025111111111111111111102
2025111111111111111111103
2025111111111111111111104
2025111111111111111111105
2025111111111111111111106
2025111111111111111111107
2025111111111111111111108
2025111111111111111111109
2025111111111111111111110
2025111111111111111111111
2025111111111111111111112
2025111111111111111111113
2025111111111111111111114
2025111111111111111111115
2025111111111111111111116
2025111111111111111111117
2025111111111111111111118
2025111111111111111111119
20251111111111111111111100
20251111111111111111111101
20251111111111111111111102
20251111111111111111111103
20251111111111111111111104
20251111111111111111111105
20251111111111111111111106
20251111111111111111111107
20251111111111111111111108
20251111111111111111111109
20251111111111111111111110
20251111111111111111111111
20251111111111111111111112
20251111111111111111111113
20251111111111111111111114
20251111111111111111111115
20251111111111111111111116
20251111111111111111111117
20251111111111111111111118
20251111111111111111111119
202511111111111111111111100
202511111111111111111111101
202511111111111111111111102
202511111111111111111111103
202511111111111111111111104
202511111111111111111111105
202511111111111111111111106
202511111111111111111111107
202511111111111111111111108
202511111111111111111111109
202511111111111111111111110
202511111111111111111111111
202511111111111111111111112
202511111111111111111111113
202511111111111111111111114
202511111111111111111111115
202511111111111111111111116
202511111111111111111111117
202511111111111111111111118
202511111111111111111111119
2025111111111111111111111100
2025111111111111111111111101
20251111111
```

```

1908 \edef\gmu@reserveda{\unexpanded{\#2}\@xa\unexpanded{%
1909   \gmu@reserveda}}%
1910 \n@melet{\#1@head}{\gmu@reserveda}%
1911 }
1912 }% 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.
`1925 \@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 1930 \def\@afterheading{%
1931   \@nobreaktrue
1932   \xdef\gmu@prevsec{\mw@HeadingType}% added now
1933   \everypar{%
1934     \grelax\gmu@prevsec% added now. All the rest is original LATEX.
1935     \if@nobreak
1936       \nobreakfalse
1937       \clubpenalty\@M
1938       \if@afterindent\else
1939         {\setbox\z@\lastbox}%
1940       \fi
1941     \else
1942       \clubpenalty\clubpenalty
1943       \everypar{}%
1944     \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 preskip or, if defined, the two-header-skip. (We put the macro defined below just before `\addvspace` in `MWCLS` inner macros.)

```

\gmu@checkaftersec 1951 \def\gmu@checkaftersec{%
1952   \gm@ifundefined{\gmu@prevsec}{}{%
1953     \ifgmu@postsec% an additional switch that is true by default but may be
1954       turned into an \ifdim in special cases, see line 1989.
1955     {\@xa\mw@getflags\@xa{\gmu@prevsec}%
1956       \glet\gmu@reserveda\mw@HeadingBreakAfter}%
1957     \if\mw@HeadingBreakBefore\def\gmu@reserveda{\fi}% if the current
1958       heading inserts page break before itself, all the play with vskips is irrele-
1959       vant.
1960     \if\gmu@reserveda\else
1961       \penalty1000\relax
1962     \skip\z@=\csname\gmu@prevsec\postskip\endcsname\relax
1963     \skip\tw@=\csname\mw@HeadingType\preskip\endcsname\relax
1964     \gm@ifundefined{\mw@HeadingType\twoheadskip}{%
}
}

```

```

1966 \ifdim\skip\z@>\skip\tw@
1967 \vskip-\skip\z@% we strip off the post-skip of previous header if it's bigger
      than current pre-skip
1969 \else
1970 \vskip-\skip\tw@% we strip off the current pre-skip otherwise
1971 \fi}{% But if the two-header-skip is defined, we put it
1973 \penalty10000
1974 \vskip-\skip\z@
1975 \penalty10000
1976 \vskip-\skip\tw@
1977 \penalty10000
1978 \vskip\csname\mw@HeadingType\@twoheadskip\endcsname
1979 \relax}%
1980 \penalty10000
1981 \hrule\height\z@\relax% to hide the last (un)skip before
      subsequent \addvspaces.
1983 \penalty10000
1984 \fi
1985 \fi
1986 }% of \gm@ifundefined{\gmu@prevsec} 'else'.
1987 }% of \def\gmu@checkaftersec.

\ParanoidPostsec 1989 \def\ParanoidPostsec{%
      this version of \ifgmu@postsec is intended for the spe-
      cial case of sections may contain no normal text, as while gmdocing.
\ifgmu@postsec 1992 \def\ifgmu@postsec{%
      note this macro expands to an open \if.
      \skip\z@=\csname\gmu@prevsec\@postskip\endcsname\relax
      \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).
1998     }%
2000 \let\ifgmu@postsec\iftrue
\gmu@getaddvs 2002 \def\gmu@getaddvs#1\addvspace#2\gmu@getaddvs{%
2003   \toks\z@={#1}%
2004   \toks\tw@={#2}}

```

And the modification of the inner macros at last:

```

\gmu@setheading 2007 \def\gmu@setheading#1{%
2008   \xa\gmu@getaddvs#1\gmu@getaddvs
2009   \edef#1{%
2010     \the\toks\z@\@nx\gmu@checkaftersec
2011     \@nx\addvspace\the\toks\tw@}}
2013 \gmu@setheading\mw@normalheading
2014 \gmu@setheading\mw@runinheading
\SetTwoheadSkip 2016 \def\SetTwoheadSkip#1#2{\@namedef{#1@twoheadskip}{#2}}
2018 }% 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.

```

2030 \@ifnotmw{}{%
2031   \def\WPheadings{%
2032     \SetSectionFormatting[breakbefore,wholewidth]
2033       {part}{\z@\@plus1fill}{}{\z@\@plus3fill}%
2035     \gma@ifundefined{chapter}{}{%
2036       \SetSectionFormatting[breakbefore,wholewidth]
2037         {chapter}
2038       {66\p@}{67\p@} for Adventor/Schola o,95.
2039       {\FormatHangHeading{\LARGE}}
2040       {27\p@\@plus0,2\p@\@minus1\p@}%
2041     }%
2043     \SetTwoheadSkip{section}{27\p@\@plus0,5\p@}%
2044     \SetSectionFormatting{section}
2045       {24\p@\@plus0,5\p@\@minus5\p@}%
2046       {\FormatHangHeading{\Large}}
2047       {10\p@\@plus0,5\p@}% ed. Krajewska of "Wiedza Powszechna", as we un-
2048         derstand her, wants the skip between a heading and text to be rigid.
2049     \SetTwoheadSkip{subsection}{11\p@\@plus0,5\p@\@minus1\p@}%
2050     \SetSectionFormatting{subsection}
2051       {19\p@\@plus0,4\p@\@minus6\p@}
2052       {\FormatHangHeading{\large}}% 12/14 pt
2053       {6\p@\@plus0,3\p@}% after-skip 6 pt due to p.12, not to squeeze the before-
2054         skip too much.
2055     \SetTwoheadSkip{subsubsection}{10\p@\@plus1,75\p@\@minus1\p@}%
2056     \SetSectionFormatting{subsubsection}
2057       {10\p@\@plus0,2\p@\@minus1\p@}
2058       {\FormatHangHeading{\normalsize}}
2059       {3\p@\@plus0,1\p@}% those little skips should be smaller than you calcu-
2060         late out of a geometric progression, because the interline skip enlarges
2061         them.
2062       \SetSectionFormatting[runin]{paragraph}
2063         {7\p@\@plus0,15\p@\@minus1\p@}
2064         {\FormatRunInHeading{\normalsize}}
2065         {2\p@}%
2066       \SetSectionFormatting[runin]{ subparagraph}
2067         {4\p@\@plus1\p@\@minus0,5\p@}
2068         {\FormatRunInHeading{\normalsize}}
2069         {\z@}%
2070     }% of \WPheadings
2071   }% 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.)

2117 \@ifnotmw{ we are not in mwcls and want to handle mwcls-like sectionings i.e.,
those written with two optionals.

```
\gm@secini 2120 \def\gm@secini{\gm@la}%
\gm@secxx 2122 \def\gm@secxx{\#1\#2[\#3]\#4}{%
2123   \ifx\gm@secstar\empty
2124     \n@melet{\gm@true@{\#1mark}}{\#1mark}%
2125       a little trick to allow a special ver-
2126       sion of the heading just to the running head.
2127     \cname\gm@true@{\#1mark}\endcsname{\#2}%
2128     \n@melet{\#1mark}{\gm@true@{\#1mark}}%
2129       after we've done what we
2130       wanted we restore original \#1mark.
2131   }%
2132 }
```

2133 \def\gm@secstar{\#3}%
2134 if \gm@secstar is empty, which means the sec-
2135 tioning command was written starless, we pass the 'true' sectioning
2136 command #3 as the optional argument. Otherwise the sectioning com-
2137 mand was written with star so the 'true' s.c. takes no optional.

```
2138 \fi
2139 \@xa\@xa\cname\gm@secini\#1\endcsname
2140 \gm@secstar{\#4}%
2141 }{ we are in mwcls and want to reverse MW's optionals order i.e., if there's just one
2142 optional, it should go both to toc and to running head.
```

```
\gm@secini 2145 \def\gm@secini{\gm@mw}%
2146 \let\gm@secmarkh\gobble%
2147 in mwcls there's no need to make tricks for special
2148 version to running headings.
```

```
\gm@secxx 2150 \def\gm@secxx{\#1\#2[\#3]\#4}{%
2151   \@xa\@xa\cname\gm@secini\#1\endcsname
2152   \gm@secstar{\#2}{\#3}{\#4}%
2153 }
```

2154 \def\gm@sec{\@dblarg{\gm@secx{\#1}}}
2155 \def\gm@secx{\#1[\#2]{%
2156 \@ifnextchar[\{\gm@secxx{\#1}{\#2}\}{\gm@secxx{\#1}{\#2}[\#2]}}%
2157 if there's
2158 only one optional, we double it not the mandatory argument.

```
\gm@straightensec 2161 \def\gm@straightensec{\#1}{ the parameter is for the command's name.
2162 \gm@ifundefined{\#1}{%
2163   we don't change the ontological status of the com-
2164   mand because someone may test it.
```

```

2164 \n@melet{\gm@secini#1}{#1}%
2165 \cnamedef{#1}{%
2166   \@ifstar{\def\gm@secstar{*}}{\gm@sec{#1}}{%
2167     \def\gm@secstar{}{\gm@sec{#1}}}}%
2168 }%
2170 \let\do\gm@straightensec
2171 \do{part}\do{chapter}\do{section}\do{subsection}\do{%
2172   subsubsection}%
2172 \cifnotmw{}{\do{paragraph}}% this ‘straightening’ of \paragraph with the stan-
dard article caused the ‘ $\text{T}_{\text{E}}\text{X}$  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 $\text{L}\text{A}\text{T}_{\text{E}}\text{X}$'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* 2188 \cnamedef{enumerate*}{%
2189   \ifnum\@enumdepth>\thr@@
2190     \atodeep
2191   \else
2192     \advance\@enumdepth\@ne
2193     \edef\@enumctr{enum\romannumeral\the\@enumdepth}%
2194     \xa\list\csname_label\@enumctr\endcsname{%
2195       \partopsep\topsep\topsep\z@\leftmargin\z@
2196       \itemindent\parindent\% \advance\itemindent\labelsep
2197       \labelwidth\parindent
2198       \advance\labelwidth-\labelsep
2199       \listparindent\parindent
2200       \usecounter\@enumctr
2201       \def\makelabel##1{\hfil}%
2202     \fi}
2203   \cnamedef{endenumerate*}{\endlist}
itemize* 2206 \cnamedef{itemize*}{%
2207   \ifnum\@itemdepth>\thr@@
2208     \atodeep
2209   \else
2210     \advance\@itemdepth\@ne
2211     \edef\@itemitem{labelitem\romannumeral\the\@itemdepth}%
2212     \xa\list\csname@itemitem\endcsname{%
2213       \partopsep\topsep\topsep\z@\leftmargin\z@
2214       \itemindent\parindent
2215       \labelwidth\parindent
2216       \advance\labelwidth-\labelsep
2217       \listparindent\parindent
2218       \def\makelabel##1{\hfil}%
2219     \fi}
2220   \cnamedef{enditemize*}{\endlist}

```

The logos

We'll modify The L^AT_EX logo now to make it fit better to various fonts.

```

2229 \let\oldLaTeX\LaTeX
2230 \let\oldLaTeXe\LaTeXe
2232 \def\TeX{T\kern-.1667em\lower.5ex\hbox{E}\kern-.125emX\@}
2234 \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 2242 \ifx\relax#1\def\gmu@reserveda{\@xa\@gobble\string#2}%
\else
\def\gmu@reserveda{#1}%
\fi
\edef\gmu@reserveda{%
    \nx\DeclareTextCommand\@nx#2{PD1}{\gmu@reserveda}}
\gmu@reserveda
\DeclareTextCommandDefault#2{#3}%
\pdef 2250 \pdef#2{#3}}% added for XHTEX

\DeclareLogo 2253 \DeclareLogo\LaTeX{%
{%
L%
\setbox\z@\hbox{\check@mathfonts
\fontsize\sf@size\z@
\math@fontsfalse\selectfont
A}%
\kern-.57\wd\z@
\sbox\tw@\T%
\vbox\to\ht\tw@{\copy\z@\vss}%
\kern-.2\wd\z@}% originally -,15 em for T.
{%
\ifdim\fontdimen1\font=\z@
\else
\count\z@=\fontdimen5\font
\multiply\count\z@ by 64\relax
\divide\count\z@ by \p@
\count\tw@=\fontdimen1\font
\multiply\count\tw@ by \count\z@
\divide\count\tw@ by 64\relax
\divide\count\tw@ by \tw@
\kern-\the\count\tw@sp\relax
\fi}%
\TeX}
\LaTeXe 2279 \DeclareLogo\LaTeXe{\mbox{\m@th\if
b\expandafter\@car\f@series\@nil\boldmath\fi
\LaTeX\kern.15em\$_{\{\textstyle\varepsilon\}}\$}}
2283 \StoreMacro\LaTeX
2284 \StoreMacro*\{LaTeX\}

```

'(L)TeX' in my opinion better describes what I work with/in than just 'L^AT_EX'.

```

\LaTeXpar 2290 \DeclareLogo[(La)TeX]{\LaTeXpar}{%
2291   {%
2292     \setbox\z@\hbox{()%
2293     \copy\z@%
2294     \kern-.2\wd\z@\L%
2295     \setbox\z@\hbox{\check@mathfonts%
2296       \fontsize\sf@size\z@%
2297       \math@fontsfalse\selectfont%
2298       A}%
2299     \kern-.57\wd\z@%
2300     \sbox\tw@\T%
2301     \vbox\to\ht\tw@\{\box\z@%
2302     \vss}%
2303   }%
2304   \kern-.07em% originally -, 15 em for T.
2305   {%
2306     \sbox\z@%
2307     \kern-.2\wd\z@\copy\z@%
2308     \kern-.2\wd\z@\TeX%
2309   }

```

“Here are a few definitions which can usefully be employed when documenting package files: now we can readily refer to *\mathcal{AMSTeX}* , *\mathcal{BibTeX}* and *\mathcal{SliTeX}* , as well as the usual \TeX and \LaTeX . There’s even a *PLAIN \TeX* and a *WEB*.”

```

\AmSTeX 2316 \gm@ifundefined{AmSTeX}
2317   {\def\AmSTeX{\leavevmode\hbox{$\mathcal{A}\kern-.2em$}%
2318     \lower.376ex\hbox{$\mathcal{M}\kern-.2em\mathcal{S}$}-\TeX}}{}}

\BibTeX 2320 \DeclareLogo\BibTeX{{\rmfamily\B\kern-.05em}%
2321   \textsc{i{\kern-.025em}b}\kern-.08em% the kern is wrapped in braces
2322   for my \fakesc's sake.%
2323   \TeX}

\SliTeX 2326 \DeclareLogo\SliTeX{{\rmfamily\kern-.06em\kern-.18em\kern-.18em}%
2327   \raise.32ex\hbox{\kern-.03em\TeX}}}

\PlainTeX 2329 \DeclareLogo\PlainTeX{\textsc{Plain}\kern2pt\TeX}

\Web    2331 \DeclareLogo\Web{\textsc{Web}}

```

There’s also the *(La)TeX* logo got with the \LaTeXpar macro provided by *gmutils*. And here *The \TeX book*’s logo:

```

\TeXbook 2334 \DeclareLogo[\TeXbook]\TeXbook{\textsl{The\TeXbook}}
2335 \let\TB\TeXbook% TUG Boat uses this.

\eTeX   2337 \DeclareLogo[e-\TeX]\eTeX{%
2338   \ensuremath{\varepsilon}-\kern-.125em\TeX}% definition sent by Karl Berry
         from TUG Boat itself.

\pdfTeX 2341 \DeclareLogo[pdf-\TeX]\pdfTeX{pdf\TeX}

\pdfTeX 2343 \DeclareLogo\pdfTeX{pdf\TeX}

\pdfTeX 2345 \gm@ifundefined{XeTeX}{%
2346   \DeclareLogo\XeTeX{\kern-.125em\relax%
2347   \gm@ifundefined{reflectbox}{%
```

```

2348      \lower.5ex\hbox{E}\kern-.1667em\relax}%
2349      \lower.5ex\hbox{\reflectbox{E}}\kern-.1667em\relax}%
2350  \TeX}{}{}}
2352 \gm@ifundefined{XeLaTeX}{%
2353   \DeclareLogo{XeLaTeX}{\kern-.125em\relax
2354   \gm@ifundefined{\reflectbox}{%
2355     \lower.5ex\hbox{E}\kern-.1667em\relax}%
2356     \lower.5ex\hbox{\reflectbox{E}}\kern-.1667em\relax}%
2357  \LaTeX}{}{}
```

As you see, if \TeX doesn't recognize `\reflectbox` (graphics isn't loaded), the first E will not be reversed. This version of the command is intended for non- \TeX usage. With \TeX , you can load the `xltextra` package (e.g. with the `gmutils \XeTeXthree` declaration) and then the reversed E you get as the Unicode Latin Letter Reversed E.

Expandable turning stuff all into ‘other’

While typesetting a unicode file contents with `inputenc` package I got a trouble with some Unicode sequences that expanded to unexpandable cses: they could'nt be used within `\csname ... \endcsname`. My \TeX Guru advised to use `\meanig` to make all the name ‘other’. So—here we are.

Don't use them in `\edefs`, they would expand not quite.

The next macro is intended to be put in `\edefs` with a macro argument. The meaning of the macro will be made all ‘other’ and the words ‘(long) macro:->’ gobbled.

```
\all@other 2382 \long\def\all@other#1{\@xa\gm@gobmacro\meaning#1}
```

The `\gm@gobmacro` macro above is applied to gobble the `\meaning`'s beginnig, `long macro:->` all ‘other’ that is. Use of it:

```
\gm@gobmacro 2387 \edef\gmu@tempa{%
2388   \def\@nx\gm@gobmacro##1\@xa\@gobble\string\macro##2->{}}
2389 \gmu@tempa
```

Brave New World of \TeX

```
\@ifXeTeX 2406 \newcommand{\@ifXeTeX}[2]{%
2407   \ifdefinable{XeTeXversion}{%
2408     \unless\ifx\XeTeXversion\relax\afterfifi{#1}\else\afterfifi{%
2409       #2}\fi
2410     \else\afterfi{#2}\fi}
2412 \def\XeTeXthree{%
2414   \@ifXeTeX{%
2415     \ifpackageloaded{gmverb}{\StoreMacro{\verb}{}}%
2416     \RequirePackage{xltextra}% since v 0.4 (2008/07/29) this package redefines \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.
2423     \ifpackageloaded{gmverb}{\RestoreMacro{\verb}{}}%
2424     \AtBeginDocument{%
2425       \RestoreMacro{\LaTeX}\RestoreMacro*{\LaTeX}{}% my version of the
2428       \}}}}
```

The `\udigits` declaration causes the digits to be typeset uppercase. I provide it since by default I prefer the lowercase (nautical) digits.

```

2433 \AtBeginDocument{%
2434   \@ifpackageloaded{fontspec}{%
2435     \pdef\udigits{%
2436       \addfontfeature{Numbers=Uppercase}}%
2437   }{%
2438     \emptyify\udigits}%

```

Fractions

`\Xedekfracc` 2443 `\def\Xedekfracc{\@ifstar\gmu@xedekfraccstar\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 `\@namedefs` 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 2458 \def\gmu@xedekfraccstar{%
\gmu@xefraccdef 2459   \def\gmu@xefraccdef##1##2{%
2460     \iffontchar\font_{##2}%
2461       \gmu@xefracc##1{\char##2}_{##2}%
2462     \else%
2463       \n@melet{\gmu@xefracc##1}{relax}%
2464     \fi}%
2465   \def\gmu@dekfracc##1##2{%
2466     \addfontfeature{VerticalPosition=Numerator}##1%
2467       \gmu@numeratorkern%
2468       \char"2044_{\gmu@denominatorkern%
2469         \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.

```

2473   \gmu@xefraccdef{1/4}{\"BC}%
2474   \gmu@xefraccdef{1/2}{\"BD}%
2475   \gmu@xefraccdef{3/4}{\"BE}%
2476   \gmu@xefraccdef{1/3}{\"2153}%
2477   \gmu@xefraccdef{2/3}{\"2154}%
2478   \gmu@xefraccdef{1/8}{\"215B}%
2479   \gmu@xefraccdef{3/8}{\"215C}%
2480   \gmu@xefraccdef{5/8}{\"215D}%
2481   \gmu@xefraccdef{7/8}{\"215E}%
2482   \pdef\dekfracc@args##1##2{%
2483     \def\gm@duppa##1##2{%
2484       \gm@ifundefined{\gmu@xefracc\all@other\gm@duppa}{%
2485         \gmu@dekfracc##1/{##2}}{%
2486           \csname\gmu@xefracc\all@other\gm@duppa\endcsname}%
2487           \if@gmu@mmbbox\egroup\fi%
2488         }% of \dekfracc@args.
2489       \@ifstar{\let\gmu@dekfracc\gmu@dekfraccsimple}{}%

```

```

2490      }
2491 \def\gmu@xedekfraccplain{\% 'else' of the main \@ifstar
2492   \pdef\dekfracc@args##1##2{%
2493     \ifmmode\hbox\fi{%
2494       \addfontfeature{Fractions=On}%
2495       ##1##2}%
2496     \if@gmu@mmhbox\egroup\fi
2497   }% of \dekfracc@args
2498 }
2499 }

\if@gmu@mmhbox 2501 \newif\if@gmu@mmhbox% we'll use this switch for \dekfracc and also for \thous
               (hacky thousand separator).

\dekfracc 2504 \pdef\dekfracc{%
2505   \ifmmode\hbox\bgroup@gmu@mmhboxtrue\fi
2506   \dekfracc@args}
2507 }

\gmu@numeratorkern 2510 \def\gmu@numeratorkern{\kern-.05em\relax}
2511 \let\gmu@denominatorkern\gmu@numeratorkern

```

What have we just done? We defined two versions of the `\Xefractions` 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 `\@ifstar` we wrote 8 (eight) `#`s to get the correct definition and in the second argument 'only' 4. (The L^AT_EX 2_E Source claims that that is changed in the 'new implementation' of `\@ifstar` so maybe it's subject to change.)

A simpler version of `\dekfracc` is provided in line [3070](#).

```

\resizegraphics

\resizegraphics 2533 \def\resizegraphics#1#2#3{%
2534   \resizebox{#1}{#2}{%
2535     \includegraphics{#3}}}

\GMtextsuperscript 2539 \def\GMtextsuperscript{%
2540   \@ifXeTeX{%
2541     \def\textsuperscript##1{%
2542       \addfontfeature{VerticalPosition=Numerator}##1}%
2543   }{\truetextsuperscript}}
2544 }

\truetextsuperscript 2545 \def\truetextsuperscript{%
2546   \pdef\textsuperscript##1{%
2547     \@textsuperscript{\selectfont##1}}%
2548 }

\@textsuperscript 2548 \def\@textsuperscript##1{%
2549   {\m@th\ensuremath{\hat{\mbox{\scriptsize\sf\size{z@##1}}}}}}}

```

Settings for mathematics in main font

`\gmath` 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 2562 \pdef\gmath{%

```

```

2563 \everymath{%
2564   \relaxen\do
2565     \newcommand*\do[4][\mathit]{\def##2##3##1{\char"##4}}}}%
2566   \do\alpha{}{\o3B1}%
2567   \do[\mathrm]\Delta{}{\o394}%
2568   \do\varepsilon{}{\o3B5}%
2569   \do\vartheta{}{\o3D1}%
2570   \do\nu{}{\o3BD}%
2571   \do\pi{}{\o3Co}%
2572   \do\phi{}{\o3D5}%
2573   \do[\mathrm]\Phi{}{\o424}%
2574   \do\sigma{}{\o3C3}%
2575   \do\varsigma{}{\o3DA}%
2576   \do\psi{}{\o3C8}%
2577   \do\omega{}{\o3C9}%
2578   \do\infty{}{\o21E}%
2579   \do[\mathrm]\neg{\mathbin}{ooAC}%
2580   \do[\mathrm]\neq{\mathrel}{2260}%
2581   \do\partial{}{\o202}%
2582   \do[\mathrm]\pm{\mathbin}{ooB1}%
2583   \do[\mathrm]\pm{\mathbin}{ooB1}%
2584   \do[\mathrm]\sim{\mathrel}{oo7E}%
2585   \def\do##1##2##3{\def##1{%
2586     \mathop{\mathchoice{\hbox{%
2587       \rm
2588       \edef\gma@tempa{\the\fontdimen8\font}%
2589       \larger[3]%
2590       \lower\dimexpr(\fontdimen8\font-\gma@tempa)/2\hbox{##2}}}\hbox{%
2591       \rm
2592       \edef\gma@tempa{\the\fontdimen8\font}%
2593       \larger[2]%
2594       \lower\dimexpr(\fontdimen8\font-\gma@tempa)/2\hbox{##2}}}}%
2595     {\mathrm{##2}}{\mathrm{##2}}}}##3}%
2596   \do\sum{\char"2211}{}%
2597   \do\forall{\gma@quantifierhook\rotatebox[origin=c]{180}{A}}%
2598     \setboxo=\hbox{A}\setboxz=\hbox{\scriptsize x}%
2599     \kern\dimexpr\htz/3*2-\wdz/2\relax\{\nolimits}%
2600   \do\exists{\rotatebox[origin=c]{180}{\gma@quantifierhook E}}%
2601     \nolimits%
2602   \def\do##1##2##3{\def##1{##3}%
2603     \mathchoice{\hbox{\rm##2}}{\hbox{\rm##2}}{\hbox{\rm\tiny##2}}}}%
2604     \do\vee{\rotatebox[origin=c]{90}{<}}\mathbin
2605     \do\wedge{\rotatebox[origin=c]{-90}{<}}\mathbin
2606     \do\leftarrow{\char"2190}\mathrel
2607     \do\rightarrow{\char"2192}\mathrel
2608     \do\leftrightarrow{\char"2190\kern-0.1em\char"2192}\mathrel
2609     \gmu@storespecials[\do\`do\"do\=do]=%
2610     \gmu@septify[\do\`12\do\"12\do\=12]%
2611   \def\do##1##2##3{%
2612     \catcode`\##1=12\relax\% to ensure ##2 be 'other' in the definition body.

```

```

2620 \scantokens{\mathcode`##1="8000\relax
2621   \foone{\catcode`##1=\active}{\def##1}{##3{%
2622     \mathchoice{\hbox{\rm##2}}{\hbox{\rm##2}}{%
2623       {\hbox{\rm\scriptsize##2}}{\hbox{\rm\tiny##2}}}}}}%
2624   \ignorespaces}}% to eat the lineend (scantokens acts as \read including
2625   line end).
2626 \do..\mathpunct\do,,\mathpunct\do....\mathpunct
2627 \do((\mathopen%
2628 \gm@ifundefined{resetMathstrut@}{}{%
2629   an error occured 'bad mathchar
2630   etc.' because amsmath.sty doesn't take account of a possibility of '(' being
2631   math-active.
2632 \def\resetMathstrut@{%
2633   \setbox\z@\hbox{%
2634     \mathchardef\@tempa\mathcode`\\(\relax
2635     \def\@tempb##1"##2##3{\the\textfont"##3\char"}%
2636     \expandafter\@tempb\meaning\@tempa\relax
2637     (%
2638   }%
2639   \ht\Mathstrutbox@\ht\z@\dp\Mathstrutbox@\dp\z@
2640   })}%
2641 \do))\mathclose
2642 \do[[\mathopen\do]]\mathclose
2643 \do-\{\char"2212}\mathbin\do+\mathbin\do==\mathrel\%
2644 \do\times\mathbin
2645 \do:\mathbin\do\cdot\mathbin\do/\mathbin\do<<\mathrel
2646 \do>>\mathrel
2647 \dooo\mathord\do11\mathord\do22\mathord\do33\mathord
2648 \do44\mathord\do55\mathord\do66\mathord\do77\mathord
2649 \do88\mathord\do99\mathord
2650 \gmu@restorespecials
2651 \def\do##1##2##3{\def##1####1{##2{\hbox{%
2652   \rm
2653   \setboxo=\hbox{####1}%
2654   \edef\gma@tempa{\the\hto}%
2655   \edef\gma@tempb{\the\dpo}%
2656   ##3%
2657   \setboxo=\hbox{####1}%
2658   \lower\dimexpr(\hto+\dpo)/2-\dpo-((\gma@tempa+%
2659   \gma@tempb)/2-\gma@tempb)\hbox{%
2660   \boxo}}}%
2661 \do\bigr\mathopen\larger
2662 \do\bigr\mathclose\larger
2663 \do\Bigl\mathopen\largerr
2664 \do\Bigr\mathclose\largerr
2665 \do\biggl\mathopen\larger[3]%
2666 \do\biggr\mathclose\larger[3]%
2667 \do\Biggl\mathopen\larger[4]%
2668 \do\Biggr\mathclose\larger[4]%
2669 \def\do##1##2{\def##1{\ifmmode##2{\mathchoice
2670   {\hbox{\rm\char`##1}}{\hbox{\rm\char`##1}}{%
2671     {\hbox{\rm\scriptsize\char`##1}}{\hbox{\rm\tiny%
2672       \char`##1}}}}}}%
2673 \else\char`##1\fi}%

```

```

2674 \do{\{\mathopen
2675 \do{\}\mathclose
2677 \def\={\mathbin{=}}%
2678 \def\neqb{\mathbin{\neq}}%
2679 \def\do##1{\edef\gma@tempa{%
2680     \def\x@nx\csname\x@xa\gobble\string##1r\endcsname{%
2681         \x@nx\mathrel{\x@nx##1}}}}%
2682     \gma@tempa}%
2683 \do\vee\do\wedge\do\neg
2684 \def\fakern{\mkern-3mu}%
2685 \thickmuskip=8mu plus 4mu\relax
2687 \gma@gmathhook
2688 }% of \everymath.
2689 \everydisplay\everymath
2690 \ifdefined\Url
2691     \ampulexdef\Url{\let\do}{\makeother
2692     \everymath{}{\let\do}{\makeother}}% I don't know why but the url package's
2693     % \url typesets the argument inside a math which caused digits not to
2694     be typewriter but Roman and lowercase.
2695     \fi
2696 }% of \def\gmath.
2697 \emptyify\gma@quantifierhook
2698 \def\quantifierhook#1{%
2701     \def\gma@quantifierhook{#1}}
2703 \emptyify\gma@gmathhook
2704 \def\gmathhook#1{\addtomacro\gma@gmathhook{#1}}
2707 \def\gma@dollar##1${{\gmath##1$}}%
2708 \def\gma@bare#1{\gma@dollar##1$}%
2709 \def\gma@checkbracket{\ifnextchar[%
2710     \gma@bracket\gma@bare}
2711 \def\gma@bracket[#1]{\gmath[#1]}\ifnextchar\par{}{%
2712     \noindent}}
2712 \def\gma{\ifnextchar$%
2713     \gma@dollar\gma@checkbracket}
2719 \def\garamath{%
2720     \addtotoks\everymath{%
2721         \quantifierhook{\addfontfeature{OpticalSize=800}}%
2722     \def\gma@arrowdash{%
2723         \setboxo=\hbox{\char"2192}\copyo\kern-0.6\wdo
2724         \bgcolor\rule[-\dpo]{0,6\wdo}{\dimexpr\hto+\dpo}%
2725             \kern-0.6\wdo}}%
2727     \def\gma@gmathhook{%
2728         \def\do####1####2####3{\def####1{####3}{%
2729             \mathchoice{\hbox{\rm####2}}{\hbox{\rm####2}}{\hbox{\rm####2}}{%
2730                 \hbox{\rm\scriptsize####2}}}}%
2731         \do\mapsto{\rule[0,4ex]{0,1ex}{0,4ex}\kern-0.05em}%
2732             \gma@arrowdash\kern-0.05em\char"2192\mathrel
2733             \do\cup{\scshape u}\mathbin
2734             \do\varnothing{\setboxo=\hbox{\gma@quantifierhook}%
2734                 \addfontfeature{Scale=1.272727}o}}%

```

```

2735 \setbox2=\hbox{\char"2044}%
2736 \copyo\kern-0.5\wdo\kern-0.5\wd2\lower0.125\wdo\copy2
2737 \kerno.5\wdo\kern-0.5\wd2}{}}%
2738 \do\leftarrow{\char"2190\kern-0.05em\gma@arrowdash}\mathrel
2739 \do\rightarrow{\gma@arrowdash\kern-0.05em\char"2192}}%
2740 \mathrel
2741 \do\in{\gma@quantifierhook\char"0454}\mathbin
2742 \}}%
2742 \everydisplay\everymath}

```

Minion and Garamond Premier kerning and ligature fixes

»Ws« shall not make long »s« because long »s« looks ugly next to »W«.

```

\gmu@tempa 2750 \def\gmu@tempa{\kern-0.08em\penalty10000\hskip0.5pt\relax
2751   s\penalty10000\hskip0.5pt\relax}
2753 \protected\edef\V{\gmu@tempa}
2755 \protected\edef\W{\gmu@tempa}
\Wz 2757 \pdef\Wz{W\kern-0.05em\penalty10000\hskip0.5pt\relax_z}

```

Varia

A very neat macro provided by doc. I copy it *verbatim*.

```

\gmu@tilde 2766 \def\gmu@tilde{%
2767   \leavevmode\lower.8ex\hbox{$\backslash$\widetilde{\mbox{}$}\backslash$, $}}}

```

Originally there was just `\`` instead of `\widetilde{\mbox{}$}` but some commands of ours do redefine `\``.

```

\* 2771 \pdef\*{\gmu@tilde}
2777 \AtBeginDocument{\% to bypass redefinition of \` as a text command with various
      encodings
\texttilde 2779 \pdef\texttilde{%
2782   \@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 13 (‘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 `\activeate` the space but also will (re)define it as the `\`` primitive. So define `\gmobeyspaces` that obeys this requirement.

(This definition is repeated in `gmverb`.)

```

\gmobeyspaces 2794 \foone{\catcode`\`=\active}%
2795 {\def\gmobeyspaces{\let`\`=\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.

```

\defobeylines 2802 \foone{\catcode`\^^M=\active}%
2803 {\def\defobeylines{\catcode`\^^M=13\def^^M{\par}}}

```

Another thing I dislike in L^AT_EX yet is doing special things for `\dotskip`’s, ’cause I like the Knuthian simplicity. So I sort of restore Knuthian meanings:

```

\deksmallskip 2812 \def\deksmallskip{\vskip\smallskipamount}

```

```

\undecksmallskip 2813 \def\undecksmallskip{\vskip-\smallskipamount}
\dekmedskip     2814 \def\dekmedskip{\vskip\medskipamount}
\dekbigskip     2815 \def\dekbigskip{\vskip\bigskipamount}
\hfillneg      2818 \def\hfillneg{\hskip\opt\plus-\ifill\relax}

```

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 2826 \long\def@firstofmany#1#2@@nil{#1}

```

A mark for the **TODO!**s:

```

\TODO 2830 \newcommand*\TODO[1][]{{%
 2831   \sffamily\bfseries\huge\textcolor{red}{TODO!}\if\relax#1\relax\else\space%
 2832   \fi#1}}

```

I like twocolumn tables of contents. First I tried to provide them by writing `\begin{multicols}{2}` and `\end{multicols}` outto the .toc file but it worked wrong in some cases. So I redefine the internal L^AT_EX macro instead.

```

\twocoltoc 2866 \newcommand*\twocoltoc{%
 2867   \RequirePackage{multicol}%
@starttoc 2868 \def@starttoc##1{%
 2869   \begin{multicols}{2}\makeatletter\@input{\jobname.\#\#1}%
 2870   \if@filesw\@xa\newwrite\csname\@name\@ext\endcsname
 2871   \immediate\openout\csname\@name\@ext\endcsname\jobname%
 2872   .\#\#1\relax
 2873   \fi
 2874   \nobreakfalse\end{multicols}}}
 2875 \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 2880 \newcommand*\enoughpage[1]{%
 2881   \par
 2882   \dimeno=\pagegoal
 2883   \advance\dimeno by-\pagetotal
 2884   \ifdim\dimeno<\#1\relax\newpage\fi}

```

An equality sign properly spaced:

```

>equals 2893 \pdef\equals{\hunskip${}={}$\ignorespaces}

```

And for the L^AT_EX's pseudo-code statements:

```

\eequals 2895 \pdef\eequals{\hunskip${}==${}\ignorespaces}
\cdot 2897 \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 2917 \def\freeze@actives{%
 2918   \count\z@\z@
 2920   \@whilenum\count\z@<\@cclvi\do{%
 2921     \ifnum\catcode\count\z@=\active
 2922       \uccode`~=\count\z@
 2923       \uppercase{\let~\relax}%
 2924     \fi
 2925   \advance\count\z@\@ne}}

```

A macro that typesets all 256 chars of given font. It makes use of `\@whilenum`.

```

>ShowFont 2931 \newcommand*{\ShowFont}[1][6]{%
 2932   \begin{multicols}{#1}[The current font (the \f@encoding \u
 2933   encoding):]
 2934   \parindent\z@
 2935   \count\z@\m@ne
 2936   \@whilenum\count\z@<\@cclv\do{%
 2937     \advance\count\z@\@ne
 2938     \u\the\count\z@:\~\char\count\z@\par}
 2939   \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 2946 \newcommand*{\liturgiques}[1][red]{% Requires the color package.
 2947   \gmu@RPfor{color}\color%
 \czerwo 2948 \newcommand*{\czerwo}{\small\color{#1}}% environment
 \czer 2949 \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.
 2950   \def\*{\czer{$*\$}}
 2951   \def\+{\czer{$\dag\$}}
 \nieczer 2954 \newcommand*{\nieczer}[1]{\textcolor{black}{##1}}%

```

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 2959 \newcommand*{\gmu@RPfor}[3][]{%
 2960   \ifx\relax#1\relax
 \gmu@resa 2962 \else\def\gmu@resa{[#1]}%
 2963   \fi
 2964   \o@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 2970 \AtBeginDocument{%
 2971   \renewcommand*{\gmu@RPfor}[3][]{%
 2972     \unless\ifdefined#3%
 2973       \@ifpackageloaded{#2}{}{%
 2974         \typeout{^^J! \Package#2 not loaded!!! (\on@line)^^J}%
 2975       \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 2981 \pprovide\continuum{%
 2982   \gmu@RPfor{eufrak}\mathfrak\ensuremath{\mathfrak{c}}}}

```

And this macro I saw in the *ltugproc* document class nad I liked it.

```

\iteracro 2986 \def\iteracro{%
\acro 2987   \pdef\acro##1{\gmu@acrospace##1\gmu@acrospace}%
2988 }
2990 \iteracro
\gmu@acrospace 2992 \def\gmu@acrospace#1#2\gmu@acrospace{%
2993   \gmu@acroinner#1\gmu@acroinner
2994   \ifx\relax#2\relax\else
2995     \space
2996     \afterfi{\gmu@acrospace#2\gmu@acrospace}% when #2 is nonempty, it
           is ended with a space. Adding one more space in this line resulted in an
           infinite loop, of course.
3000   \fi}
\gmu@acroinner 3003 \def\gmu@acroinner#1{%
3004   \ifx\gmu@acroinner#1\relax\else
3005     \ifcat_a\@nx#1\relax%
3006       \ifnum`#1=\uccode`#1%
3007         {\acrocore{#1}}%
3008         \else{#1}% tu bylo \smallerr
3009         \fi
3010       \else{#1}%
3011       \fi
3012       \afterfi\gmu@acroinner
3013   \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.

```
\acrocore 3017 \def\acrocore{\scshape\lowercase}
```

Since the fonts I am currently using do not support required font feature, I skip the following definition.

```
\IMO 3022 \newcommand*\IMO{\acro{IMO}}
\AKA 3023 \newcommand*\AKA{\acro{AKA}}
\usc 3025 \pdef\usc#1{{\addfontfeature{Letters=UppercaseSmallCaps}#1}}
\uscacro 3027 \def\uscacro{\let\acro\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 3045 \def\secondclass{%
\ifSecondClass 3046   \newif\ifSecondClass
3047   \SecondClasstrue
3048   \c@fileswithoptions\c@clsextension% [outeroff,gmeometric]{gmdocc}
           it's loading gmdocc.cls with all the bells and whistles except the error message.
```

Cf. *The TeXbook* exc. 11.6.

A line from L^AT_EX:

```
%\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@dekfracsimple 3060 \def\gmu@dekfracsimple#1/#2{\leavevmode\kern.1em}
```

```

3061  \raise.5ex\hbox{%
3062    \smaller[3]#1}\gmu@numeratorkern
3063  \dekfraccslash\gmu@denominatorkern
3065  {%
3066    \smaller[3]#2}%
3067  \if@gmu@mmhbox\egroup\fi}

\dekfracsimple
3070 \def\dekfracsimple{%
3071   \let\dekfracc@args\gmu@dekfracsimple
3072 }

\dekfraccslash
3073 \@ifXeTeX{\def\dekfraccslash{\char"2044}}{%
3074   \def\dekfraccslash{/}}% You can define it as the fraction
3075   slash, \char"2044
3076 \dekfracsimple

```

A macro that acts like `\,` (thin and unbreakable space) except it allows hyphenation afterwards:

```
\ikern 3084 \newcommand*\ikern{\,,\penalty1000\hskip0pt\relax}
```

And a macro to forbid hyphenation of the next word:

```
\nohy 3088 \newcommand*\nohy{\leavevmode\kern0pt\relax}
```

```
\yeshy 3089 \newcommand*\yeshy{\leavevmode\penalty1000\hskip0pt\relax}
```

In both of the above definitions ‘osp’ not `\z@` to allow their writing to and reading from files where @ is ‘other’.

```
\@isempty
```

```
\@isempty 3095 \long\pdef\@isempty#1#2#3{%
3096   \def\gmu@reserveda{#1}%
3097   \ifx\gmu@reserveda\@empty\afterfi{#2}%
3098   \else\afterfi{#3}\fi
3099 }
```

\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
3111 \def\gmu@gettext#1.#2\@nil{%
3112   \def\gmu@filename{#1}%
3113   \def\gmu@fileext{#2}%
3115 \def\include#1{\relax
3116   \ifnum\@auxout=\@partaux
3117     \@latex@error{\string\include\space cannot be nested}\@eha
3118   \else\@include#1\fi}
3120 \def\@include#1{%
3121   \gmu@gettext#1.\@nil
3123   \ifx\gmu@fileext\empty\def\gmu@fileext{tex}\fi
3124   \clearpage
3125   \if@files
3126     \immediate\write\@mainaux{\string\@input{\gmu@filename.aux}}%
3127   \fi

```

```

3128  \@tempsw@true
3129  \if@partsw
3130    \@tempsw@false
3131    \edef\reserved@b{\#1}%
3132    \@for\reserved@a:=\@partlist\do{%
3133      \ifx\reserved@a\reserved@b\@tempsw@true\fi}%
3134    \fi
3135  \if@tempswa
3136    \let\@auxout\@partaux
3137    \if@filesw
3138      \immediate\openout\@partaux\gmu@filename.aux
3139      \immediate\write\@partaux{\relax}%
3140    \fi
3141    \@input{\gmu@filename.\gmu@fileext}%
3142    \inlasthook
3143    \clearpage
3144    \@writeckpt{\gmu@filename}%
3145    \if@filesw
3146      \immediate\closeout\@partaux
3147    \fi
3148  \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.

```

3152  \deadcycles\z@
3153  \nameuse{cp@\gmu@filename}%
3154  \fi
3155  \let\@auxout\@mainaux}

\whenonly 3158 \newcommand\whenonly[3]{%
\gmu@whonly 3159  \def\gmu@whonly{\#1,}%
3160  \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 3164 \def\inlasthook{\thispagestyle{closing}}
```

Faked small caps

```

\gmu@scapLetters 3170 \def\gmu@scapLetters#1{%
3171  \ifx#1\relax\relax\else% two \relaxes to cover the case of empty #1.
3172  \ifcat_a#1\relax
3173    \ifnum\the\lccode`#1=\#1\relax
3174      {\fakescapscore\MakeUppercase{\#1}}% not Plain \uppercase because
3175      that works bad with inputenc.
3176    \else#1%
3177    \fi
3178  \else#1%
3179  \fi%
3180  \gmu@scapLetters
3181  \fi}%
\gmu@scapSpaces 3183 \def\gmu@scapSpaces#1\#2\@nil{%
3184  \ifx#1\relax\relax
3185  \else\gmu@scapLetters#1\relax
3186  \fi

```

```

3187  \ifx#2\relax\relax
3188  \else\afterfi{\gmu@scapSpaces#2\@nil}%
3189  \fi}
\gmu@scapss#1\@nil{{\def~{{\nobreakspace}}}}%
3191 \def\gmu@scapss#1\@nil{{\def~{{\nobreakspace}}}}%
3192      \gmu@scapSpaces#1\@nil}%% \def\\{{\newline}}\relax adding re-
                           definition of \\ caused stack overflow. Note it disallows hyphenation
                           except at \-
\nobreakspace
\fakescaps 3196 \pdef\fakescaps#1{{\gmu@scapss#1\@nil}}
3198 \let\fakescapscore\gmu@scalematchX
Experimente z akcentami patrz no3.tex.
\tinycae 3201 \def\tinycae{{\tiny AE}}% to use in \fakescaps[\tiny]{...}
3203 \RequirePackage{calc}
      wg \zf@calc@scale pakietu fontspec.

\gmu@scalar 3207 \@ifpackageloaded{fontspec}{%
3208   \def\gmu@scalar{1.0}%
3209   \def\zf@scale{}%
\gmu@scalematchX 3210 \def\gmu@scalematchX{%
3211   \begingroup
3212     \ifx\zf@scale\empty\def\gmu@scalar{1.0}%
3213     \else\let\gmu@scalar\zf@scale\fi
3214     \setlength{\tempdima}{\fontdimen5\font}5—ex height
3215     \setlength{\tempdimb}{\fontdimen8\font}8—XTEX synthesized up-
                           percase height.
3216     \divide{\tempdimb}{1000}\relax
3217     \divide{\tempdima}{\tempdimb}
3218     \setlength{\tempdima}{\tempdima*\real{\gmu@scalar}}%
3219     \gm@ifundefined{fakesc@extrascale}{}{%
3220       \setlength{\tempdima}{\tempdima*\real{%
3221         \fakesc@extrascale}}%
3222       \tempcpta=\tempdima
3223       \divide{\tempcpta}{1000}\relax
3224       \tempcntb=-1000\relax
3225       \multiply{\tempcntb}{\tempcpta}
3226       \advance{\tempcntb}{\tempdima}
3227       \xdef\gmu@scscale{\the\tempcpta.%
3228         \ifnum\tempcntb<100\o\fi
3229         \ifnum\tempcntb<10\o\fi
3230         \the\tempcntb}%
3231       \endgroup
3232       \addfontfeature{Scale=\gmu@scscale}%
3233     }{\let\gmu@scalematchX\smallerr}
3234   }{\def\fakescextrascale#1{\def\fakesc@extrascale{#1}}}
\fakescextrascale
\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 3242 \newcommand*\wyzejnizej[1]{%
3243   \edef\gmu@tempa{\gm@ifundefined{r@#1}{\arabic{page}}{%
3244     \xa\x@\xa\@secondoftwo\csname_r@#1\endcsname}}%

```

```

3245  \ifnum\gmu@tempa<\arabic{page}\relax_wy\.zej\fi
3246  \ifnum\gmu@tempa>\arabic{page}\relax_ni\.zej\fi
3247  \ifnum\gmu@tempa=\arabic{page}\relax_\@xa\ignorespaces\fi
3248 }

```

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 3258 \def\napapierkistretch{o,3pt}%
3259 It's quite much for 11/13pt leading.
\napapierkicore   3260 \def\napapierkicore{\advance\baselineskip%
3261   by_\optplus\napapierkistretch\relax}
napapierki    3263 \newenvironment*{napapierki}{%
3264   \par\global\napapierkicore}{%
3265   \par\dimen\z@=\baselineskip
3266   \global\baselineskip=\dimen\z@}%
so that you can use \endnapapierki in
interlacing environments.

```

```

\gmu@luzniej 3270 \newcount\gmu@luzniej
\luzniejcore 3272 \newcommand*\luzniejcore[1][1]{%
3273   \advance\gmu@luzniej\@ne}%
We use this count to check whether we open the
environment or just set \looseness inside it again.
3275 \ifnum\gmu@luzniej=\@ne\multiply\tolerance_by_2\fi
3276 \looseness=#1\relax}

```

After \begin{luzniej} we may put the optional argument of \luzniejcore

```

luzniej 3280 \newenvironment*{luzniej}{\par\luzniejcore}{\par}

```

The starred version does that \everypar, which has its advantages and disadvantages.

```

luzniej* 3285 \newenvironment*{luzniej*}[1][1]{%
3286   \multiply\tolerance_by_2\relax
3287   \everypar{\looseness=#1\relax}}{\par}
\nawj 3289 \newcommand*\nawj{\kern0.1em\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). It begins with \ifhmode to be usable also at the beginning of a line as the mark of a dialogue.

```

3297 \ifdefined\XeTeXversion
\npauza@skipcore 3298 \def\npauza@skipcore{\hskip.2em\plus.1em\relax
3299   \pauzacore\hskip.2em\plus.1em\relax\ignorespaces}%
\nppauza@skipcore 3301 \def\nppauza@skipcore{\unskip\penalty10000\hskip.2em\plus.1em\%
3302   \relax
3303   -\hskip.2em\plus.1em\ignorespaces}
3304 \AtBeginDocument{%
3305   \pdef\-\{}%
3306   \ifhmode
3307     \unskip\penalty10000
3308     \afterfi{%
3309       \@ifnextspace{\pauza@skipcore}%
3310       {\@ifnextMac{\pauza@skipcore{%
3311         \pauzacore\penalty\hyphenpenalty\hskip\z@}}}}%
3312   \else

```

According to *Instrukcja technologiczna. Skład ręczny i maszynowy* the dialogue dash should be followed by a rigid hskip of $\frac{1}{2}$ em.

```
3316      \leavevmode\pauzacore\penalty10000\hskip0,5em\ignorespaces
3317      \fi}%

```

The next command's name consists of letters and therefore it eats any spaces following it, so \@ifnextspace would always be false.

```
\pauza 3320  \pdef\pauza{%
3321    \ifhmode
3322      \unskip\penalty10000
3323      \hskip0.2em\plus0.1em\relax
3324      \pauzacore\hskip.2em\plus0.1em\relax\ignorespaces%
3325    \else
3326      \pauzadial
3327      \fi}%

```

According to *Instrukcja technologiczna. Skład ręczny i maszynowy* the dialogue dash should be followed by a rigid hskip of $\frac{1}{2}$ em.

```
\pauzadial 3332  \pdef\pauzadial{%
3333    \leavevmode\pauzacore\penalty10000\hskip0,5em\ignorespaces}

```

And a version with no space at the left, to begin a \noindent paragraph or a dialogue in quotation marks:

```
\lpauza 3337  \pdef\lpauza{%
3338    \pauzacore\hskip.2em\plus0.1em\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 space₁₀. 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...

```
\- 3346  \pdef\-\{%
3347    \ifvmode\PackageError{gmutils}{%
3348      command\bslash\ppauza(en\_dash)not\_intended\_for\_vmode.}{%
3349      Use\bslash\ppauza(en\_dash)only\_in\_number\_and\_numeral\_ranges.}%
3350  \else
3351    \afterfi{%
3352      \ifnextspace{\ppauza@skipcore}{%
3353        \ifnextMac\ppauza@skipcore{\unskip\discretionary{-}{-}{-}}{%
3354          }%
3355        \fi
3356      }%
3357    \pdef\ppauza{%
3358      \ifvmode\PackageError{gmutils}{%
3359        command\bslash\ppauza(en\_dash)not\_intended\_for\_vmode.}{%
3360        Use\bslash\ppauza(en\_dash)only\_in\_number\_and\_numeral\_ranges.}%
3361      \else
3362        \unskip\discretionary{-}{-}{-}%
3363        \fi}%
3364    \def\emdash{\char`\-}%
3365  }%
3366 }% of at begin document

```

```

\longpauza 3369 \def\longpauza{\def\pauzacore{-}}
\pauzacore 3370 \longpauza
\shortpauza 3371 \def\shortpauza{%
\pauzacore 3372   \def\pauzacore{-\kern,23em\relax\llap{-}}}
3373 \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 `-`.

```

3379 \ifdefined\XeTeXversion
3380 \foone{\catcode`-\active}\catcode`-\active\catcode`-\active}{%
\adashes 3381   \def\adashes{\AtBeginDocument\adashes}%
because \pauza is defined at
begin document.
\adashes 3383   \AtBeginDocument{\def\adashes{%
3384     \catcode`-\active\let-\-%
3385     \catcode`-\active\let-\-%
3387 }}}
3388 \else
3389 \relaxen\adashes
3390 \fi

```

The hyphen shouldn't be active IMO because it's used in T_EX 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 3399 \def\gmu@dywiz{\ifmmode-\else
3400   \ifvmode-\else\afterfifi\dywiz\fi\fi}%
3402 \foone{\catcode`-\active}{%
\ahyphen 3403   \def\ahyphen{\let-\gmu@dywiz\catcode`-\active}}

```

To get current time. Works in ε-T_EXs, including X_ET_EX. `\czas` typesets 19.00 and `\czas[:]` typesets 19:00.

```

\czas 3408 \newcommand*\czas[1][.]{%
3409   \the\numexpr(\time-30)/60\relax#1%
3410   \tempcnta=\numexpr\time-(\time-30)/60*60\relax
3411   \ifnum\tempcnta<10\fi\the\tempcnta}

3414 \@ifXeTeX{%
\textbullet 3415   \pdef\textbullet{%
3418     \iffontchar\font"2022\char"2022\else\ensuremath{\bullet}%
     \fi}}
\textbullet 3419 {\def\textbullet{\ensuremath{\bullet}}}

\tytulowa 3421 \newenvironment*{\tytulowa}{\newpage}{\par\thispagestyle{empty}%
\newpage}

```

To typeset peoples' names on page 4 (the editorial page):

```

\nazwired 3424 \def\nazwired{\quad\textsc}

```

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 \thedata and the same time to be able to turn \datef off in some cases (for SevSevo4.tex).

```

\polskadata 3438 \newcommand*\polskadata{%
\gmu@datef 3439   \def\gmu@datef##1-##2-##3##4,##5\gmu@datef{%
3440     \ifx\relax##2\relax##3##4%
3441     \else
3442       \ifnum##3<@firstofmany##4o\@nil=o\relax
3443       \else
3444         \ifnumo##3=o\relax
3445         \else##3%
3446         \fi##4%
3447       \fi
3448       \ifcase##2\relax\or\_\_stycznia\or\_\_lutego%
3449       \or\_\_marca\or\_\_kwietnia\or\_\_maja\or\_\_czerwca\or\_\_lipca\or\_\_
3450         \_sierpnia%
3451       \or\_\_wrze\u\145nia\or\_\_pa\u\145dziernika\or\_\_listopada\or\_\_grudnia\else
3452     \{}%
3453     \fi
3454     \if\relax##1\relax\else\_\fi\#1%
3455     \fi
3456     \gmu@datecomma{\#5}\}%
3457   }% of \gmu@datef.

\gmu@datefsl 3457 \def\gmu@datefsl##1##2##3##4,##5\gmu@datefsl{%
3458   \if\relax##2\relax##3##4%
3459   \else
3460     \ifnum##3<@firstofmany##4o\@nil=o\relax
3461     \else
3462       \ifnumo##3=o\relax
3463       \else##3%
3464       \fi##4%
3465     \fi
3466     \ifcase##2\relax\or\_\_stycznia\or\_\_lutego%
3467     \or\_\_marca\or\_\_kwietnia\or\_\_maja\or\_\_czerwca\or\_\_lipca\or\_\_
3468       \_sierpnia%
3469     \or\_\_wrze\u\145nia\or\_\_pa\u\145dziernika\or\_\_listopada\or\_\_grudnia\else
3470   \{}%
3471   \fi
3472   \if\relax##1\relax\else\_\fi\#1%
3473   \fi
3474 }% of \polskadata
3475 \polskadata

```

For documentation in English:

```

\englishdate 3482 \newcommand*\englishdate{%
\gmu@datef 3483   \def\gmu@datef##1-##2-##3##4,##5\gmu@datef{%
3484     \if\relax##2\relax##3##4%
3485     \else
3486       \ifcase##2\relax\or\_\_January\or\_\_February%
3487       \or\_\_March\or\_\_April\or\_\_May\or\_\_June\or\_\_July\or\_\_August%
3488       \or\_\_September\or\_\_October\or\_\_November\or\_\_December\else
3489     \{}%
3490     \fi

```

```

3491 \ifnum##3@firstofmany##4o@@nil=o\relax
3492   \else
3493     \%
3494     \ifnumo##3=o\relax
3495       \else##3%
3496         \fi##4%
3497         \ifcase##3@firstofmany##4\relax@@nil\relax\or_st\or_nd%
3498           \or_rd\else_th\fi
3499         \fi
3500         \ifx\relax##1\relax\else,\_\fi##1%
3501       \fi
3502     \gmu@datecomma{##5}%
3503     \def\gmu@datefs1##1##2##3##4##5\gmu@datefs1{%
3504       \if\relax##2\relax##3##4%
3505       \else
3506         \ifcase##2\relax\or_January\or_February%
3507           \or_March\or_April\or_May\or_June\or_July\or_August%
3508           \or_September\or_October\or_November\or_December\else
3509             {}%
3510           \fi
3511         \ifnum##3@firstofmany##4o@@nil=o\relax
3512           \else
3513             \%
3514             \ifnumo##3=o\relax
3515               \else##3%
3516                 \fi##4%
3517                 \ifcase##3@firstofmany##4\relax@@nil\relax\or_st\or_nd%
3518                   \or_rd\else_th\fi
3519                   \fi
3520                   \if\relax##1\relax\else,\_\fi##1%
3521                 \fi
3522               \gmu@datecomma{##5}%
3523             }
3524
3525 \def\gmu@datecomma#1{%
3526   sometimes we want to typeset something like '11 wrześ-
3527   nia, czwartek' so we add handling for comma in the \ldate's argument.
3528   \ifx\gmu@datecomma#1\gmu@datecomma\else
3529     ,\gmu@stripcomma#1%
3530   \fi
3531 }% of \gmu@datecomma
3532
3533 \def\gmu@stripcomma#1,{#1}
3534 \ifgmu@dash
3535   \newif\ifgmu@dash
3536 \ifgmu@dash
3537   \def\gmu@ifnodash#1-#2@@nil{%
3538     \def\gmu@tempa{#2}%
3539     \ifx\gmu@tempa\@empty}
3540
3541 \gmu@testdash
3542   \pdef\gmu@testdash#1\ifgmu@dash{%
3543     \gmu@ifnodash#1-\@nil
3544       \gmu@dashfalse
3545     \else
3546       \gmu@dashtrue
3547   \fi
3548   \ifgmu@dash}

```

A word of explanation to the pair of macros above. `\gmu@testdash` sets `\iftrue` the `\ifgmu@dash` switch if the argument contains an explicit `-`. To learn it, an auxiliary `\gmu@ifdash` macro is used that expands to an open (un\fied) `\ifx` that tests whether the dash put by us is the only one in the argument string. This is done by matching the parameter string that contains a dash: if the investigated sequence contains (another) dash, #2 of `\gmu@ifdash` becomes the rest of it and the ‘guardian’ dash put by us so then it’s nonempty. Then #2 is took as the definiens of `\@tempa` so if it was empty, `\@tempa` becomesx equal `\@empty`, otherwise it isx not.

Why don’t we use just `\gmu@ifdash`? Because we want to put this test into another `\if...`. A macro that doesn’t *mean* `\if...` wouldn’t match its `\else` nor its `\fi` while TeX would skip the falsified branch of the external `\if...` and that would result in the ‘extra `\else`’ or ‘extra `\fi`’ error.

Therefore we wrap the very test in a macro that according to its result sets an explicit Boolean switch and write this switch right after the testing macro. (Delimiting `\gmu@testdash`’es parameter with this switch is intended to bind the two which are not one because of TeXnical reasons only.)

Warning: this pair of macros may result in ‘extra `\else`/extra `\fi`’ errors however, if `\gmu@testdash` was `\expandafter`d.

Dates for memoirs to be able to typeset them also as diaries.

```

\ifdate 3579 \newif\ifdate
\bidate 3581 \pdef\bidate#1{%
 3582   \ifdate\gmu@testdash#1{%
 3583     \ifgmu@dash
 3584       \gmu@datef#1,\gmu@datef
 3585     \else
 3586       \gmu@datefsl#1,\gmu@datefsl
 3587     \fi\fi}
\linedate 3589 \pdef\linedate{\@ifstar\linedate@\@linedate@}
\linedate@ 3590 \pdef\linedate@#1{\linedate@{--{}{}#1}}
\linedate@ 3591 \pdef\linedate@#1{\par\ifdate\addvspace{\dateskip}{%
 3592   \date@line{\footnotesize\itshape\bidate{#1}}%
 3593   \nopagebreak\else%\ifnum\arabic{dateinsection}>0\dekbigskip\fi
 3594   \addvspace{\bigskipamount}%
 3595   \fi}% end of \linedate.
 3597 \let\dateskip\medskipamount
\rdate 3605 \pdef\rdate{\let\date@line\rightline\linedate}
\ldate 3606 \pdef\ldate{%
\date@line 3608   \def\date@line##1{\par\raggedright##1\par}%
 3609   \linedate}
\runindate 3610 \newcommand*\runindate[1]{%
 3611   \paragraph{\footnotesize\itshape\gmu@datef#1\gmu@datef}%
 3612   \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.

```

 3615 \let\thedata\ldate
\zwrobcy 3618 \pdef\zwrobcy#1{\emph{#1}}% ostinato, allegro con moto, garden party etc.,
          także kompliment
\tytul 3621 \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 3627 \newenvironment{maszynopis}[1][]{{\#1\ttfamily
3628   \hyphenchar\font=45\relax% this assignment is global for the font.
3629   \tempskipa=\glueexpr\rightskip+\leftskip\relax
3630   \ifdim\gluestretch\tempskipa=\z@%
3631   \tolerance900
      it worked well with tolerance = 900.
3633   \advance\rightskip by \z@ pluso,5em\relax\fi
3634   \fontdimen3\font=\z@% we forbid stretching spaces...
% \fontdimen4\font=\z@ but allow shrinking them.
3636   \hyphenpenaltyo% not to make TeX nervous: in a typewriting this marvellous
      algorithm of hyphenation should be turned off and every line broken at the
      last allowable point.
3639   \StoreMacro\pauzacore
\pauzacore 3640   \def\pauzacore{-\rlap{\kern-o,3em-}}%
3641 }{\par}

\justified 3645 \newcommand*\justified{%
3646   \leftskip=1\leftskip% to preserve the natural length and discard stretch and
      shrink.
3648   \rightskip=1\rightskip
3649   \parfillskip=1\parfillskip
3650   \advance\parfillskip by \osp plus \ifil\relax
3651   \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 3656 \newcommand*\fullpar{%
3657   \hunskip
3658   \bgroup\parfillskip\z@skip\par\egroup}

To conform Polish recommendation for typesetting saying that the last line of a para-
graph 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 interword spaces so we provide a declaration to catch the proper glue where the
parindent is set (e.g. in footnotes parindent is 0pt)
\twoparinit 3667 \newcommand*\twoparinit{%
      the name stands for 'last paragraph line's length
      minimum two \parindent.
3669   \edef\twopar{%
3670     \hunskip% it's \protected, remember?
3671     \bgroup
3672     \parfillskip=\the\glueexpr
3673     \dimexpr\textwidth-2\parindent\relax
3674     minus\dimexpr\textwidth-2\parindent\relax
3675     \relax% to delimit \glueexpr.
3676     \relax% to delimit the assignment.
3677     \par\egroup
3678   }% of \gmu@twoparfill
3683 }% of \twoparinit.

For dati under poems.

\wherncore 3690 \newcommand\wherncore[1]{%
3691   \rightline{%
3692     \parbox{o,7666\textwidth}{%
3693       \leftskip\osp plus \textwidth
3694       \parfillskip\relax
}

```

```

3695      \let\\linebreak
3696      \footnotesize#1}}
\whern 3698 \def\whern{%
3699   \@ifstar\wherncore{\vskip\whernskip\wherncore}{}
\whernskip 3702 \newskip\whernskip
3703 \whernskip2\baselineskip minus 2\baselineskip\relax
\whernup 3705 \newcommand\whernup[1]{\par\wherncore{#1}}

```

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 3729 \def\leftslanting{%
\litshape 3730  \pdef\litshape{%
3732   \itshape
3733   \tempdima=-2\fontdimen1\font
3734   \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.
3738   \edef\gmu@tempa{%
3739     @nx\addfontfeature{RawFeature={slant=\strip@pt%
           \tempdima}}% when not \edefed, it caused an error, which is
           perfectly understandable.
           \gmu@tempa}%
3742   \pdef\textlit##1{%
3745     \pdef\textlit##1{%
3746       {\litshape##1}}%
3748   \pdef\lit{\rm\litshape}%
3751   \pdef\lsl{{\it
3754     \tempdima=-\fontdimen1\font
3755     \xdef\gmu@tempa{%
3756       @nx\addfontfeature{RawFeature={slant=\strip@pt%
           \tempdima}}}}%
3757   \rm% Note in this declaration we left-slant the basic Roman font not the it-
           shape of the current family.
           \gmu@tempa}%
3759

```

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 3767 \pdef\em{%
3768   \ifdim\fontdimen1\font=\z@\itshape
3769   \else
3770     \ifdim\fontdimen1\font>\z@\rm\itshape

```

```

3771      \else\upshape
3772      \fi
3773      \fi}%
3776  \pdef\emph##1{%
3777  {\em##1}}%
3778 }% of \leftslanting.

```

Thousand separator

\thousep 3782 \pdef\thousep#1{%
 a macro that'll put the thousand separator between every two
 three-digit groups.

First we check whether we have at least five digits.

```

3786  \gmu@thou@fiver#1\relax\relax\relax\relax% we put
      five \relaxes after the parameter to ensure the string will
      meet \gmu@thou@fiver's definition.
3789  \gmu@thou@fiver{#1}{%
  if more than five digits:
  \emptyify\gmu@thou@put
  \relaxen\gmu@thou@o\relaxen\gmu@thou@i\relaxen\gmu@thou@ii
  \tempcnta\z@
  \gmu@thou@putter#1\gmu@thou@putter
  \gmu@thou@put
}
3795

```

\gmu@thou@fiver 3797 \def\gmu@thou@fiver#1#2#3#4#5\gmu@thou@fiver#6#7{%
 3798 \ifx\relax#5\relax\afterfi{#6}\else\afterfi{#7}\fi}

\gmu@thou@putter 3800 \def\gmu@thou@putter#1#2{%
 we are sure to have at least five tokens before the
 guardian \gmu@thou@putter.
 3802 \advance\tempcnta\@ne
 3803 \tempcntb\tempcnta
 3804 \divide\tempcntb\@relax
 3805 \tempcnta=\numexpr\tempcnta-\tempcntb*3
 3806 \edef\gmu@thou@put{\gmu@thou@put#1%
 3807 \ifx\gmu@thou@putter#2\else
 3808 \ifcase\tempcnta
 3809 \gmu@thou@o\or\gmu@thou@i\or\gmu@thou@ii%
 all three cases are
 yet \relax so we may put them in an \edef safely.
 3810 \fi
 3811 \fi}%
 3812 \fi
 3813 \fi}%
 3814 \ifx\gmu@thou@putter#2%
 if we are at end of the digits...
 3815 \edef\tempa{%
 3816 \ifcase\tempcnta
 3817 \gmu@thou@o\or\gmu@thou@i\or\gmu@thou@ii
 3818 \fi}%
 3819 \xa\let\tempa\gmu@thousep% ... we set the proper cs...
 3820 \else%
 or ...
 3821 \afterfi{%
 iterate.
 3822 \gmu@thou@putter#2}%
 3823 \fi%
 of if end of digits.
 3824 }%
 of \gmu@thou@putter.

\gmu@thousep 3826 \def\gmu@thousep{\,}%
 in Polish the recommended thousand separator is a thin
 space.

So you can type `\thousep{7123123123123}` to get `7 123 123 123 123`. 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 3834 \pdef\xathousep#1{\@xa\thousep\@xa{\the#1}}
            Now write \xathousep{\numexpr\z@*9*8*7*6*\z@} to get 3 628 800.

\shortthousep 3838 \def\shortthousep{%
    \thous 3839 \pdef\thous{%
        3840     \ifmmode\hbox\bgroup\gmu@mmhboxtrue\fi
        3841     \afterassignment\thous@inner
        3842     \tempcnta=\z@%
    }
    \thous@inner 3844 \def\thous@inner{%
        3845         \ifnum\tempcnta<\z@-$-
        3846             \tempcnta=-\tempcnta
        3847             \fi
        3848             \xathousep\tempcnta
        3849             \if@gmu@mmhbox\egroup
        3850             \else\afterfi{\ifnextcat\z@a\space{}\z@}%
        3851             \fi}%
    }% of \shortthousep.
```

And now write `\thous\z@3628800` to get `3 628 800` even with a blank space (beware of the range of TeX's counts).

Storing and restoring the catcodes of specials

```
\gmu@storespecials 3859 \newcommand*\gmu@storespecials[1][]{%
    we provide a possibility of adding
    stuff. For usage see line 2615.
    3861 \def\do##1{\catcode`@nx##1=\the\catcode`##1\relax}%
    3862 \edef\gmu@restorespecials{\dospecials\do\^\^M#1}

\gmu@septify 3864 \newcommand*\gmu@septify[1][]{%
    restoring the standard catcodes of specials.
    The name is the opposite of 'sanitize' :-). It restores also the original catcode
    of \^\^M
    3867 \def\do{\relax\catcode`}%
    3868 \do\z@\do\\o\do\{1\do\}2\do\$3\do\&4%
    3869 \do\#6\do\^7\do\_\_8\do\%14\do\~13\do\^\^M5#1\relax}
```

hyperref's \nolinkurl into \url*

```
\urladdstar 3873 \def\urladdstar{%
    3874     \AtBeginDocument{%
        3875         \ifpackage{hyperref}{%
            3876             \StoreMacro\url
        \url 3877             \def\url{\ifstar{\nolinkurl}{\storedcsname{url}}}}%
        3878     }{}}
    3880 \onlypreamble\urladdstar
    3883 \endinput
```

d. The gmiflink Package¹

Written by Grzegorz ‘Natror’ Murzynowski,
natror at o2 dot pl

© 2005, 2006 by Grzegorz ‘Natror’ 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”.

```
44 \NeedsTeXFormat{LaTeX2e}
45 \ProvidesPackage{gmiflink}
46 [2006/08/16 vo.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 \gmhypertarget for conditional making of hyperlinks in your document.

\gmhypertarget[⟨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.

gmiflink.sty
README

¹ This file has version number vo.97 dated 2006/08/16.

gmiflink.pdf
gmiflink.tds.zip

The code

```

144 \@ifpackageloaded{hyperref}{}{\message{^^J^^J gmiflink package:
145 There's no use of me without hyperref package, I end my
      input.^^J}\endinput}
147 \providecommand{\empty}{}
      A new counter, just in case
GMlabel 149 \newcounter{GMlabel}
150 \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 160 \DeclareRobustCommand*\gmhypertarget{%
161 \ifnextchar[]{\gmhypertarget}{\@dblarg{\gmhypertarget}}}
\gmhypertarget 164 \def\gmhypertarget[#1]{% If argument #1 = \empty, then we'll use #2, i.e.,
      the same as name of hypertarget.
167 \refstepcounter{GMlabel}% we \label{\gmht@firstpar}
169 \hypertarget{#1}{#2}%
170 \protected@write\auxout{}{%
171 \string\newlabel{#1}{#2}\the\page\relax{GMlabel.%}
      \arabic{GMlabel}}{}%
172 }% end of \gmhypertarget.

```

We define a macro such that if the target exists, it makes `\ref`, else it typesets ordinary text.

```

\gmiref 177 \DeclareRobustCommand*\gmiref{\ifnextchar[]{\gmiref}{%
178 \@dblarg{\gmiref}}}
\gmiref 180 \def\gmiref[#1]{%
181 \expandafter\ifx\csname r@#1\endcsname\relax\relax%
182 #2\else\ref{#1}\fi%
183 }% end of \gmiref
\gmiflink 186 \DeclareRobustCommand*\gmiflink{\ifnextchar[]{\gmiflink}{%
187 \@dblarg{\gmiflink}}}
\gmiflink 189 \def\gmiflink[#1]{%
190 \expandafter\ifx\csname r@#1\endcsname\relax\relax%
191 #2\else\hyperlink{#1}{#2}\fi%
192 }% end of \gmiflink

```

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`.

```

198 \endinput
      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 Package¹

October 6, 2008

This is (a documentation of) file gmverb.sty, intended to be used with L^AT_EX 2_E 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 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>
for the details of that license.

LPPL status: "author-maintained".

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

```
71 \NeedsTeXFormat{LaTeX2e}
72 \ProvidesPackage{gmverb}
73 [2008/08/30 vo.89 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 his 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\verb|\example| instead of \verb|\a\verb|\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.

\fixbslash
\VerbHyphen
\fixbrace (If you don't like linebreaking at backslash, there's the \fixbslash declaration (observing the common scoping rules, hence OCSR) and an analogous declaration for the left brace: \fixbrace.)

\VerbbeolOK
\VerbHyphen
\fixbrace (The default 'hyphen' is % since it's the default comment char. If you wish another char to appear at the linebreak, use the \VerbHyphen declaration that takes \<char> as the only argument. This declaration is always global.)

\VerbbeolOK
\VerbHyphen
\fixbrace (Another difference is the \VerbbeolOK declaration (OCSR). Within its scope, \verb allows an end of a line in its argument and typesets it just as a space.)

¹ This file has version number vo.89 dated 2008/08/30.

- As in the standard version(s), the plain `\verb` typesets the spaces blank and `\verb*` makes them visible.
- `\MakeShortVerb` Moreover, gmverb provides the `\MakeShortVerb` 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 `\MakeShortVerb*`|` (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 `|%` `\mymacro` to get `\mymacro` instead of typing `\verb+|`|\mymacro+`. Because the char used in this example is my favourite and used just this way by DEK in the *The TeXbook*'s format, gmverb provides a macro `\dekclubs` as a shorthand for `\MakeShortVerb(*)%`|`.
- `\DeleteShortVerb` Be careful because such active chars may interfere with other things, e.g., the `|` with the vertical marker in tables 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.
- `\OldMakeShortVerb` One more difference between gmverb and shortverb is that the chars `\activeated` by `\MakeShortVerb` in the math mode behave as if they were 'other', so you may type e.g., `$|$`|`+` `\activeated` this way is in the math mode typeset properly etc.
- However, if you don't like such a conditional behaviour, you may use `\OldMakeShortVerb` instead, what I do when I like to display short verbatims in displaymath.
- `\dekclubs` There's one more declaration provided by gmverb: `\dekclubs`, which is a shorthand for `\MakeShortVerb`|`, `\dekclubs*` for `\MakeShortVerb*`|` and `\olddekclubs` for `\OldMakeShortVerb`|`.
- `\edverbs` There's one more declaration, `\edverbs` that makes `\[` checks if the next token is an active char and opens an `\hbox` if so. That is done so that you can write (in `\edverbs'` and `\dekclubs'` scope)
- ```
\[|<verbatim stuff>|\]
```
- instead of
- ```
\[\hbox{|<verbatim stuff>|}\]
```
- to get a displayed shortverb.
- `\VisSpacesGrey` 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.
- 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

```
249 \RequirePackage{gmutils}[2008/08/06]
```

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 261 \let\verbhyphen\xiipercent
```

Provide a declaration for easy changing it. Its argument should be of `\<char>` form (of course, a `\<char>` is also allowed).

```
\VerbHyphen 267 \def\VerbHyphen#1{%
268   {\escapechar\m@ne
269    \@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 278 \def\breakbrace{%
279   \discretionary{\xilbrace\verbhyphen}{}{\xilbrace}
280   \foone{\catcode`\[=1\catcode`\{=\active\catcode`\]=2\}%
281   [%
282     \def\dobreakbrace[\catcode`\{=\active
283     \def{%
284       [\breakbrace\gm@lbracehook]%
285     ]
286   ]
```

Now we only initialize the hook. Real use of it will be made in gmdoc.

```
287 \relaxen\gm@lbracehook
```

The `\bslash` macro defined below I use also in more 'normal' TeXing, e.g., to `\typeout` some `\outer` macro's name.

```
296 \foone{\catcode`\!=o\makeother\}\}%
297 {%
\bslash 298 !def!bslash{\}%
299 !def!breakbslash{!discretionary{!verbhyphen}{\}{\}}%
300 }
```

Sometimes linebreaking at a backslash may be unwelcome. The basic case, when the first cs in a `verbatim` breaks at the lineend leaving there `%`, is covered by line 614. For the others let's give the user a countercrank:

```
\fixbslash 307 \newcommand*\fixbslash{\let\breakbslash=\bslash}%
to use due to the com-
mon scoping rules. But for the special case of a backslash opening a verbatim
scope, we deal specially in the line 614.
```

Analogously, let's provide a possibility of 'fixing' the left brace:

```
\fixlbrace 313 \newcommand*\fixlbrace{\let\breakbrace=\xilbrace}%
316 \foone{\catcode`\!=o\catcode`\!=\active}\}
```

```

318  {%
\def !dobreakbslash{!catcode`!`!=!active!def\{!breakbslash}}%
\breakbslash
320 }

The macros defined below, \visiblebreakspaces and \xiiclub we'll use in the
almost Knuthian macro making verbatim. This 'almost' makes a difference.

326 \foone{\catcode`\_=12}% note this space is _10 and is gobbled by parsing the
number. \visiblespace is \let in gmutils to \xiispace or \xxt@visiblespace
of \ltextra if available.

\breakablevisspace 330 \def\breakablevisspace{\discretionary{\visiblespace}{}{%
\visiblespace}}
333 \foone\obeyspaces% it's just re\catcode'ing.
334 {%
335 \newcommand*\activespace{_}%
336 \newcommand*\dobreakvisiblespace{\def\breakablevisspace\obeyspaces}{%
\defining it caused a stack overflow disaster with gmdoc.
338 \newcommand*\dobreakblankspace{\let\=space\obeyspaces}{%
339 }
342 \bgroup\@makeother\|%
343 \firstofone{\egroup\def\xiiclub{|}}}

```

Almost-Knuthian \ttverbatim

\ttverbatim comes from *The TeXbook* too, but I add into it a L^AT_EX macro changing the \catcodes and make spaces visible and breakable and left braces too.

```

\ttverbatim 352 \newcommand*\ttverbatim{%
353   \let\do=\do@noligs\verbatim@nolig@list
354   \let\do=\@makeother\dospecials
355   \dobreaklbrace\dobreakbslash
356   \dobreakspace
357   \tt
358   \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.

```

365 \emptyify\ttverbatim@hook
368 \def\VerbT1{\def\ttverbatim@hook{\fontencoding{T1}\selectfont}}
\VerbT
\VerbT
\ttverbatim@hook 372 \let\dobreakspace=\dobreakvisiblespace

```

The core: from shortverb

The below is copied verbatim ;-) from doc.pdf and then is added my slight changes.

```

\MakeShortVerb 381 \def\MakeShortVerb{%
382   \@ifstar
383   {\def\@shortverbdef{\verb*}\@MakeShortVerb}%
384   {\def\@shortverbdef{\verb}\@MakeShortVerb}%
387 \def\@MakeShortVerb#1{%
388   \@xa\ifx\csname\string#1\endcsname\relax

```

```

389  \">@shortvrbinfo{Made_\#1}\@shortvrbdef
390  \add@special{\#1}%
391  \AddtoPrivateOthers{\#1}{ a macro to be really defined in gmdoc.}
392  \@xa
393  \xdef\csname_cc\string#\#1\endcsname{\the\catcode`#\#1}%
394  \begingroup
395  \catcode`\~\active_\lccode`\~`#\#1%
396  \lowercase{%
397    \global\@xa\let
398    \csname_ac\string#\#1\endcsname%
399    \@xa\gdef\@xa~\@xa{%
400      \@xa\ifmmode\@xa\string\@xa~%
401      \@xa\else\@xa\afterfi{\@shortvrbdef~}\fi}}% This terrible number
402          of \expandafters is to make the shortverb char just other in the math
403          mode (my addition).
404  \endgroup
405  \global\catcode`\#1\active
406  \else
407  \@shortvrbinfo{\empty\#1already}{\empty\verb(*)}%
408  \fi}
409
\DeleteShortVerb{#1}%
410  \def\DeleteShortVerb#1{%
411    \@xa\ifx\csname_cc\string#\#1\endcsname\relax
412    \@shortvrbinfo{\empty\#1not}{\empty\verb(*)}%
413    \else
414    \@shortvrbinfo{Deleted_\#1as}{\empty\verb(*)}%
415    \rem@special{\#1}%
416    \global\catcode`\#1\csname_cc\string#\#1\endcsname
417    \global_\@xa\let_\csname_cc\string#\#1\endcsname_\relax
418    \ifnum\catcode`\#1=\active
419    \begingroup
420    \catcode`\~\active_\lccode`\~`#\#1%
421    \lowercase{%
422      \global\@xa\let\@xa~%
423      \csname_ac\string#\#1\endcsname}%
424    \endgroup_\fi_\fi}
425
My little addition
426
427  \@ifpackageloaded{gmdoc}{%
428    \def\gmv@packname{gmdoc}%
429    \def\gmv@packname{gmverb}%
430
\@shortvrbinfo{#1#2#3}{%
431    \PackageInfo{\gmv@packname}{%
432      ^\#1\empty_\#1\@xa\@gobble\string#\#2_a_short_reference
433      for_\#1\@xa\string#\#3}%
434
\add@special{#1}{%
435    \def\add@special#1{%
436      \rem@special{\#1}%
437      \@xa\gdef\@xa\dospecials\@xa
438      {\dospecials_\do_\#1}%
439      \@xa\gdef\@xa\@sanitize\@xa
440      {\@sanitize_\@makeother_\#1}%
441
For the commentary on the below macro see the doc package's documentation. Here
442 let's only say it's just amazing: so tricky and wicked use of \do. The internal macro
443
444
445
446
File e: gmverb.sty Date: 2008/08/30 Version vo.89
177

```

\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 457 \def\rem@special#1{%
458   \def\do##1{%
459     \ifnum`#1=\#\else\@nx\do\@nx##1\fi}%
460   \xdef\dospecials{\dospecials}%
461   \begingroup
462   \def\@makeother##1{%
463     \ifnum`#1=\#\else\@nx\@makeother\@nx##1\fi}%
464   \xdef\@sanitize{\@sanitize}%
465   \endgroup}
```

And now the definition of \verb+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+verbatim+ inside them. I.e., as with the verbatim package, you may write \verb+verbatim+ in the begdef of your environment and then necessarily \endverb+ in its enddef. Of course (or maybe surprisingly), the commands written in the begdef after \verb+verbatim+ will also be executed at \begin{environment}.

```
verbatim 478 \def\verb+verbatim+{%
\verb+verbatim+ 479   \edef\gmv@hyphenpe{\the\hyphenpenalty}%
480   \edef\gmv@exhyphenpe{\the\exhyphenpenalty}%
481   \begin{parpenalty}\predisplaypenalty\verb+verbatim+
482   \frenchspacing\gmobeyspaces\verb+xverbatim%
483   \hyphenpenalty=\gmv@hyphenpe\relax
484   \exhyphenpenalty=\gmv@exhyphenpe
485   \hyphenchar\font=\m@ne}%
in the LATEX version there's \vobeyspaces instead of \gmobeyspaces.
verbatim* 490 \namedef{verb+verbatim+*}{\begin{parpenalty}\predisplaypenalty\%
\verb+verbatim+
\@sxverb+}
\endverb+ 493 \def\endverb+{\@@par
494   \ifdim\lastskip>\z@
495   \tempskip\lastskip\vskip-\lastskip
496   \advance\tempskip\parskip\advance\tempskip-%
\outerparskip
497   \vskip\tempskip
498   \fi
499   \addvspace\topsepadd
500   \endparenv}
503 \n@melet{\endverb+*}{\endverb+}
506 \begin{group}\catcode`!=\o%
507 \catcode`[=\i\catcode`]=\z%
508 \catcode`\{=\active
509 \makeother\}%
510 \catcode`\\"=\active%
511 !gdef!\verb+xverb+[%
512   !edef!\verb+verbatim+@edef[%%
513   !def!noexpand!\verb+verbatim+@end%
514   #####1!noexpand\end!noexpand{\currenvir}[%
```

```

515     #####1!noexpand!end[!@currenvir]]]%
516     !verbatim@edef
517     !verbatim@end]%
518 !endgroup
522 \let\@sxverbatim=\@xverbatim
      F. Mittelbach says the below is copied almost verbatim from LATEX source, modulo
      \check@percent.

\@verbatim 527 \def\@verbatim{%
      Originally here was just \trivlist\item[], but it worked badly in my docu-
      ment(s), so let's take just highlights of if.
      533 \parsep\parskip
      From \trivlist:
      535 \if@noskipsec\leavevmode\fi
      536 \topsepadd\topsep
      537 \ifvmode
      538   \advance\topsepadd\partopsep
      539 \else
      540   \unskip\par
      541 \fi
      542 \topsep\topsepadd
      543 \advance\topsep\parskip
      544 \outerparskip\parskip
      (End of \trivlistlist and \trivlist highlights.)
      546 \@@par\addvspace\topsep
      547 \if@minipage\else\vskip\parskip\fi
      548 \leftmargin\parindent% please notify me if it's a bad idea.
      549 \advance\totalleftmargin\leftmargin
      550 \raggedright
      551 \leftskip\totalleftmargin% so many assignments to preserve the list
      thinking for possible future changes. However, we may be sure no inter-
      nal list shall use \totalleftmargin as far as no inner environments are
      possible in verbatim(*) .
      555 \@@par% most probably redundant.
      558 \tempswafalse
      559 \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
        \ifvmode\if@tempswa\addvspace\stanzaskip\tempswafalse\fi\fi
        \@@par
        \penalty\interlinepenalty\check@percent}%
      567 \everypar{\tempswatrue\hangindent\verbatimhangindent\hangafter%
        \one}%
        since several chars are breakable, there's a possibility of breaking
        some lines. We wish them to be hanging indented.
      570 \obeylines
      571 \ttverbatim}
\stanzaskip 573 \ifundefined{stanzaskip}{\newlength\stanzaskip}{}%
574 \stanzaskip=\medskipamount
\verbatimhangindent 578 \newlength\verbatimhangindent

```

```

579 \verbatimhangindent=3em
\check@percent 581 \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.

```
\AddtoPrivateOthers 588 \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* 597 \def\verb{\relax\ifmmode\hbox\else\leavevmode\null\fi
\verb 598 \bgroup
599 \ttverbatim
600 \gm@verb@eol
601 \@ifstar{\@sverb@chbsl}{\gmobyspaces\frenchspacing@sverb@chbsl}}% in
the LATEX version there's \@vobyspaces instead of \gmobyspaces.
@sverb@chbsl 605 \def\@sverb@chbsl#1{\@sverb#1\check@bslash}
\def@breakbslash 608 \def\@def@breakbslash{\breakbslash}% because \ is \defined as \break-
slash 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.

```

\check@bslash 614 \def\check@bslash{\ifnextchar{\@def@breakbslash}{\bslash%
\@gobble}{}}
618 \let\verb@balance@group@\empty
\verb@egroup 621 \def\verb@egroup{\global\let\verb@balance@group@\empty\egroup}
\gm@verb@eol 625 \let\gm@verb@eol\verb@eol@error
```

The latter is a LATEX 2_E 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.

```

\begin{group} 637 \begingroup
\obeylines\obeyspaces% 638
\gdef\verb@eolOK{\obeylines% 639
\def^M{\check@percent}}% 640
}% 641
\endgroup 642
```

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:

```
\verb@eolOK 647 \def\verb@eolOK{\let\gm@verb@eol\verb@eolOK}
```

And back to the main matter,

```

650 \def\@sverb#1{%
651 \catcode`#1\active\lccode`\~`#1%
```

```

652 \gdef\verb@balance@group{\verb@egroup%
653   \@latex@error{Illegal use of \bslash_verb_ command}\@ehc}%
654 \aftergroup\verb@balance@group
655 \lowercase{\let~\verb@egroup{}}

\verbatim@nolig@list
657 \def\verbatim@nolig@list{\do\`\do\<\do\>\do\,,\do\`\do\-\}

\do@noligs
659 \def\do@noligs#1{%
660   \catcode`#1\active
661   \begingroup
662   \lccode`\~=`#1\relax
663   \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
668 \def\dekclubs{\ifstar{\MakeShortVerb*}{\MakeShortVerb}}
\olddekclubs
669 \def\olddekclubs{\OldMakeShortVerb}

```

But even if a shortverb is unconditional, the spaces in the math mode are not printed. So,

```

\edverbs
677 \newcommand*\edverbs{%
678   \let\gmv@dismath\[%%
679   \let\gmv@edismath\]%
680   \def\[{%
681     \@ifnextac\gmv@disverb\gmv@dismath}%
682   \relaxen\edverbs}%

\gmv@disverb
684 \def\gmv@disverb{%
685   \gmv@dismath
687   \hbox\bgroup\def\[]{\egroup\gmv@edismath}}

```

doc- and shortverb-compatibility

One of minor errors while \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
699 \def\old@MakeShortVerb#1{%
700   \@xa\ifx\csname_cc\string#1\endcsname\relax
701   \@shortvrbinfo{Made_}{#1}\@shortvrbdef
702   \add@special{#1}%
703   \AddtoPrivate@Others#1% a macro to be really defined in gmdoc.
704   \@xa
705   \xdef\csname_cc\string#1\endcsname{\the\catcode`#1}%
706   \begingroup
707   \catcode`\~\active\lccode`\~`#1%
708   \lowercase{%
709     \global\@xa\let\csname_ac\string#1\endcsname\%
710     \@xa\gdef\@xa~\@xa{%
711       \@shortvrbdef~}}%
712   \endgroup
713   \global\catcode`#1\active
714   \else
715   \@shortvrbinfo{\empty{#1already}}{\empty{\verb(*)}}%
716   \fi}

```

```

\OldMakeShortVerb 720 \def\OldMakeShortVerb{\begingroup
721   \let\@MakeShortVerb=\old@MakeShortVerb
722   \@ifstar{\eg@MakeShortVerbStar}{\eg@MakeShortVerb}}
725 \def\eg@MakeShortVerbStar#1{\MakeShortVerb*#1\endgroup}
726 \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 recognize possible redefinition of it when xltextra is loaded.

```

738 \let\gmd@preambleABD\AtBeginDocument
739 \AtBeginDocument{\let\gmd@preambleABD\firstofone}
741 \RequirePackage{xcolor}% for \providecolor
\VisSpacesGrey 743 \def\VisSpacesGrey{%
745   \providecolor{visspacesgrey}{gray}{0.5}%
746   \gmd@preambleABD{%
747     \edef\visiblespace{%
748       \hbox{\@nx\textcolor{visspacesgrey}{%
749         {\@xa\unexpanded\@xa{\visiblespace}}}}}%
750   }%
756 \endinput% for the Tradition.

```

f. The gmeometric Package¹

Written by Grzegorz Murzynowski,
natror at o2 dot pl

© 2006, 2007 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>
for the details of that license.

LPPL status: "author-maintained".

```
55 \NeedsTeXFormat{LaTeX2e}
56 \ProvidesPackage{gmeometric}
57     [2008/08/06 vo.72 to allow the `geometry' macro in the
      document (GM)]
```

Introduction, usage

This package allows you to use the \geometry macro, provided by the geometry v3.2 by Hideo Umeki, anywhere in a document: originally it's clause \onlypreamble and the main work of gmeometric is to change that.

Note it's rather queer to change the page layout *inside* a document and it should be considered as drugs or alcohol: it's O.K. only if you *really* know what you're doing.

In order to work properly, the macro should launch the \clearpage or the \cleardoublepage to 'commit' the changes. So, the unstarred version triggers the first while the starred the latter. If that doesn't work quite as expected, try to precede or succeed it with \onecolumn or \twocolumn.

It's important that \clear(double)page launched by \geometry not to be a no-op, i.e., \clear(double)page immediately preceding \geometry (nothing is printed in between) discards the 'commitment'.

You may use gmeometric just like geometry i.e., to specify the layout as the package options: they shall be passed to geometry.

This package also checks if the engine is X_ET_EX and sets the proper driver if so. Probably it's redundant since decent X_ET_EX packages provide their geometry.cfg file that does that.

The remarks about installation and compiling of the documentation are analogous to those in the chapter gmdoc.sty and therefore omitted.

Contents of the gmeometric.zip archive

The distribution of the gmeometric package consists of the following four files.

gmeometric.sty
README
gmeometric.pdf

¹ This file has version number vo.72 dated 2008/08/06.

gmeometric.tds.zip

Usage

The main use of this package is to allow the \geometry command also inside the document (originally it's \onlypreamble). To make \geometry work properly is quite a different business. It may be advisable to 'commit' the layout changes with \newpage, \clearpage, or \cleardoublepage and maybe \one/twocolumn.

Some layout commands should be put before \one/twocolumn and other after it. An example:

```
\thispagestyle{empty}
\advance\textheight 3.4cm\relax
\onecolumn
\newpage
\advance\footskip-1.7cm
\geometry{hmargin=1.2cm,vmargin=1cm}
\clearpage
```

And another:

```
\newpage
\geometry{bottom=3.6cm}
```

In some cases it doesn't work perfectly anyway. Well, the (LPPL) license warns about it.

The code

```
176 \RequirePackage{gmutils}[2007/04/23] % this package defines the storing and
restoring commands.
```

redefine \onlypreamble, add storing to BeginDocument.

```
\gme@tobestored
180 \newcommand*\gme@tobestored[{\%
181   \Gm@cnth\Gm@cntv\c@Gm@tempcnt\Gm@bindingoffset\Gm@wd@mp
182   \Gm@odd@mp\Gm@even@mp\Gm@orgw\Gm@orgh\Gm@dimlist}\}
185 \x@AtBeginDocument\x@\{ \x@\StoreMacros\gme@tobestored\}
187 \StoreMacro\onlypreamble
188 \let\onlypreamble\gobble
```

To make it work properly in X_ET_EX:

```
191 \@ifXeTeX{%
192   \ifundefined{pdfoutput}{\newcount\pdfoutput}{}%
193   \PassOptionsToPackage{dvipdfm}{geometry}%
194 }{%
196 \RequirePackageWithOptions{geometry}
199 \RestoreMacro\onlypreamble
```

Hypothesis: \ifx...@\undefined fails in the document because something made \csname\Gm@lines\endcsname. So we change the test to decent. And i think I've found the guilty: \ifundefined in \Gm@showparams. So I change it to the more elegant \ifx@\undefined.

```

\Gm@showparams 336 \def\Gm@showparams{%
340   ----- Geometry parameters ^^J%
341   \ifGm@pass
342     'pass' is specified!! (disables the geometry layouter) ^^J%
343   \else
344     paper:\ifx\Gm@paper\@undefined\class\default\else\Gm@paper%
345       \fi^^J%
346     \Gm@checkbool{landscape}%
347     twocolumn:\if@twocolumn\Gm@true\else--\fi^^J%
348     twoside:\if@twoside\Gm@true\else--\fi^^J%
349     asymmetric:\if@mparswitch--\else\if@twoside\Gm@true\else--%
350       \fi\fi^^J%
351     h-parts:\Gm@lmargin,\Gm@width,\Gm@rmargin%
352     \ifnum\Gm@cnth=\z@\space(default)\fi^^J%
353     v-parts:\Gm@tmargin,\Gm@height,\Gm@bmargin%
354     \ifnum\Gm@cntv=\z@\space(default)\fi^^J%
355     hmarginratio:\ifnum\Gm@cnth<5\ifnum\Gm@cnth=3--\else%
356       \Gm@hmarginratio\fi\else--\fi^^J%
357     vmarginratio:\ifnum\Gm@cntv<5\ifnum\Gm@cntv=3--\else%
358       \Gm@vmarginratio\fi\else--\fi^^J%
359     lines:\ifx\Gm@lines\@undefined--\else\Gm@lines\fi^^J% here I (na-
360       tor) fix the bug: it was \@ifundefined that of course was assigning
361       \% \relax to \Gm@lines and that resulted in an error when \geometry was
362       used inside document.
363     \Gm@checkbool{heightrounded}%
364     bindingoffset:\the\Gm@bindingoffset^^J%
365     truedimen:\ifx\Gm@truedimen\@empty--\else\Gm@true\fi^^J%
366     \Gm@checkbool{includehead}%
367     \Gm@checkbool{includefoot}%
368     \Gm@checkbool{includemp}%
369     driver:\Gm@driver^^J%
370   \fi
371   ----- Page layout dimensions and switches ^^J%
372   \string\paperwidth\space\space\the\paperwidth^^J%
373   \string\paperheight\space\the\paperheight^^J%
374   \string\textwidth\space\space\the\textwidth^^J%
375   \string\textheight\space\the\textheight^^J%
376   \string\oddsidemargin\space\space\the\oddsidemargin^^J%
377   \string\evensidemargin\space\the\evensidemargin^^J%
378   \string\topmargin\space\space\the\topmargin^^J%
379   \string\headheight\space\the\headheight^^J%
380   \string\headsep\@spaces\the\headsep^^J%
381   \string\footskip\space\space\space\the\footskip^^J%
382   \string\marginparwidth\space\the\marginparwidth^^J%
383   \string\marginparsep\space\space\space\the\marginparsep^^J%
384   \string\columnsep\space\space\the\columnsep^^J%
385   \string\skip\string\footins\space\space\the\skip\footins^^J%
386   \string\hoffset\space\the\hoffset^^J%
387   \string\voffset\space\the\voffset^^J%
388   \string\mag\space\the\mag^^J%
389   \if@twocolumn\string\@twocolumntrue\space\fi%
390   \if@twoside\string\@twosidetrue\space\fi%
391   \if@mparswitch\string\@mparswitchtrue\space\fi%

```

```
391 \if@reversemargin\string\@reversemargintrue\space\fi^^J%
392 (1in=72.27pt,1cm=28.45pt)^^J%
393 -----}
```

Add restore to BeginDocument:

```
397 \AtBeginDocument{\RestoreMacros\tobestored}
398 \endinput
```

g. The gmoldcomm Package¹

October 6, 2008

This is a package for handling the old comments in LATEX 2_E Source Files when LATEX 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 LATEX 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 vo.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 }{%
43   {\escapechar\m@ne
44   \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}
\gmc@checkenv 52 [\def\gmc@checkenv{%
53   \ifnextchar{%
54     [\gmc@checkenvinn] []}%
55     [\gmc@checkenvinn] []}%
56   \def\gmc@checkenvinn[#1]{%
57     \def\gmc@resa[#1]{%
58       \ifx\gmc@resa[#1]\gmc@maccname
59         \def\gmc@resa[#1]{%
60           \def\gmc@next{%
61             \begingroup
62 }
```

¹ This file has version number vo.99 dated 2007/11/10.

```

63     \def\@currenvir[macrocode]%
64     \RestoreMacro\finish@macroscan
65     \catcode`\\=\z@
66     \catcode`\{=\let\catcode`\}=
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     \next]%
78 ]
79
80 \foone{%
81     \catcode`/=\z@
82     \catcode`\\=\active}
83 {/def/gmoc@defbslash{%
84     /let\scan@macro}}}
85
86 \gmoc@defbslash
87
88 \task
89 \def\task#1#2{}
90
91 \endinput
92

```

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. 83.
 a-5978

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-7743

gmdoc v0.99c
General:
 A bug fixed in \DocInput and all \expandafters changed to \@xa and \noexpands to \@nx, a-7743
 The TeX-related logos now are declared with \DeclareLogo provided in gmutils, a-7743

\DocInput:
 added ensuring the code delimiter to be the same at the end as at the beginning, a-2398

\gmd@bslashEOL:
 a bug fix: redefinition of it left solely to \QueerEOL, a-3403

gmdoc v0.99d
General:
 \@namelet renamed to \n@melet to solve a conflict with the beamer class (in gmutils at first), a-7743
 \afterfi & pals made two-argument, a-7743

\FileInfo:
 added, a-6850

gmdoc v0.99e
General:

a bug fixed in \DocInput and \IndexInput, a-7743
 CheckSum 4574, a-0

gmdoc v0.99g
General:
 CheckSum 5229, a-0
 The bundle goes XeTeX. The TeX-related logos now are moved to gmutils. ^^A becomes more comment-like thanks to re\catcode'ing. Automatic detection of definitions implemented, a-7743

\gmd@ifinmeaning:
 made more elegant: \if changed to \ifx made four parameters and not expanding to an open \iftrue/false. Also renamed from \@ifismember, a-3627

hyperref:
 added bypass of encoding for loading url, a-2118

\inverb:
 added, a-7032

\OldDocInput:
 obsolete redefinition of the macro environment removed, a-7589

gmdoc v0.99h
General:
 Fixed behaviour of sectioning commands (optional two heading skip check) of mwcls/gmutils and respective macro added in gmdocc. I made a tds archive, a-7743

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-7743
 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-6999

gmdoc vo.99k

- General:
 - CheckSum 5261, a-0
- hyperref:**
 - removed some lines testing if \TeX colliding with tikz and most probably obsolete, a-2136

gmdoc vo.99l

- General:
 - CheckSum 5225, a-0
- \CodeSpacesGrey:**
 - added due to Will Robertson's suggestion, a-2590
- codespacesgrey:**
 - added due to Will Robertson's suggestion, a-2097
- \gmd@Firescan:**
 - \scantokens used instead of \write and \@input which simplified the macro, a-6882
- macrocode:**
 - removed \CodeSpacesBlank, a-5073
- \SelfInclude:**
 - Made a shorthand for $\text{\Docinclude}\text{\jobname}$ instead of repeating 99% of \DocInclude's code, a-6590

gmdoc vo.99m

- \@oldmacrocode:**
 - renamed from \VerbMacrocodes, a-5165
- $\text{\^\^M}:$**
 - there was $\text{\let}\text{\^\^M}$ but \QueerEOL is better: it also redefines \^\^M , a-2381
- General:
 - CheckSum 5354, a-0
 - CheckSum 5356, a-0
 - Counting of all lines developed (the countalllines package option), now it uses \inputlineno, a-7743
- \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-4892
- \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-6526
- \FileInfo:**
 - \egroup of the inner macro moved to the end to allow \gmd@ctallsetup. From the material passed to

\gmd@Firescan ending \^\^M stripped not to cause double labels., a-6867

\gmd@bslashEOL:

- also \StraightEOL with countalllines package option lets \^\^M to it, a-3403

\thefilediv:

- let to \relax by default, a-6740

theglossary:

- added \IndexLinksBlack, a-6049

gmdoc vo.99n

- General:
 - CheckSum 5409, a-0
 - CheckSum 5547, a-0
 - Inline comments' alignment developed, a-7743
- \CS:**
 - added, a-7123
- \CSes:**
 - added, a-7131
- \CSS:**
 - added, a-7129
- enumargs:**
 - developed for the case of inline comment, a-7191
- \finish@macroscan:**
 - the case of _ taken care of, a-3724
- \gmd@eatlspace:**
 - \afterfifi added—a bug fix, a-2857
- \gmd@nlperc:**
 - added \hboxes in \discretionary to score \hyphenpenalty not \exhyphenpenalty, a-7048
- \gmd@percenthack:**
 - \space replaced with a tilde to forbid a linebreak before an inline comment, a-2960
- \HideDef:**
 - added the starred version that calls \UnDef, a-4243
- \HideDefining:**
 - Added the starred version that hides the defining command only once, a-4404
- \ilrr:**
 - added, a-3074
- \nostanza:**
 - added adding negative skip if in vmode and \par, a-2319
- \UnDef:**
 - a bug fixed: \gmd@charbychar appended to \next—without it a subsequent inline comment was typeset verbatim, a-4234
- \verbcodecorr:**
 - added, a-3095

gmdoc vo.99o

\@codetonarrskip:

a bug fix: added \nostanzagtrue, a-3220

enumargs:
added the optional argument which is the number of hashes (1 by default or 2 or 4), a-7191

gmdoc v0.99p
General:
CheckSum 5607, a-o

\DeclareDocumentCommand:
added, a-4263

enumargs:
added optional arguments' handling, a-7191

gmdoct v0.74
\edverbs:
used to simplify displaying shortverbs, b-450

gmdoct v0.75
General:
CheckSum 130, b-o

gmdoct v0.76
General:
CheckSum 257, b-o

\EOFMark:
The gmeometric option made obsolete and the gmeometric package is loaded always, for XeTeX-compatibility. And the class options go xkeyval., b-469

gmdoct v0.77
General:
CheckSum 262, b-o

\EOFMark:
Bug fix of sectioning commands in mwcls and the default font encoding for TeXing old way changed from QX to T1 because of the 'corrupted NTFs tables' error, b-469

gmdoct v0.78
General:
CheckSum 267, b-o

\EOFMark:
Added the pagella option not to use Adobe Minion Pro that is not freely licensed, b-469

gmdoct v0.79
General:
CheckSum 271, b-o

gmdoct v0.80
General:
CheckSum 276, b-o

gmcc@fontspec:
added, b-259

gmeometric v0.69
General:
CheckSum 40, f-o

gmeometric v0.70
General:
Back to the v0.68 settings because \not@onlypreamble was far too little. Well, in this version the redefinition of \geometry is given up since the 'committing' commands depend on the particular situation so defining only two options doesn't seem advisable, f-399
CheckSum 36, f-o

gmeometric v0.71
General:
a tds-compliant zip archive made, f-399
CheckSum 41, f-o

gmeometric v0.72
General:
2008/08/06 only the way of documenting changes so I don't increase the version number, f-399
CheckSum 239, f-o

\Gm@showparams:
a bug fix:
\@ifundefined{Gm@lines} raised an error when \geometry used inside the document, I change it to \ifx\@undefined, f-336

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-1005

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-3883

gmutils v0.75
\@ifnextcat:
\let for #1 changed to \def to allow things like \noexpand~, c-775

\@ifnextif:
\let for #1 changed to \def to allow things like \noexpand~, c-811

\@ifnif:
added, c-836

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-3883

\freeze@actives:
added, c-2897

gmutils vo.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-3883

gmutils vo.78
General:
\@namelet renamed to \n@melet to solve a conflict with the beamer class. The package contents regrouped, c-3883

gmutils vo.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-1594

gmutils vo.80
General:
CheckSum 1689, c-0

\hfillneg:
added, c-2818

gmutils vo.81
\dekfracslash:
moved here from pmlectionis.cls, c-3076

\ifSecondClass:
moved here from pmlectionis.cls, c-3048

gmutils vo.82
\ikern:
added, c-3084

gmutils vo.83
\~:
postponed to \begin{document} to avoid overwriting by a text command and made sensible to a subsequent /, c-2771

gmutils vo.84
General:
CheckSum 2684, c-0

gmutils vo.85
General:
CheckSum 2795, c-0
fixed behaviour of too clever headings with gmdoc by adding an \ifdim test, c-3883

gmutils vo.86
\texttilde:
renamed from \~ since the latter is one of LATEX accents, c-2779

gmutils vo.87

General:
CheckSum 4027, c-0
the package goes ε-TEx even more, making use of \ifdefined and the code using UTF-8 chars is wrapped in a XeTEX-condition, c-3883

gmutils vo.88
General:
CheckSum 4040, c-0

\RestoreEnvironment:
added, c-1533

\storedcsname:
added, c-1524

\StoreEnvironment:
added, c-1529

gmutils vo.89
General:
removed obsolete adjustment of pgf for XTEX, c-3883

gmutils vo.90
General:
CheckSum 4035, c-0

\XeTeXthree:
adjusted to the redefinition of \verb in xltxtra 2008/07/29, c-2412

gmutils vo.91
General:
CheckSum 4055, c-0
removed \jobnamewoe since \jobname is always without extension. \xiinspace forked to \visiblespace\let to \xxt@visiblespace of xltxtra if available. The documentation driver integrated with the .sty file, c-3883

gmutils vo.92
\@checkend:
shortened thanks to \@ifenvir, c-1037

\@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-276

\@ifenvir:
added, c-1029

General:
CheckSum 4133, c-0

gmutils vo.93
\@nameedef:
added, c-224

General:
A couple of \DeclareRobustCommand* changed to \pdef, c-3883

CheckSum 4140, c-0

CheckSum 4501, c-0

The numerical macros commented out as obsolete and never really used, c-3883

\ampulexdef:
 added, c-668

\em:
 added, c-3767, c-3776

\gmu@RPfor:
 renamed from \gmu@RPIf and #3
 changed from a csname to cs, c-2959

\litshape:
 copied here from E. Szarzyński's *The Letters*, c-3730

\lsl:
 copied here from E. Szarzyński's *The Letters*, c-3751

\nocite:
 a bug fixed: with natbib an 'extra)' error. Now it fixes only the standard version of \nocite, c-1632

\pdef:
 added, c-182

\pprovide:
 added, c-211

\provide:
 added, c-196

\textlit:
 added, c-3745

\thousep:
 added, c-3782

gmutils vo.94

\@xau:
 added, c-178

General:

 \begin{group} and \end{group} 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-3883

CheckSum 4880, c-0

removed \unex@namedef and \unex@nameuse, probably never really used since they were incomplete: \edef@other undefined, c-3883

The code from ancient xparse (1999) of TeXLive 2007 rewritten here, c-3883

\afterfifi:
 \if removed from parameters' string, c-916

\ampulexdef:
 made xparse-ish and \ampulexset
 removed, c-668

\dekfrac:
 made to work also in math mode, even with math-active digits, c-2504

\gm@ifundefined:

added. All \ifundefined used by me changed to this, c-933 made robust to unbalanced \ifs and \fis the same way as L^AT_EX's \ifundefined (after a heavy debug :-), c-933

\gmath:
 removed definition of \langleletter>s and \langle digit>s, c-2563

\ldate:
 \leftline replaced with \par... \par to work well with floatfl, c-3606

\prependtomacro:
 order of arguments reversed, c-373

\resizegraphics:
 \includegraphics works well in X_ET_EX so I remove the complicated version with \XeTeXpicfile, c-2533

\textbullet:
 the X_ET_EX version enriched with \iffontchar due to lack of bullets with the default settings reported by Morten Høgholm and Edd Barrett, c-3415

gmverb vo.79

\edverbs:
 added, e-669

gmverb vo.80

\edverbs:
 debugged, i.e. \hbox added back and redefinition of \[, e-669

\ttverbatim:
 \ttverbatim@hook added, e-343

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-756

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 286), e-756

 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-756

 CheckSum 673, e-0

gmverb vo.87

General:

CheckSum 661, e-o
visible space tidyied and taken from
`xltxtra` if available. gutils required.
The `\xii...` cses moved to gutils.
The documentation driver moved
into the .sty file, e-756

gmverb vo.88

General:

CheckSum 682, e-o
`\VisSpacesGrey`:

added, or rather moved here from
`gmdoc`, e-743

gmverb vo.89

General:

`\dekclubs`, `\dekclubs*` and
`\olddekclubs` made more
consistent, shorthands for
`\MakeShortVerb\|`,
`\MakeShortVerb*\|` and
`\OldMakeShortVerb\|`
respectively., e-756

CheckSum 686, e-o

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.

*, a-3507, a-3791, a-5725,
 a-7512, c-864, c-865,
 c-866, c-868, c-872,
 c-873, c-874, c-878,
 c-2771, c-2952
\+, 21, a-5725, a-7116, c-2953
\-, a-5725, c-1322, c-3403,
 e-657
\<...>, c-1253, 104
\@codeline@wrindex,
 a-4922
\@nil, a-4736, a-4822,
 a-5244, a-5262,
 a-5882, a-5940,
 a-5943, a-5962,
 a-6879, c-1279,
 c-1281, c-1283,
 c-2826, c-3111,
 c-3121, c-3183,
 c-3188, c-3191,
 c-3192, c-3196,
 c-3442, c-3460,
 c-3491, c-3497,
 c-3511, c-3517,
 c-3538, c-3543
\@par, a-2461, a-3030,
 a-3048, a-3155, a-5382
\@settexcodehangi,
 a-2217, a-2217,
 a-2715, a-2777
\@EOF, a-7727, a-7730
\@M, a-5946, c-1937
\@MakeShortVerb, a-7592,
 e-383, e-384, e-387,
 e-721
\@NoEOF, a-7725, a-7729
\@alph, a-6488, a-6489
\@aftercodegfalse,
 a-2748, a-3037, a-3220
\@aftercodegtrue,
 a-2324, a-2755,
 a-2784, a-3006,
 a-5737, a-5771
\@afterheading, c-1930
\@afternarrgfalse,
 a-2324, a-2755,
 a-3006, a-5737, a-5771
\@afternarrgtrue, a-2437
\@badend, c-1037
\@beginputhook, a-2386,
 a-2492, a-2493
\@begnamedgroup, c-981,
 c-998, c-1003
\@begnamedgroup@ifcs,
 c-999, c-1002
\@car, c-2280
\@cclv, c-2935
\@cclvi, c-2920
\@charlb, a-6474
\@charrb, a-6476
\@checkend, c-1037
\@clsextension, c-3048
\@clubpenalty, a-2368,
 c-1942
\@codeskipput, 42
\@codeskipputfalse,
 a-2437, a-2726,
 a-3007, a-5737,
 a-5772, a-6993
\@codeskipputtrue,
 a-2308, a-2314,
 a-2323, a-2728,
 a-3156, a-5061,
 a-5072, a-5204, a-5211
\@codetonarrskip,
 a-2388, a-2608,
 a-2629, a-2982,
 a-3003, a-3047,
 a-3066, a-3201, a-3247
\@countalllinestrue,
 a-2028, a-2032
\@ctrerr, a-6495
\@currenvir, a-5130,
 a-5153, a-5154,
 a-7006, a-7010,
 a-7184, c-984, c-1031,
 e-514, e-515, g-63
\@currenvir*, a-5123
\@currenvline, c-988
\@currsize, c-1081,
 c-1082, c-1083,
 c-1084, c-1085,
 c-1086, c-1087,
 c-1088, c-1089, c-1090
\@ddc, c-429, c-461, c-477,
 c-496, c-524, c-529, c-530
\@ddc@, c-431, c-453
\@ddc@C, c-549
\@ddc@O, c-533
\@ddc@c, c-538
\@ddc@c@, c-540, c-546, c-551
\@ddc@m, c-485, c-561
\@ddc@o, c-488
\@ddc@o@, c-490, c-493, c-535
\@ddc@s, c-480
\@ddc@x, c-590
\@ddc@xm, c-563
\@ddc@xmm, c-566
\@ddc@xmmmm, c-569
\@ddc@xmffff, c-572
\@ddc@xmffffmm, c-575
\@ddc@xmffffmmmm, c-578
\@ddc@xmffffmmmmmm, c-581
\@ddc@xmffffmmmmmm, c-584
\@ddc@xmffffmmmmmm, c-587
\@debugtrue, b-188
\@def@breakbslash,
 e-608, e-614
\@defentryze, a-3700,
 a-4204, a-4626,
 a-4633, a-4637, a-4942

```

\@docinclude, a-6375, a-6381      \@ifncat, c-782, c-785, c-800      \@noindextrue, a-2041
\@dsdirgfalse, a-2737,             \@ifnextMac, c-888,                  \@normalcr, c-3651
    a-2758, a-2826,                c-3310, c-3353      \@nostanzagfalse, a-3156
    a-2892, a-2973,                \@ifnextac, c-847, e-681      \@nostanzagtrue, a-2323,
    a-2988, a-2994, a-5189      \@ifnextcat, a-3558,                  a-3220
\@dsdirgtrue, a-2446, a-2721      a-3586, a-7125,                  \@nx, c-176
\@emptify, a-2535, a-2686,        a-7129, a-7131, c-775,      \@oarg, c-1357, c-1359, c-1360
    a-4531, a-4654,                  c-848, c-3850      \@oargsq, c-1357, c-1360,
    a-4751, a-4834,                \@ifnextif, c-811      c-1375
    a-4840, a-4841,                \@ifnextspace, c-871,      \@oldmacrocode, a-5124,
    a-5591, a-5920,                c-3309, c-3352      a-5149
    a-6484, a-6617,                \@ifnif, c-818, c-821, c-836      \@oldmacrocode@launch,
    a-6767, a-7036,                \@ifnotmw, b-433, c-1751,      a-5101, a-5103, a-5106
    a-7042, a-7082,                c-1776, c-1925,      \@onlypreamble, a-6611,
    a-7084, a-7091,                c-2030, c-2117, c-2172      a-7395, a-7408,
    a-7093, a-7181,                \@ifstarl, a-3792, a-3801,      a-7412, c-1659,
    a-7187, a-7539,                a-4408, a-4618,      c-2875, c-3880, f-187,
    a-7540, a-7544,                a-4659, a-4676,      f-188, f-199
    c-383, c-384, c-388, e-365      a-4723, a-4758,      \@pageincludexfalse,
    \@endinputhook, a-2414,          a-4775, a-4854,      a-4486
    a-2488, a-2489      a-4858, a-4970,      \@pageincludextrue,
\@enumctr, a-7203, a-7207,          a-4993, a-7300      a-5038
    c-2193, c-2194, c-2200      \@ilgroupfalse, a-2786,      \@pageindexfalse, a-7407
\@enumdepth, c-2189,          a-3379      a-3027,      \@pageindextrue, a-2046,
    c-2192, c-2193      \@ilgrouptrue, a-3027,      a-4551, a-7410
\@fileswfalse, a-6969      a-3076, a-3091      \@parg, c-1364, c-1366, c-1367
\@fileswoptions, c-3048      \@include, c-3118, c-3120      \@pargp, c-1364, c-1367, c-1376
\@firstofmany, a-4736,          \@indexallmacrostrue,      \@parindent, c-2196,
    a-4822, a-5882,          a-2055,      c-2197, c-2199,
    a-6879, c-2826,          \@itemdepth, c-2207,      c-2214, c-2215, c-2217
    c-3442, c-3460,          c-2210, c-2211      \@parsed@endenv, c-443,
    c-3491, c-3497,          \@itemitem, c-2211, c-2212,      c-447, c-597
    c-3511, c-3517      \@latexerr, a-6374      \@parsed@endenv@, c-448,
\@firstofone, c-455, c-615      \@linesnotnumtrue, a-2012,      c-450, c-598
\@fshdafalse, a-6815      \@ltxDocIncludetrue,      \@pkgextension, c-1619
\@fshdatrue, a-6813      a-6608      \@preamblecmds, c-1613,
\@gif, c-262, c-263, c-270      \@makefntext, a-6666,      c-1615, c-1628, c-1632
\@glossaryfile, a-4888      \@marginparsusedfalse,      \@printalllinenosfalse,
\@gmccnochangepstrue, b-200      a-2082,      a-2029
\@gmu@mmhboxtrue,          a-2045,      \@printalllinenosttrue,
    c-2506, c-3840,          \@minus, c-2040,      a-2033
\@if, c-285, c-286, c-289      c-2051, c-2053,      \@relaxen, a-2295, a-3108,
\@ifeOLactive, a-2517,          c-2058, c-2060,      a-3135, a-5531,
    a-2522, a-3278,          c-2067, c-2072,      a-5919, a-6029,
    a-3312, a-3454,          \@mparswitchtrue, f-390,      a-6439, a-6500,
\@ifQueerEOL, a-2504,          a-2759, a-2923,      a-6738, a-6739,
    a-2521, a-5647, a-6117,      a-2941, a-2951,      a-6740, a-7457,
\@ifXeTeX, b-244, b-312,          \@newlinefalse, a-2627,      a-7726, c-392, c-393,
    c-2406, c-2414,          a-2759, a-2923,      c-397
    c-2540, c-3073,          a-2941, a-2951,      \@reversemargintrue, f-391
    c-3414, f-191      \@newlinetrue, a-2444,      \@secondofthree, c-609,
\@ifempty, c-692, c-1872,          a-2720,          c-621
    c-1892, c-3095      \@nobreakfalse, c-1936,      \@shortvrbdef, e-383,
\@ifenvir, c-1029, c-1037      c-2873,          e-384, e-389, e-402,
\@ifl@aded, c-1618      \@nobreaktrue, c-1931,      e-701, e-712

```

File Key: a=gmdoc.sty, b=gmdocc.cls, c=gmutils.sty, d=gmiflink.sty, e=gmverb.sty,
f=gmeometric.sty, g=gmoldcomm.sty

\@shortvrbinfo, e-389,
e-408, e-414, e-416,
e-435, e-701, e-716
\@starttoc, c-2868
\@sverb@chbsl, a-7077,
e-601, e-605
\@tempcntb*, c-3805
\@tempdima, c-3214,
c-3218, c-3219,
c-3221, c-3222,
c-3226, c-3733,
c-3739, c-3754, c-3756
\@tempdima*, c-3219, c-3221
\@tempdimb, c-3215,
c-3217, c-3218
\@temptokena, c-427,
c-437, c-445, c-474,
c-475, c-517, c-518
\@temptokenb, c-413,
c-414, c-428, c-465,
c-469, c-471, c-501,
c-505, c-507, c-558
\@textsuperscript,
c-2547, c-2548
\@thirdofthree, c-613, c-622
\@toodeep, c-2190, c-2208
\@topnewpage, c-1832
\@topsep, e-542, e-543, e-546
\@topsepadd, e-499, e-536,
e-538, e-542
\@trimandstore, a-2447,
a-2607, a-3234,
a-3234, a-3242, a-3245
\@trimandstore@hash,
a-3235, a-3236
\@trimandstore@ne,
a-3242, a-3245
\@twocolumntrue, f-388
\@twosidetru, f-389
\@typeset@protect, c-454
\@undefined, c-158, f-344,
f-357
\@uresetlinecounttrue,
a-2019
\@usgentryze, a-4650,
a-4664, a-4670,
a-4727, a-4731,
a-4949, a-4983,
a-7308, a-7315
\@whilenum, c-2920, c-2935
\@xa, c-175, c-178
\@xau, c-178
\@xifncat, c-787, c-800
\@xifnif, c-823, c-836
\@zf@euenctrue, b-346
\^A, 7, a-3303

\^B, 7, a-3269
\^M, 7, a-3396
\^M, a-2381, a-2719
\aaalph, a-6488, a-6535
\abovedisplayskip, a-2266
\acro, a-7124, c-2987,
c-3022, c-3023, c-3027
\acrocore, c-3007, c-3017
\activespace, e-335
\actualchar, 20, a-3478,
a-3625, a-4494,
a-5844, a-5902,
a-5907, a-6334, a-7483
\adashes, c-3381, c-3381,
c-3383, c-3389
\add@special, e-390,
e-441, e-702
\addfontfeature, c-2436,
c-2467, c-2469,
c-2495, c-2542,
c-2721, c-2734,
c-3025, c-3233,
c-3739, c-3756
\addto@estoindex,
a-4632, a-4669,
a-4683, a-4941,
a-4948, a-4958
\addto@estomarginpar,
a-4797, a-4940,
a-4947, a-4952
\addto@macro, c-363, c-370
\addtoheading, c-1906
\addtomacro, a-4955,
a-4960, a-5101,
a-5102, a-5298, c-370,
c-2704
\AddtoPrivateOthers,
19, a-3436, e-391,
e-588, e-703
\addtotoks, c-379, c-2720
\AE, c-3201
\ae, b-362
\afterassignment, c-3841
\afterfi, a-1933, a-2518,
a-2522, a-2827,
a-2830, a-2925,
a-2927, a-3242,
a-3433, a-3558,
a-3572, a-3597,
a-3600, a-3632,
a-3633, a-4117,
a-4121, a-4125,
a-4714, a-5193,
a-5263, a-5266,

a-7007, a-7008,
a-7011, a-7012,
a-7168, b-310, b-384,
c-201, c-851, c-875,
c-876, c-914, c-946,
c-1003, c-1004,
c-1033, c-1034,
c-1271, c-1302,
c-1373, c-1416,
c-1493, c-2409,
c-2996, c-3012,
c-3097, c-3098,
c-3160, c-3188,
c-3308, c-3351,
c-3798, c-3821,
c-3850, e-402
\afterfifi, a-2857,
a-2859, a-5190,
a-5191, a-5353,
a-5363, c-198, c-199,
c-916, c-941, c-943,
c-1472, c-1477,
c-2408, c-3400
\afterfififi, c-924
\afteriffifi, a-2853, c-918
\afteriffififi, c-923
\afterififfififi, c-922
\AfterMacrocode, 23, a-7171
\grave, b-334, b-352
\hyphen, c-3403
\AKA, c-3023
\all@other, c-689, c-690,
c-699, c-700, c-701,
c-2382, c-2484, c-2486
\alpha, c-2567
\AlsoImplementation,
20, a-7427, a-7441
\AltMacroFont, a-7544
\ampulexdef, c-668, c-710,
c-725, c-1656, c-2691
\ampulexhash, c-729
\AmSTeX, 22, c-2317
\and, a-6718, a-6753
\arg, c-1372, c-1373
\article, b-174
\AtBeginDocument,
a-2094, a-2102,
a-2151, a-2288,
a-3624, a-4552,
a-4900, a-6103,
a-7394, b-243, b-315,
b-361, c-1191, c-1371,
c-1626, c-1662,
c-1722, c-2424,

c-2433, c-2777,
 c-2970, c-3304,
 c-3381, c-3383,
 c-3874, e-738, e-739,
 f-185, f-397
`\AtBeginInput`, 9, a-2492,
 a-2502, a-2536,
 a-3278, a-3312,
 a-3430, a-3451,
 a-5928, a-6643,
 a-6661, a-6966
`\AtBeginInputOnce`, 9,
 a-2538, a-7591
`\AtDIProllogue`, 20, a-5592
`\AtEndDocument`, a-6224
`\AtEndInput`, 9, a-2488,
 a-6191, a-7360, a-7385
`\AtEndOfPackage`, a-2093,
 a-2100
`\author`, a-1888, a-6712
`\AVerySpecialMacro`, a-7630

`\begin`, c-998
`\begin*`, c-998
`\belowdisplayshortskip`,
 a-2272, a-2274, a-2275
`\belowdisplayskip`, a-2271
`\beth`, b-344
`\bgcolor`, c-2725
`\BibTeX`, 22, c-2320
`\bidate`, c-3581, c-3592
`\Biggl`, c-2666
`\biggl`, c-2664
`\Biggr`, c-2667
`\biggr`, c-2665
`\Bigl`, c-2662
`\bigl`, c-2660
`\Bigr`, c-2663
`\bigr`, c-2661
`\bigskipamount`, c-2815,
 c-3594
`\boldmath`, c-2280
`\BooleanFalse`, c-483, c-631
`\BooleanTrue`, c-482, c-632
`\box`, c-2301, c-2659
`\breakablevisspace`,
 a-2571, a-2821,
 a-7065, e-330, e-336
`\breakbslash`, a-7067,
 e-299, e-307, e-319,
 e-608
`\breakbrace`, a-7069,
 e-278, e-286, e-313
`\bslash`, a-2701, a-3625,
 a-3744, a-4080,

 a-4282, a-4285,
 a-4286, a-4289,
 a-4291, a-4302,
 a-4324, a-4325,
 a-4326, a-4327,
 a-4334, a-4495,
 a-4538, a-4736,
 a-4750, a-4822,
 a-4833, a-5836,
 a-5871, a-5872,
 a-5882, a-5902,
 a-5904, a-7068,
 a-7115, a-7184,
 a-7259, a-7369,
 c-1170, c-1321,
 c-1424, c-1501,
 c-1525, c-1899,
 c-1900, c-1901,
 c-3348, c-3349,
 c-3360, c-3361, e-298,
 e-307, e-614, e-653, g-39
`\bullet`, c-3418, c-3419

`\c@ChangesStartDate`,
 a-5936, a-5941,
 a-5962, a-5964,
 a-5965, a-5967
`\c@CheckSum`, a-6161,
 a-6200, a-6205,
 a-6217, a-6237, a-6242
`\c@codelinenum`, a-3111,
 a-3115, a-4876,
 a-4890, a-7185
`\c@DocInputsCount`, a-3114
`\c@footnote`, a-6716, a-6763
`\c@GlossaryColumns`,
 a-6043, a-6043, a-6051
`\c@gm@PronounGender`,
 c-1689
`\c@Gm@tempcnt`, f-181
`\c@gmd@mc`, a-7162, a-7167,
 a-7168, a-7184
`\c@GMlabel`, d-149
`\c@IndexColumns`, a-5609,
 a-5609, a-5611, a-5641
`\c@NoNumSecs`, c-1724
`\c@secnumdepth`, c-1781
`\c@StandardModuleDepth`,
 a-7529
`\acute`, b-335, b-353
`\cataactive`, 21, a-6982
`\catletter`, 21, a-6984
`\catother`, 21, a-6979
`\CDAnd`, 23, a-7140
`\CDPerc`, 23, a-7142

 \changes, a-5829, a-5835,
 a-5840
`\changes@`, a-5833, a-5851
`\ChangesGeneral`, a-5922,
 a-5928
`\ChangesStart`, 17, a-5959
`\ChangesStartDate`, 17
`\Character@Table`,
 a-7251, a-7257
`\CharacterTable`, a-7249
`\check@bslash`, e-605, e-614
`\check@checksum`, a-6191,
 a-6194
`\check@percent`, a-3430,
 e-566, e-581, e-640
`\check@sum`, a-6157,
 a-6159, a-6195,
 a-6205, a-6216, a-6227
`\CheckModules`, a-7540
`\CheckSum`, a-6161
`\CheckSum`, 17, a-6159, a-6248
`\ChneOelze`, a-4315
`\chschange`, a-6234,
 a-6238, a-6245
`\chunkskip`, 18, 22, a-2305
`\class`, b-157
`\ClassError`, c-1898
`\cleardoublepage`, c-1738
`\clubpenalty`, a-2368,
 a-2484, c-1937, c-1942
`\cmd`, c-1346
`\cmd@to@cs`, c-1346, c-1348
`\Code@CommonIndex`,
 a-4676, a-4679
`\Code@CommonIndexStar`,
 a-4676, a-4682
`\Code@DefEnvir`, a-4854,
 a-4938
`\Code@DefIndex`, a-4618,
 a-4623, a-4862, a-5273
`\Code@DefIndexStar`,
 a-4618, a-4630, a-5277
`\Code@DefMacro`, a-4854,
 a-4861
`\Code@Delim`, a-2186,
 a-2188, a-2193
`\code@delim`, a-2190,
 a-2376, a-2398,
 a-2403, a-2468,
 a-2477, a-2855,
 a-2879, a-2960,
 a-3433, a-5110,

File Key: a=gmdoc.sty, b=gmdocc.cls, c=gmutils.sty, d=gmiflink.sty, e=gmverb.sty,
 f=gmeometric.sty, g=gmoldcomm.sty

a-6933, a-6937,
 a-6938, a-7204
`\Code@Delim@St`, a-2186,
 a-2193
`\code@escape@char`,
 a-2891, a-3496
`\Code@MarginizeEnvir`,
 a-4794, a-4797
`\Code@MarginizeMacro`,
 a-3699, a-4784,
 a-4787, a-4863, a-4868
`\Code@UsgEnvir`, a-4858,
 a-4945
`\Code@UsgIndex`, a-4659,
 a-4662, a-4867, a-4976
`\Code@UsgIndexStar`,
 a-4659, a-4667
`\Code@UsgMacro`, a-4858,
 a-4866
`\CodeCommonIndex`,
 a-4674, a-7477
`\CodeCommonIndex*`, 15
`\CodeDelim`, 19, a-2186,
 a-2398, a-5111,
 a-6934, a-7140
`\CodeDelim*`, a-2202, a-7142
`\CodeEscapeChar`, 19,
 a-3493, a-3503,
 a-5063, a-5074, a-7346
`\CodeIndent`, 18, a-2227,
 a-2230, a-2741,
 a-3060, a-3426,
 a-7340, b-315, 102
`\codeline@glossary`,
 a-4880, a-4907
`\codeline@wrindex`,
 a-4873, a-4906,
 a-4915, a-4920
`\CodelineIndex`, a-7407,
 a-7408
`codelinenum`, 19, a-3111,
 a-3115
`\CodelineNumbered`,
 a-7394, a-7395, 103
`\CodeMarginize`, 15, a-4773
`\CodeSpacesBlank`, 11,
 a-2094, a-2578
`codespacesblank`, 11, a-2092
`\CodeSpacesGrey`, 11,
 a-2102, a-2590
`codespacesgrey`, 11, a-2097
`\CodeSpacesSmall`, a-2583
`\CodeSpacesVisible`,
 a-2569, a-2593, a-2600

`\CodeTopsep`, 18, a-2245,
 a-2264, a-2307,
 a-2313, a-2322,
 a-3099, a-3155,
 a-5061, a-5063,
 a-5072, a-5074,
 a-5203, a-7342
`\CodeUsage`, 14, a-4856
`\CodeUsgIndex`, 15, a-4657
`\color`, c-2947, c-2948
`\columnsep`, a-5623, f-383
`\CommonEntryCmd`, 19,
 a-4477, a-4600
`\continue@macroscan`,
 a-3580, a-3600
`\continuum`, c-2981
`\copy`, c-2263, c-2293,
 c-2307, c-2724, c-2736
`copyrnote`, 22, a-6990
`\count`, a-5947, a-5951,
 a-5952, c-2268,
 c-2269, c-2270,
 c-2271, c-2272,
 c-2273, c-2274,
 c-2275, c-2918,
 c-2920, c-2921,
 c-2922, c-2925,
 c-2934, c-2935,
 c-2936, c-2937
`countalllines`, 10, a-2027
`countalllines*`, 10, a-2031
`\CS`, 23, a-7123, a-7129, a-7131
`\cs`, 21, a-7092, a-7095,
 c-1321, c-1327,
 c-1331, c-1346
`\CSes`, a-7131
`\CSS`, a-7129
`\cup`, c-2733
`\currentfile`, a-6355,
 a-6356, a-6357,
 a-6358, a-6360,
 a-6362, a-6364,
 a-6366, a-6372,
 a-6411, a-6418,
 a-6443, a-6555,
 a-6556, a-6558, a-6617
`\czas`, c-3408
`\czer`, c-2949, c-2952, c-2953
`\czerwo`, c-2948, c-2949

`\dag`, c-2953
`\daleth`, b-344
`\date`, a-1889, a-6713
`\date@line`, c-3592,
 c-3605, c-3608

`\dateskip`, c-3591, c-3597
`\day`, a-6236, a-6240
`\deadcycles`, c-3152
`debug`, b-188, 109
`\debug@special`, a-2906
`\Declare@Dfng`, a-3802,
 a-3803, a-3808
`\Declare@Dfng@inner`,
 a-3810, a-3813, a-3820
`\DeclareBoolOption`,
 a-4325, a-4336
`\DeclareComplementaryOption`,
 a-4326, a-4337
`\DeclareDefining`, 12,
 a-3799, a-4223,
 a-4224, a-4225,
 a-4226, a-4254,
 a-4255, a-4256,
 a-4257, a-4258,
 a-4259, a-4260,
 a-4261, a-4262,
 a-4263, a-4267,
 a-4268, a-4269,
 a-4270, a-4271,
 a-4272, a-4285,
 a-4286, a-4289,
 a-4291, a-4324,
 a-4325, a-4326, a-4327
`\DeclareDefining*`,
 a-4274, a-4275,
 a-4276, a-4282
`\DeclareDocumentCommand`,
 a-4263, c-419, c-440,
 c-593, c-668
`\DeclareDocumentEnvironment`,
 c-439, c-592
`\DeclareDOXHead`, 13, a-4300
`\DeclareKVOFam`, 13, a-4331
`\DeclareLogo`, c-2234,
 c-2253, c-2279,
 c-2290, c-2320,
 c-2326, c-2329,
 c-2331, c-2334,
 c-2337, c-2341,
 c-2343, c-2346, c-2353
`\DeclareOption`, a-2012,
 a-2019, a-2027,
 a-2031, a-2041,
 a-2046, a-2055,
 a-2080, a-2082,
 a-2092, a-2097, a-4276
`\DeclareOptionX`, a-4291,
 a-4303, a-4310,
 a-4315, b-144
`\DeclareOptionX*`, b-270

File Key: a=gmdoc.sty, b=gmdocc.cls, c=gmutils.sty, d=gmiflink.sty, e=gmverb.sty,
 f=gmeometric.sty, g=gmoldcomm.sty

\DeclareRobustCommand, a-4270
\DeclareRobustCommand*, c-1112, c-1113, c-1114, c-1115, c-1321, d-160, d-177, d-186
\DeclareStringOption, a-4324, a-4335
\DeclareTextCommand, a-4271, c-2247
\DeclareTextCommandDefault, a-4272, c-2249
\DeclareVoidOption, a-4327, a-4338
\defaultfontfeatures, b-245
\DefaultIndexExclusions, 16, a-5381, a-5517, a-5545
\DefEntry, 19, a-4597, a-7519
\DefIndex, 15, a-4616, a-7469
\Define, 14, a-4848
\define@boolkey, a-3862, a-4286
\define@choicekey, a-3919, a-4289
\define@key, a-3871, a-3886, a-3907, a-4285
\definecolor, a-2118
\defobeylines, c-2803
\dekbigsip, c-2815
\dekclubs, 11, e-668, 174
\dekclubs*, b-450, 174
\dekfracc, c-2504
\dekfracc@args, c-2482, c-2493, c-2507, c-3071
\dekfraccsimple, c-3070, c-3076
\dekfracsplash, c-3063, c-3073, c-3074
\dekmedskip, c-2814
\deksmallskip, c-2812
\DeleteShortVerb, 11, e-412, 174
\Delta, c-2568
\Describe, 14, a-7297
\Describe@Env, a-7286, a-7292, a-7300, a-7312
\Describe@Macro, a-7286, a-7300, a-7305
\DescribeEnv, a-7291, 101
\DescribeMacro, a-7284, 101
\dimen, c-2882, c-2883, c-2884, c-3265, c-3266
\dimexpr, c-2592, c-2597, c-2603, c-2658, c-2725, c-3673, c-3674
\DisableCrossrefs, a-7416, a-7419
\discre, a-7116, c-1264, c-1273, c-1290
\discret, c-1266, c-1271
\divide, c-2270, c-2273, c-2274, c-3217, c-3218, c-3223, c-3804
\division, 21, a-6569, a-7150
\Do@Index, a-5524, a-5531
\do@noligs, e-353, e-659
\do@properindex, a-4698, a-4753, a-5036
\dobreakblankspace, e-338
\dobreakbslash, e-319, e-355
\dobreaklbrace, e-284, e-355
\dobreakspace, e-356, e-372
\dobreakvisiblespace, e-336, e-372
\Doc@Include, a-6318, a-6321
\Doc@Input, a-2352, a-2356, a-7597
\DocInclude, 8, 10, 24, a-1911, a-1918, a-1919, a-1920, a-1921, a-1922, a-6318, a-6368, a-6374, a-6590
\DocInput, 8, a-2352, a-6615, a-6642, a-6938
\DocInputsCount, a-3114
\docstrips@percent, a-5116
\DocstyleParms, a-7534
\document, c-1659
\documentclass, a-1882
\DoIndex, 16, a-1906, a-5524, a-5543
\DoNot@Index, a-5330, a-5338
\DoNotIndex, 15, a-5330, a-5541, a-5543, a-5547, b-466
\dont@index, a-5342, a-5345, a-5353, a-5363, a-5531
\Don'tCheckModules, a-7539
\doprivateothers, a-3437, a-3438, a-3515, a-3519
\dp, c-2639, c-2655, c-2658, c-2725
\ds, 22, a-7120
\dywiz, c-3400
\acute{e}, b-336, b-355
\edverbs, b-454, e-677, e-682, 174
\eequals, c-2895
\eg@MakeShortVerb, e-722, e-726
\eg@MakeShortVerbStar, e-722, e-725
\egCode@MarginizeEnvir, a-4776, a-4793
\egCode@MarginizeMacro, a-4777, a-4783
\egfirstofone, c-241, c-243
\egRestore@Macro, c-1486, c-1488
\egRestore@MacroSt, c-1486, c-1489
\egStore@Macro, c-1406, c-1411
\egStore@MacroSt, c-1406, c-1412
\egText@Marginize, a-4993, a-4996
\em, c-3767, c-3777
\emdash, c-3366
\emptify, a-2497, a-2671, a-2690, a-3984, a-4381, a-5107, a-7211, a-7212, c-384, c-384, c-2438, c-2699, c-2703, c-3790
\EnableCrossrefs, a-6479, a-7418
\encapchar, 20, a-3480, a-3625, a-4499, a-4889, a-5845
\endenumargs, a-7226
\endenumerate, a-7219
\endenvironment, a-5315
\endlinechar, a-6897
\endlist, c-2203, c-2220
\endmacro, a-5211, a-5226
\endmacro*, a-5226
\endmacrocode, a-5094
\endoldmc, a-5094
\endskiplines, 25
\endtheglossary, a-6481
\endverbatim, e-493
\englishdate, c-3482
\enoughpage, c-2880
\enspace, b-436

File Key: a=gmdoc.sty, b=gmoccc.cls, c=gmutils.sty, d=gmiflink.sty, e=gmverb.sty, f=gmeometric.sty, g=gmoldcomm.sty

\ensuremath, b-464,
c-1225, c-1238,
c-2338, c-2549,
c-2982, c-3418, c-3419
\EntryPrefix, 19, a-4490,
a-4492, a-4531,
a-4883, a-4885,
a-5637, a-6329
\enumargs, a-7226
enumargs, 23, a-7191
enumargs*, 23, a-7223
\enumerate, a-7199
enumerate*, c-2188
\env, 21, a-7204, c-1327
\environment, a-5314
environment, 15, a-5314
\envirs@toindex, a-4654,
a-4810, a-4813,
a-4814, a-4841, a-4960
\envirs@tomarginpar,
a-4804, a-4807,
a-4808, a-4840, a-4955
\EOF, a-7730
\EOFMark, 18, a-2396,
a-2556, a-6937,
a-7726, b-464, 109
>equals, c-2893
\errorcontextlines, b-381
\TeX, 22, c-2337, c-2341
\evensidemargin, a-6286,
f-376
\everydisplay, c-2689,
c-2742
\everyeof, 18, a-2410
\everymath, c-2563,
c-2689, c-2692,
c-2720, c-2742
\everypar, a-2388, a-2388,
a-2447, a-2607,
a-2608, a-2628,
a-2715, a-2982,
a-3003, a-3046,
a-3065, a-3242,
a-3250, a-5299,
a-6991, c-1933,
c-1943, c-3287, e-567
\ExecuteOptionsX, b-145
\exhyphenpenalty,
b-458, e-480, e-484
\exists, c-2604
\f@encoding, c-2932
\f@series, c-2280
\fakern, c-2684
\fakesc@extrascale,
c-3221, c-3236
\fakescaps, c-3196
\fakescapscore, c-3174,
c-3198
\fakescextrascale, c-3236
\file, 21, c-1295
\filedate, 22, a-6560,
a-6826, a-6915
\filediv, a-6504, a-6521,
a-6568, a-6586,
a-6738, a-6797
\filedivname, a-6505,
a-6515, a-6518,
a-6522, a-6535,
a-6537, a-6566,
a-6585, a-6739
\FileInfo, 23, a-6841
\fileinfo, 22, a-6828
\filekey, a-6443, a-6540,
a-6543
\filename, a-6559, a-6824
\filenote, 23, a-6915, a-6917
\filesep, a-6328, a-6329,
a-6484, a-6539
\fileversion, 22, a-6235,
a-6239, a-6561,
a-6827, a-6915
\Finale, 20, a-7428, a-7457
\finish@macroscan,
a-3558, a-3572,
a-3586, a-3597,
a-3692, g-36, g-37, g-64
\Finv, b-344
\fixbslash, e-307, 173
\fixlbrace, e-313, 173
\fontencoding, e-368
\fontseries, b-394
\fontspec, b-407
fontspec, b-259
\fooatletter, c-245
\foone, a-2520, a-2706,
a-3268, a-3302,
a-3340, a-3395,
a-3791, a-4002,
a-4092, a-4135,
a-5126, a-5140,
a-5720, a-5764,
a-6853, a-6881,
a-6928, a-6977,
a-7253, a-7724,
c-241, c-245, c-887,
c-1127, c-1130,
c-1138, c-1154,
c-1174, c-1178, c-1181,
c-1330, c-2621,
c-2794, c-2802,
c-3380, c-3402,
e-282, e-296, e-316,
e-326, e-333, g-51, g-82
\footins, f-384
\footskip, f-380
\forall, c-2601
\FormatHangHeading,
c-2039, c-2046,
c-2054, c-2061
\FormatRunInHeading,
c-2068, c-2073
\freeze@actives, c-2917
\fullcurrentfile,
a-6356, a-6372, a-6420
\fullpar, c-3656
\g@emptyify, a-2551,
a-3611, a-4808,
a-4814, a-6120,
a-6679, a-6680,
a-6800, a-6960,
c-388, c-389
\g@relaxen, a-3752,
a-4518, a-4520,
a-4529, a-5292,
a-6799, c-397, c-398
\gaddtomacro, 20, a-2551,
a-3426, a-3834, c-358
\gag@index, a-2151,
a-4913, a-7394, a-7416
\Game, b-344
\garamath, c-2719, 149
\gemptyify, c-389, c-389
\GeneralName, a-5890,
a-5891, a-5920,
a-5974, a-6334
\generalname, a-5851,
a-5857, a-5907, a-6016
\geometry, b-371
\GetFileInfo, 22, a-6418,
a-6558, a-6823
\gimel, b-344
\glet, a-2417, a-2777,
a-3610, a-4638,
a-4788, a-5110,
a-5721, a-5765,
a-6545, a-6550,
a-6564, c-350, c-1957
\glossary@prologue,
a-5972, a-6052,
a-6090, a-6097, a-6547
\glossaryentry, a-4889
\GlossaryMin, 16, a-6041,
a-6041, a-6052

File Key: a=gmdoc.sty, b=gmdocc.cls, c=gmutils.sty, d=gmiflink.sty, e=gmverb.sty,
f=gmeometric.sty, g=gmoldcomm.sty

\GlossaryParms, 17,
a-6053, a-6104

\GlossaryPrologue, 17,
a-6089
a-6089

\glueexpr, a-7201, c-3629,
c-3672

\gluestretch, c-3630

\gm@atppron, c-1691,
c-1695, c-1696,
c-1697, c-1698,
c-1700, c-1701,
c-1702, c-1703

\Gm@bindingoffset,
f-181, f-363

\Gm@bmargin, f-351

\Gm@checkboxl, f-345,
f-362, f-365, f-366, f-367

\gm@clearpagesduetoopenright
c-1737, c-1800

\Gm@cnth, f-181, f-350, f-353

\Gm@cntv, f-181, f-352, f-355

\Gm@dimlist, f-182

\gm@dontnumbersectionsoutofmainmatter
c-1735, c-1784

\gm@DOX, b-144, b-157,
b-164, b-167, b-171,
b-174, b-182, b-188,
b-193, b-200, b-204,
b-225, b-234, b-253,
b-254, b-259

\Gm@driver, f-368

\gm@duppa, c-2483, c-2484,
c-2486

\gm@EOX, b-145, b-263, b-266

\Gm@even@mp, f-182

\gm@gobmacro, c-2382, c-2388

\Gm@height, f-351

\Gm@hmarginratio, f-354

\gm@hyperrefstepcounter,
c-1727, c-1730, c-1813

\gm@hypertarget, d-161,
d-164

\gm@iflink, d-186, d-187,
d-189

\gm@ifnac, c-848, c-850

\gm@ifref, d-177, d-178,
d-180

\gm@ifundefined, c-933,
c-1751, c-1784,
c-1800, c-1876,
c-1897, c-1952,
c-1965, c-2035,
c-2162, c-2316,
c-2345, c-2347,
c-2352, c-2354,

c-2484, c-2628,
c-3220, c-3243

\gm@bracehook, a-4114,
e-286, e-291

\gm@letspace, c-867, c-875

\Gm@lines, f-357

\Gm@lmargin, f-349

\gm@notprerr, c-1624, c-1631

\Gm@odd@mp, f-182

\Gm@orgh, f-182

\Gm@orgw, f-182

\Gm@paper, f-344

gm@PronounGender, c-1689

\gm@pswords, c-1279,
c-1281, c-1283

\Gm@rmargin, f-349

\gm@sec, c-2155, c-2166, c-2167

\gm@secini, c-2120,
c-2139, c-2145,
c-2151, c-2164

\gm@secmarkh, c-2147

\gm@secstar, c-2123,

\gm@secstar, c-2133, c-2140,
c-2152, c-2166, c-2167

\gm@secx, c-2155, c-2156

\gm@secxx, c-2122, c-2150,
c-2157

\Gm@showparams, f-336

\gm@straightensec,
c-2161, c-2170

\gm@targetheading,
c-1728, c-1731

\Gm@tmargin, f-351

\Gm@true, f-346, f-347,
f-348, f-364

\Gm@truedimen, f-364

\gm@verb@eol, a-3451,
a-7074, e-600, e-625,
e-647

\Gm@vmarginratio, f-356

\Gm@wd@mp, f-181

\Gm@width, f-349

\gm@xistar, a-5127, a-5130

\gma, c-2712

\gma@arrowdash, c-2723,
c-2732, c-2738, c-2739

\gma@bare, c-2708, c-2710

\gma@bracket, c-2710, c-2711

\gma@checkbracket,
c-2709, c-2713

\gma@dollar, c-2707,
c-2708, c-2713

\gma@gmathhook, c-2687,
c-2703, c-2704, c-2727

\gma@quantifierhook,
c-2601, c-2604,
c-2699, c-2701,
c-2734, c-2740

\gma@tempa, c-2590,
c-2592, c-2595,
c-2597, c-2654,
c-2658, c-2679, c-2682

\gma@tempb, c-2655, c-2658

\gmath, c-2562, c-2707,
c-2711, 149

\gmathhook, c-2704

\gmbboxedspace, a-7043,
a-7046, a-7056,
a-7066, a-7068,
a-7072, a-7085,
a-7094, a-7116

\gmcc@article, b-174

\gmcc@CLASS, b-159, b-161,
b-288, b-296

\gmcc@class, b-157, b-164,
b-167, b-171, b-174

\gmcc@debug, b-188

\gmcc@fontspec, b-259

\gmcc@gmeometric, b-204

\gmcc@minion, b-253

\gmcc@mptt, b-234

\gmcc@mwart, b-164

\gmcc@mwbk, b-171

\gmcc@mwcclsfalse, b-288

\gmcc@mwcclstrue, b-161

\gmcc@mwrrep, b-167

\gmcc@nochanges, b-200

\gmcc@noindex, b-193

\gmcc@oldfontsfalse,
b-225, b-241

\gmcc@oldfontstrue,
b-224, b-312

\gmcc@outeroff, b-182

\gmcc@PAGELLA, b-255

\gmcc@pagella, b-254

\gmcc@resa, b-160, b-161

\gmcc@setfont, b-240,
b-253, b-254

\gmcc@sysfonts, b-225

\gmd@toc, a-2502, a-2504,
a-2505

\gmd@ABIOnce, a-2535,
a-2536, a-2551

\gmd@adef@altindex,
a-4180, a-4190,
a-4191, a-4193,
a-4194, a-4197, a-4199

File Key: a=gmdoc.sty, b=gmdocc.cls, c=gmutils.sty, d=gmiflink.sty, e=gmverb.sty,
f=gmeometric.sty, g=gmoldcomm.sty

\gmd@adef@checkD0Xopts,
 a-4031, a-4047
 \gmd@adef@checklbracket,
 a-4016, a-4037
 \gmd@adef@cs, a-3992
 \gmd@adef@cshookfalse,
 a-3702
 \gmd@adef@cshooktrue,
 a-3992
 \gmd@adef@currdef,
 a-3825, a-3831,
 a-3835, a-3839,
 a-3840, a-3842,
 a-3844, a-3847,
 a-3850, a-3873,
 a-3890, a-3896,
 a-3909, a-3911,
 a-3978, a-4059,
 a-4064, a-4163,
 a-4169, a-4181,
 a-4184, a-4192, a-4195
 \gmd@adef@defaulttype,
 a-3802, a-3803, a-3822
 \gmd@adef@deftext,
 a-4156, a-4176
 \gmd@adef@dfKVpref,
 a-4013, a-4026,
 a-4054, a-4058
 \gmd@adef@dk, a-4008
 \gmd@adef@dofam, a-4029,
 a-4088, a-4096,
 a-4146, a-4162
 \gmd@adef@dox, a-4019
 \gmd@adef@fam, a-4028,
 a-4086, a-4089,
 a-4094, a-4097,
 a-4144, a-4147,
 a-4170, a-4171
 \gmd@adef@indextext,
 a-4179, a-4200, a-4203
 \gmd@adef@KVfam, a-3907
 \gmd@adef@KVpref, a-3886
 \gmd@adef@prefix, a-3871
 \gmd@adef@scanDKfam,
 a-4123, a-4143
 \gmd@adef@scanD0Xfam,
 a-4021, a-4049, a-4072
 \gmd@adef@scanfamact,
 a-4077, a-4093
 \gmd@adef@scanfamoth,
 a-4074, a-4085
 \gmd@adef@scanKVpref,
 a-4009, a-4020,
 a-4039, a-4048, a-4053
 \gmd@adef@scancode,
 a-4119, a-4127, a-4151
 \gmd@adef@selfrestore,
 a-4238, a-4418, a-4421
 \gmd@adef@setkeysdefault,
 a-3828, a-3855
 \gmd@adef@setKV, a-3891,
 a-3915, a-3973, a-3976
 \gmd@adef@settype,
 a-3943, a-3945,
 a-3947, a-3949,
 a-3951, a-3953,
 a-3955, a-3957,
 a-3959, a-3961, a-3969
 \gmd@adef@text, a-4000
 \gmd@adef@TYPE, a-3846,
 a-3970
 \gmd@adef@type, a-3919
 \gmd@adef@typenr,
 a-3920, a-3942
 \gmd@adef@typevals, a-3920
 \gmd@auxext, a-6346,
 a-6348, a-6388, a-6397
 \gmd@bslashEOL, a-3348,
 a-3397, a-3400
 \gmd@charbychar, a-2737,
 a-2793, a-2874,
 a-2929, a-3727,
 a-3992, a-4000,
 a-4039, a-4049,
 a-4055, a-4090,
 a-4098, a-4148,
 a-4158, a-4434
 \gmd@checkifEOL, a-2609,
 a-2980
 \gmd@checkifEOLmixd,
 a-2885, a-3001
 \gmd@chschangeline,
 a-6201, a-6209,
 a-6218, a-6233
 \gmd@closingspacewd,
 a-2722, a-3376,
 a-3386, a-3388
 \gmd@codecheckifds, a-5186
 \gmd@codeskip, a-2748,
 a-3037, a-3154,
 a-3181, a-3209
 \gmd@continuenarration,
 a-2479, a-2604, a-2857
 \gmd@countnarrline@,
 a-2626, a-2666
 \gmd@counttheline,
 a-2897, a-2929, a-2936
 \gmd@cpnarrline, a-2606,
 a-2664, a-2671,
 a-2686, a-2981,
 a-3002, a-3431
 \gmd@ctallsetup, a-2674,
 a-2690, a-5109, a-6843
 \gmd@currentlabel@before,
 a-2361, a-2417
 \gmd@currenvxistar,
 a-5123, a-5129
 \gmd@DefineChanges,
 a-5828, a-6028
 \gmd@detectors, a-3728,
 a-3833, a-3834,
 a-3984, a-4378,
 a-4381, a-4389, a-4519
 \gmd@difilename, a-6342,
 a-6345
 \gmd@dip@hook, a-5584,
 a-5591, a-5592
 \gmd@docincludeaux,
 a-6354, a-6498, a-6500
 \gmd@docstripdirective,
 a-2974, a-2995,
 a-5190, a-5723
 \gmd@docstripinner,
 a-5731, a-5733
 \gmd@docstripshook, a-5773
 \gmd@docstripverb,
 a-5730, a-5768
 \gmd@doindexingtext,
 a-4207, a-4812, a-4817
 \gmd@doIndexRelated,
 a-6410, a-6427, a-6478
 \gmd@dolspaces, a-2480,
 a-2737, a-2824
 \gmd@DoTeXCodeSpace,
 a-2465, a-2570,
 a-2579, a-2584, a-5112
 \gmd@ea@bwrap, a-7205,
 a-7212, a-7215, a-7225
 \gmd@ea@ewrap, a-7208,
 a-7211, a-7215, a-7225
 \gmd@ea@wraps, a-7210,
 a-7213, a-7217, a-7224
 \gmd@eatlspace, a-2829,
 a-2848, a-2853
 \gmd@endpe, a-3010,
 a-3015, a-3043,
 a-3050, a-3057
 \gmd@EOLorcharbychar,
 a-2901, a-2916
 \gmd@evpaddonce, a-5282,
 a-5288

File Key: a=gmdoc.sty, b=gmoccc.cls, c=gmutils.sty, d=gmiflink.sty, e=gmverb.sty,
 f=gmeometric.sty, g=gmoldcomm.sty

\gmd@fileinfo, a-6850,
 a-6862
 \gmd@finishifstar,
 a-3558, a-3586, a-3596
 \gmd@FIrescan, a-6867,
 a-6882
 \gmd@glossary, a-4903,
 a-4907, a-5889
 \gmd@glossCStest,
 a-5884, a-5888,
 a-5904, a-5913
 \gmd@gobbleuntilM,
 a-3306, a-3307
 \gmd@grefstep, a-2627,
 a-2636, a-2682,
 a-2687, a-2759,
 a-2941, a-2951
 \gmd@guardedinput,
 a-2390, a-2411
 \gmd@iedir, a-5339,
 a-5358, a-5531
 \gmd@ifinmeaning,
 a-3609, a-3627,
 a-3830, g-38
 \gmd@ifonetoken, a-5209,
 a-5224, a-5259, a-7286
 \gmd@ifsingle, a-5244,
 a-5262
 \gmd@iihook, a-2395,
 a-2497, a-6935
 \gmd@in@@, a-3631, a-3631,
 a-3635
 \gmd@inputname, a-2359,
 a-4028, a-6198,
 a-6208, a-6215
 \gmd@inverb, a-7041,
 a-7044, a-7060
 \gmd@jobname, a-6341, a-6345
 \gmd@justadot, a-4635,
 a-4638, a-4651,
 a-4788, a-5339
 \gmd@KVprefdefault,
 a-3878, a-3886,
 a-3888, a-4013,
 a-4026, a-4300
 \gmd@lbracecase, a-4000,
 a-4012, a-4025,
 a-4115, a-4118,
 a-4122, a-4126, a-4130
 \gmd@ldspaceswd, a-2757,
 a-2767, a-2768,
 a-2781, a-2813,
 a-2828, a-2850, a-2856
 \gmd@maybequote, a-3549,
 a-3570, a-3582,
 a-3610, a-3611, a-4713
 \gmd@mc, a-7162
 \gmd@mcdiag, a-7166,
 a-7181, a-7183, a-7187
 \gmd@mchook, a-7165
 \gmd@modulehashone,
 a-5735, a-5739,
 a-5770, a-5774
 \gmd@narrcheckifds,
 a-2990, a-2993
 \gmd@narrcheckifds@ne,
 a-2966, a-2972
 \gmd@nlperc, a-7048,
 a-7061, a-7083,
 a-7086, a-7092, a-7095
 \gmd@nocodeskip, a-2743,
 a-2750, a-3039,
 a-3041, a-3175,
 a-3183, a-3203, a-3211
 \gmd@oldmcfinis, a-5154
 \gmd@concenum, a-5289,
 a-5291, a-5293,
 a-5298, a-5300, a-5303
 \gmd@parfixclosingspace,
 a-2708, a-3375
 \gmd@percenthack,
 a-2882, a-2959
 \gmd@preambleABD, e-738,
 e-739, e-746
 \gmd@preverypar, a-2210,
 a-2629, a-2985,
 a-3003, a-3047,
 a-3066, a-3240,
 a-3248, a-3250
 \gmd@providefii, a-6899,
 a-6904
 \gmd@quotationname,
 a-6998, a-7006, a-7010
 \gmd@resa, a-3821, a-3823,
 a-3872, a-3875,
 a-3887, a-3888,
 a-3894, a-3895,
 a-3897, a-3900,
 a-3908, a-3910,
 a-3912, a-3914,
 a-3977, a-3980,
 a-4825, a-4828, a-4830
 \gmd@resetlinecount,
 a-2378, a-3108, a-3121
 \gmd@ResumeDfng, a-4452,
 a-4454
 \gmd@revprefix, a-4564,
 a-4566
 \gmd@setChDate, a-5940,
 a-5943, a-5962
 \gmd@setclosingspacewd,
 a-3387
 \gmd@setclubpenalty,
 a-2366, a-2456,
 a-2460, a-2484
 \gmd@setilrr, a-2837,
 a-3028, a-3086, a-3377
 \gmd@skipgmltext,
 a-6959, a-6960, a-6970
 \gmd@skiplines, a-2698,
 a-2701
 \gmd@spacewd, a-2810,
 a-2827, a-2850
 \gmd@texcodeEOL, a-2740,
 a-2918
 \gmd@texcodespace,
 a-2580, a-2586,
 a-2736, a-2821,
 a-2825, a-2849, a-3722
 \gmd@textEOL, a-2433,
 a-2522, a-2986,
 a-3008, a-3343,
 a-5107, a-5739, a-5774
 \gmd@typesettexcode,
 a-2707, a-2843, a-2859
 \gmd@writeckpt, a-6431,
 a-6471
 \gmd@writeFI, a-6866, a-6875
 \gmd@writemauxinpaux,
 a-6388, a-6449
 \gmdindexpagecs, a-4557,
 a-4563
 \gmdindexrefcs, a-4554,
 a-4557, a-4561
 \gmdmarginpar, 15, 22,
 a-5009, a-5015, a-5019
 \gmdnoindent, 23, a-7018
 \gmddocMargins, b-370,
 b-375
 \gmdocIncludes, 9, a-6641
 \gme@tobestored, f-180,
 f-185, f-397
 \gmeometric, b-204
 \mglo.list, 84
 \GMlabel, d-149
 \gmhypertarget, a-3142,
 d-160, 170
 \gmiflink, a-4561, d-186, 170
 \gmifref, d-177, 170

File Key: a=gmdoc.sty, b=gmddoc.cls, c=gmutils.sty, d=gmiflink.sty, e=gmverb.sty,
 f=gmeometric.sty, g=gmoldcomm.sty

\gml@StoreCS, c-1451,
c-1474, c-1511
\gml@storemacros,
c-1452, c-1463,
c-1472, c-1477, c-1514
\gmlonely, 22, a-6955, a-6967
\gmobeyspaces, a-2579,
c-2795, e-482, e-601
\gmoc@checkenv, g-40, g-54
\gmoc@checkenvinn,
g-56, g-58
\gmoc@defbslash, g-34, g-86
\gmoc@maccname, g-47, g-60
\gmoc@notprinted, g-38,
g-45
\gmoc@ocname, g-48, g-69
\gmoc@resa, g-59, g-60, g-69
\gmshowlists, c-737
\GMtextsuperscript, c-2539
\gmu@acroinner, c-2993,
c-3003, c-3004, c-3012
\gmu@acrospace, c-2987,
c-2992, c-2992, c-2996
\gmu@checkaftersec,
c-1951, c-2010
\gmu@dashfalse, c-3544
\gmu@dashtrue, c-3546
\gmu@datecomma, c-3455,
c-3473, c-3501,
c-3521, c-3525, c-3528
\gmu@datef, c-3439,
c-3439, c-3483,
c-3483, c-3584, c-3611
\gmu@datefs1, c-3457,
c-3457, c-3503,
c-3503, c-3586
\gmu@dekfrac, c-2466,
c-2485, c-2489
\gmu@dekfraccsimple,
c-2489, c-3060, c-3071
\gmu@denominatorkern,
c-2468, c-2511, c-3063
\gmu@discretionaryslash,
c-1290, c-1301
\gmu@dywiz, c-3399, c-3403
\gmu@fileext, c-3113,
c-3123, c-3141
\gmu@filename, c-3112,
c-3126, c-3138,
c-3141, c-3144, c-3153
\gmu@getaddvs, c-2002,
c-2002, c-2008
\gmu@gettext, c-3111, c-3121
\gmu@gobdef, c-199, c-205
\gmu@ifnodash, c-3538,
c-3543
\gmu@luzniej, c-3270,
c-3273, c-3275
\gmu@nl@reserveda,
c-1583, c-1586,
c-1591, c-1594
\gmu@nocite@ampulex,
c-1649, c-1662
\gmu@numeratorkern,
c-2467, c-2510,
c-2511, c-3062
\gmu@prevsec, c-1932,
c-1934, c-1956,
c-1963, c-1993
\gmu@printslashes,
c-1295, c-1297,
c-1297, c-1299, c-1302
\gmu@resa, a-4062, a-4068,
a-4167, a-4173,
c-2962, c-2964
\gmu@reserveda, c-864,
c-866, c-872, c-874,
c-1031, c-1033,
c-1464, c-1467,
c-1470, c-1908,
c-1957, c-1958,
c-1961, c-2242,
c-2244, c-2246,
c-2247, c-2248,
c-3096, c-3097
\gmu@reservedb, c-1032,
c-1033
\gmu@restorespecials,
c-2649, c-3862
\gmu@RPfor, c-2947,
c-2959, c-2971, c-2982
\gmu@scalar, c-3208,
c-3212, c-3213, c-3219
\gmu@scalematchX,
c-3198, c-3210, c-3234
\gmu@scapLetters,
c-3170, c-3180, c-3185
\gmu@scapSpaces, c-3183,
c-3188, c-3192
\gmu@scapss, c-3191, c-3196
\gmu@scscale, c-3227, c-3233
\gmu@septify, c-2616, c-3864
\gmu@setheading, c-2007,
c-2013, c-2014
\gmu@setsetSMglobal,
c-1450, c-1455, c-1510
\gmu@setSMglobal,
c-1457, c-1459, c-1477
\gmu@SMdo@scope, c-1553,
c-1555, c-1558,
c-1559, c-1573
\gmu@SMdo@setscope,
c-1551, c-1557, c-1571
\gmu@SMglobalfalse,
c-1418, c-1432,
c-1459, c-1468,
c-1495, c-1504, c-1561
\gmu@SMglobaltrue,
c-1394, c-1457
\gmu@smtempa, c-1422,
c-1431, c-1498, c-1503
\gmu@storespecials,
c-2615, c-3859
\gmu@stripcomma, c-3529,
c-3533
\gmu@tempa, a-2700,
a-2702, a-3841,
a-3849, a-4490,
a-4492, a-4494,
a-4499, a-4500,
a-4567, a-4568,
a-4739, a-4742,
a-4745, a-4883,
a-4885, a-4887,
a-4889, a-4891,
a-4953, a-4955,
a-4959, a-4960,
a-5245, a-5246,
a-5265, a-5266,
a-5346, a-5349,
a-5351, a-5356,
a-5358, a-5889,
a-5912, a-5971,
a-5977, a-6196,
a-6206, a-6213,
a-6223, a-6224,
a-6393, a-6829,
a-6830, a-6958,
a-6965, a-6966,
a-7035, a-7042,
a-7052, a-7081,
a-7084, a-7090,
a-7093, a-7197,
a-7201, c-206, c-683,
c-689, c-700, c-708,
c-2387, c-2389,
c-2750, c-2753,
c-2755, c-3243,
c-3245, c-3246,
c-3247, c-3539,
c-3540, c-3738,
c-3742, c-3755,
c-3759, c-3815, c-3819

File Key: a=gmdoc.sty, b=gmoccc.cls, c=gmutils.sty, d=gmiflink.sty, e=gmverb.sty,
f=gmeometric.sty, g=gmoldcomm.sty

\gmu@tempb, a-5260,
a-5263, a-5265,
a-5863, a-5872,
a-5881, a-5901,
a-5903, a-5904,
a-5905, a-6392,
a-6393, a-6825,
a-6830, a-7036,
a-7043, a-7057,
a-7082, a-7085,
a-7091, a-7094,
a-7198, a-7201, c-684,
c-690, c-701, c-709
\gmu@tempc, c-685, c-715
\gmu@tempd, c-687, c-688,
c-691, c-694, c-699,
c-701, c-706, c-707,
c-719, c-725
\gmu@tempe, c-698, c-704,
c-723, c-726
\gmu@tempf, c-722, c-725
\gmu@testdash, c-3542,
c-3582
\gmu@thou@five, c-3786,
c-3789, c-3797, c-3797
\gmu@thou@i, c-3791,
c-3809, c-3817
\gmu@thou@ii, c-3791,
c-3809, c-3817
\gmu@thou@o, c-3791,
c-3809, c-3817
\gmu@thou@put, c-3790,
c-3794, c-3806
\gmu@thou@putter,
c-3793, c-3800,
c-3807, c-3814, c-3822
\gmu@thousep, c-3819, c-3826
\gmu@tilde, c-2766,
c-2771, c-2782
\gmu@whonly, c-3159, c-3160
\gmu@xedekfraccplain,
c-2443, c-2492
\gmu@xedekfraccstar,
c-2443, c-2458
\gmu@xefraccdef, c-2459,
c-2473, c-2474,
c-2475, c-2476,
c-2477, c-2478,
c-2479, c-2480, c-2481
\gmv@dismath, e-678,
e-681, e-685
\gmv@disverb, e-681, e-684
\gmv@edismath, e-679, e-687
\gmv@exhyphenpe, e-480,
e-484

\gmv@hyphenpe, e-479, e-483
\gmv@packname, e-431,
e-432, e-436
\gn@melet, a-4412, a-4413,
c-1590
\gobble, c-189, c-191, c-2680
\gobbletwo, c-192
\grab@default, c-509,
c-512, c-525
\grab@ms, c-555
\grefstepcounter,
a-2651, c-317, c-333
\grelaxen, a-5913, a-5922,
c-398, c-398, c-1934

\hathat, c-1331
\headheight, f-378
\HeadingNumber, c-1810,
c-1812
\HeadingNumberedfalse,
c-1736, c-1781
\HeadingRHeadText, c-1794
\HeadingText, c-1796
\HeadingT OCText, c-1795
\headsep, f-379
\HeShe, c-1700
\heshe, 7, c-1695
\hfillneg, c-2818
\hrefstepcounter,
a-2682, a-2687, c-332
\hidden@iffalse, c-295,
c-692
\hidden@iftrue, c-296, c-692
\Hide@Dfng, a-4408, a-4410
\Hide@DfngOnce, a-4408,
a-4417
\HideAllDefining, 13,
a-4376
\HideDef, 13, a-4243
\HideDef*, 13
\HideDefining, 13,
a-4245, a-4404
\HimHer, c-1702
\himher, c-1697
\HisHer, c-1701
\hisher, c-1696
\HisHers, c-1703
\hishers, c-1698
\HLPrefix, 19, a-3143,
a-4490, a-4492,
a-4570, a-4876,
a-4883, a-4885,
a-4890, a-5575, a-6328

\hoffset, f-385
\hrule, c-1981
\hunskip, c-340, c-2893,
c-2895, c-2897,
c-3657, c-3670
\Hybrid@DefEnvir,
a-5209, a-5276
\Hybrid@DefMacro,
a-5209, a-5272
hyperindex, 62
\hyperlabel@line,
a-2667, a-2763,
a-2942, a-2952, a-3137
\hypersetup, a-2131, a-5616
\hyphenpenalty, b-458,
c-1283, c-3311,
c-3636, e-479, e-483

\idiaeres, b-337, b-356
\if*, a-3597, a-5130
\if@aftercode, a-2742,
a-2836, a-2840,
a-3026, a-3032,
a-3075, a-3090,
a-3191, a-3204,
a-3377, a-7197,
a-7200, a-7204
\if@afterindent, c-1938
\if@afternarr, a-2745,
a-2836, a-2840,
a-3026, a-3031,
a-3196, a-3203
\if@codeskipput, a-2307,
a-2313, a-2322,
a-2727, a-2747,
a-3037, a-3167,
a-3202, a-5061,
a-5072, a-5203
\if@countalllines,
a-2024, a-2616
\if@debug, b-186, b-378,
b-384, b-385
\if@dsdir, a-2339, a-5189
\if@filesw, a-4873,
a-6388, a-6396,
a-6432, c-2870,
c-3125, c-3137, c-3145
\if@fshda, a-6756, a-6774,
a-6809
\if@gmccnochanges,
b-198, b-443
\if@gmu@mmhbox, c-2487,
c-2497, c-2501,
c-3067, c-3849
\if@ilgroup, a-2786,
a-3027, a-3033,

File Key: a=gmdoc.sty, b=gmdocc.cls, c=gmutils.sty, d=gmiflink.sty, e=gmverb.sty,
f=gmetric.sty, g=gmoldcomm.sty

a-3076, a-3084,
 a-3091, a-3379
`\if@indexallmacros,`
 a-2053, a-5516
`\if@linesnotnum,` a-2010,
 a-3135, a-4551
`\if@ltxDocInclude,`
 a-6411, a-6417,
 a-6421, a-6603
`\if@mainmatter`, c-1736
`\if@marginparsused,`
 a-2065, a-5001
`\if@mparswitch`, f-348, f-390
`\if@newline`, a-2331,
 a-2665, a-2759,
 a-2919, a-2938, a-2949
`\if@nobreak`, c-1935
`\if@noindex`, a-2039, a-2150
`\if@noskipsec`, e-535
`\if@nostanza`, a-2322, a-3170
`\if@openright`, c-1738
`\if@pageincliindex,`
 a-4489, a-4536, a-4882
`\if@pageindex`, a-2044,
 a-3138, a-4486,
 a-4553, a-4901,
 a-5568, a-5571,
 a-5572, a-5574
`\if@printalllinenos,`
 a-2025, a-2662, a-5037
`\if@RecentChange,`
 a-5854, a-5939
`\if@reversemargin`, f-391
`\If@SomethingTF`, c-599,
 c-601, c-604, c-606, c-618
`\if@specialpage`, c-1799
`\if@twoside`, c-1825,
 f-347, f-348, f-389
`\if@uresetlinecount,`
 a-2017, a-3107
`\IfBooleanF`, c-645
`\IfBooleanT`, c-641
`\IfBooleanTF`, c-633,
 c-642, c-646
`\ifcsname`, a-7167, c-939,
 c-1003
`\ifdate`, c-3579, c-3582,
 c-3591
`\ifdefined`, c-167, c-197,
 c-413, c-1133, c-1192,
 c-2407, c-2690,
 c-2972, c-3297, c-3379
`\ifdtraceoff`, b-385
`\ifdtraceon`, b-384
`\iffontchar`, c-2460, c-3418
`\ifGm@pass`, f-341
`\ifgmcc@mwcls`, b-154,
 b-287, b-291, b-310
`\ifgmcc@oldfonts`,
 b-223, b-323, b-390
`\ifgmd@adef@cshook,`
 a-3695, a-3990
`\ifgmd@adef@star,`
 a-3811, a-3862
`\ifgmd@glosscs`, a-3688
`\ifgmu@dash`, c-3536,
 c-3542, c-3548, c-3583
`\ifgmu@postsec`, c-1953,
 c-1992, c-2000
`\ifgmu@SMglobal`, c-1392,
 c-1416, c-1423,
 c-1456, c-1493,
 c-1499, c-1558
`\ifHeadingNumbered`,
 c-1780, c-1808
`\ifilrr`, a-2837, a-2841,
 a-3028, a-3072, a-3377
`\IfNoValueF`, c-627, c-629
`\IfNoValueT`, c-626, c-630
`\IfNoValueTF`, c-625, c-628
`\ifodd`, c-1693
`\ifprevhmode`, a-2866,
 a-2960, a-3058
`\ifSecondClass`, c-3046
`\IfSomethingF`, c-603, c-627
`\IfSomethingT`, c-600, c-626
`\IfSomethingTF`, c-599, c-625
`\IfValueF`, c-630
`\IfValueT`, c-629
`\IfValueTF`, c-628
`\ikern`, c-3084
`\ilju`, 21, a-3088
`\ilrr`, 21, a-3074
`\ilrr`, 21
`\ilrrfalse`, a-3092
`\ilrrtrue`, a-3081
`\im@firstpar`, a-3716,
 a-3718, a-3719,
 a-4695, a-4696, a-4699
`\IMO`, c-3022
`\in`, c-2740
`\incl@DocInput`, a-6420,
 a-6615, a-6638, a-6642
`\incl@filedivtitle`,
 a-6768, a-6797
`\incl@titletotoc`,
 a-6756, a-6769
`\inlasthook`, c-3142, c-3164
`\InclMaketitle`, a-6414,
 a-6750
`\includegraphics`, c-2537
`\incs`, 21, a-7088, a-7097
`\index@macro`, a-3719,
 a-4470, a-4699,
 a-4754, a-4835
`\index@prologue`, a-5559,
 a-5566, a-5611, a-6541
`\indexallmacros`, 10, a-2055
`\IndexColumns`, 20
`\indexcontrols`, a-3609,
 a-3624
`\indexdiv`, a-5562, a-5566,
 a-6097
`\indexentry`, a-4875
`\IndexInput`, 10, a-6930
`\IndexLinksBlack`, 20,
 a-5579, a-5612,
 a-5616, a-6053
`\IndexMin`, 20, a-5606,
 a-5606, a-5611
`\IndexParms`, 20, a-5613,
 a-5620, a-6104
`\IndexPrefix`, 19, a-4494,
 a-4542
`\IndexPrologue`, 20,
 a-5559, 104
`\inenv`, 21, a-7097
`\infty`, c-2579
`\inputlineno`, a-2643,
 a-2644, a-2681
`\interlinepenalty`, e-566
`\inverb`, 21, a-7032
`\itemindent`, c-2196, c-2214
`\itemize*`, c-2206
`\iteracro`, c-2986, c-2990
`\justified`, c-3645
`\kernel@ifnextchar`, a-6899
`\kind@fentry`, a-4477,
 a-4479, a-4483,
 a-4490, a-4492
`\Kvfam`, 13, a-3907
`\Kvpref`, 13, a-3886
`\labelsep`, c-2198, c-2216
`\labelwidth`, c-2197,
 c-2198, c-2215, c-2216
`\larger`, c-1112, c-2591,
 c-2596, c-2660,
 c-2661, c-2664,
 c-2665, c-2666,
 c-2667, 127

File Key: a=gmdoc.sty, b=gmocc.cls, c=gmutils.sty, d=gmiflink.sty, e=gmverb.sty,
 f=gmeometric.sty, g=gmoldcomm.sty

\largerr, c-1116, c-2662,
c-2663, 127
\last@defmark, a-4520,
a-4641, a-4646,
a-5859, a-5871,
a-5872, a-5873,
a-5919, a-5922
\LaTeXe, c-2230, c-2279
\LaTeXpar, 22, c-2290
\ldate, c-3606, c-3615
\leftarrow, c-2611, c-2738
\leftmargin, c-2195,
c-2213, e-548, e-549
\leftrightarrow, c-2613
\leftslanting, c-3729
\levelchar, 20, a-3481,
a-3625, a-5845,
a-5895, a-5909
\linebreak, c-3695
\linedate, c-3589, c-3605,
c-3609
\linedate@, c-3589,
c-3590, c-3591
\linedate@@, c-3589, c-3590
\LineNumFont, 19, a-2668,
a-3130, a-3133,
a-7362, 102
\lineskip, a-6700
linesnotnum, 10, a-2012
\list, c-2194, c-2212
\listparindent, c-2199,
c-2217
\lit, c-3748
\litshape, c-3730, c-3746,
c-3748, c-3770
\liturgiques, c-2946
\LoadClass, b-295, b-300
\longpauza, c-3369, c-3370
\looseness, c-3276, c-3287
\lpauza, c-3337
\lsl, c-3751
\ltxLookSetup, 9, a-6605,
a-6611
\ltxPageLayout, 9,
a-6270, a-6607
luzniej, c-3280
luzniej*, c-3285
\luzniejcore, c-3272, c-3280
macro, a-5201, a-5224, c-2388
macro, 15, a-5201
macro*, a-5224
\macro@iname, a-3549,
a-3567, a-3570,
a-3582, a-3719,
a-4699, a-4705,
a-4713, a-4754, a-4835
\macro@pname, a-3551,
a-3571, a-3583,
a-3697, a-3699,
a-3700, a-3709,
a-3719, a-3721,
a-3722, a-3723,
a-3845, a-4177,
a-4178, a-4199,
a-4204, a-4209,
a-4429, a-4689,
a-4690, a-4694,
a-4699, a-4735,
a-4736, a-4739,
a-4742, a-4754, g-38,
g-39
\macrocode, a-5093, g-67
macrocode, 8, 24, a-5071
macrocode*, a-5060
\MacrocodeTopsep, a-7342
\MacroFont, a-7332, 102
\MacroIndent, a-7340, 102
\MacroTopsep, a-2246,
a-2265, a-2306,
a-5202, a-5211, 102
\mag, f-387
\main, a-7519
\MakeGlossaryControls,
17, a-5832, a-5843
\MakePercentComment,
a-7551
\MakePercentIgnore,
a-5830, a-7550
\MakePrivateLetters,
14, 19, a-2467,
a-3507, a-3800,
a-4407, a-4451,
a-4617, a-4658,
a-4675, a-4722,
a-4757, a-4774,
a-4849, a-4857,
a-4969, a-4992,
a-5114, a-5205,
a-5330, a-5524,
a-5831, a-7285, a-7299
\MakePrivateOthers,
a-3515, a-4618,
a-4659, a-4676,
a-4723, a-4758,
a-4776, a-4854,
a-4858, a-4970,
a-4993, a-5205,
a-7292, a-7300
\MakeShortVerb, 11,
e-381, e-668, e-726, 174
\MakeShortVerb*, e-668,
e-725
\maketitle, 8, a-1901,
a-1906, a-5496,
a-6414, a-6662, 92
\MakeUppercase, c-3174
\mand, 23, a-7209
\mapsto, c-2731
\marg, c-1352, c-1376
\marginparpush, a-5003
\marginparsep, f-382
\marginpartt, 15, a-5019,
a-5026, b-394, b-407
\marginparwidth, a-5004,
a-6284, f-381
\mark@envir, a-2765,
a-2947, a-4803
maszynopis, c-3627
\math@arg, c-1372, c-1373
\mathbin, c-2580, c-2584,
c-2609, c-2610,
c-2643, c-2644,
c-2677, c-2678,
c-2733, c-2740
\mathchoice, c-2588,
c-2607, c-2622,
c-2669, c-2729
\mathclose, c-2641,
c-2642, c-2661,
c-2663, c-2665,
c-2667, c-2675
\mathfrak, c-2982
\mathindent, b-315
\mathit, c-2566
\mathop, c-2588
\mathopen, c-2627, c-2642,
c-2660, c-2662,
c-2664, c-2666, c-2674
\mathord, c-2646, c-2647,
c-2648
\mathpunct, c-2626
\mathrel, c-2581, c-2585,
c-2611, c-2612,
c-2613, c-2643,
c-2644, c-2645,
c-2681, c-2732,
c-2738, c-2739
\mathrm, c-2568, c-2574,
c-2580, c-2581,
c-2583, c-2584,
c-2585, c-2599
\Mathstrutbox@, c-2639
\maybe@marginpar,
a-3721, a-3741

File Key: a=gmdoc.sty, b=gmdocc.cls, c=gmutils.sty, d=gmiflink.sty, e=gmverb.sty,
f=gmeometric.sty, g=gmoldcomm.sty

\mcdiagOff, a-7187
 \mcdiagOn, a-7183
 \medmuskip, c-1271
 \meta, c-1217, c-1253,
 c-1352, c-1359,
 c-1366, 104
 \meta@font@select,
 c-1228, c-1247
 minion, b-253
 \mkern, c-2684
 \mod@math@codes, a-5778,
 a-5780, a-5782
 \Module, a-5736, a-5778
 \ModuleVerb, a-5771, a-5780
 \month, a-1889, a-6236, a-6240
 mppt, b-234
 \mpptversion, b-234
 \mskip, c-1271
 \multiply, a-5946, a-5951,
 c-2269, c-2272,
 c-3225, c-3275, c-3286
 \mw@getflags, c-1956
 \mw@HeadingBreakAfter,
 c-1801, c-1821,
 c-1836, c-1840,
 c-1871, c-1957
 \mw@HeadingBreakBefore,
 c-1798, c-1870, c-1958
 \mw@HeadingLevel,
 c-1778, c-1781
 \mw@HeadingRunIn,
 c-1816, c-1870
 \mw@HeadingType, c-1797,
 c-1932, c-1964,
 c-1965, c-1978
 \mw@HeadingWholeWidth,
 c-1819, c-1871
 \mw@normalheading,
 c-1823, c-1832,
 c-1835, c-1839, c-2013
 \mw@processflags, c-1872
 \mw@runinheading,
 c-1817, c-2014
 \mw@secdef, c-1877,
 c-1878, c-1879, c-1885
 \mw@section, c-1876
 \mw@sectionxx, c-1777
 \mw@secundef, c-1881,
 c-1893, c-1896
 \mw@setflags, c-1882
 mwart, b-164, 109
 mwbk, b-171, 109
 mwrep, a-1882, b-167, 109
 \n@melet, a-3846, a-3847,
 a-5096, a-5097,
 a-5873, c-1582,
 c-1907, c-1909,
 c-2124, c-2130,
 c-2164, c-2463, e-503
 \nacute, b-338, b-357
 \nameshow, c-741
 \nameshowthe, c-742
 napapierki, c-3263
 \napapierkicore, c-3260,
 c-3264
 \napapierkistretch,
 c-3258, c-3261
 \nawj, c-3289
 \nazwired, c-3424
 \neg, c-2580, c-2683
 \neq, c-2581, c-2678
 \neqb, c-2678
 NeuroOncer, a-5291
 \newbox, a-4259
 \newcount, a-4254, a-5303,
 a-5609, a-5936,
 a-6043, a-6157,
 c-3270, f-192
 \newcounter, a-3111,
 a-3114, a-3115,
 a-4282, a-6161,
 a-7162, a-7529,
 c-1689, c-1724, d-149
 \newdimen, a-4255, a-5606,
 a-6041
 \newgif, a-2331, a-2339,
 a-2866, a-3167,
 a-3170, a-3191,
 a-3196, c-259
 \newlength, a-2223,
 a-2227, a-2233,
 a-2810, a-2813,
 a-4262, e-573, e-578
 \newlinechar, a-6887
 \newread, a-4260
 \newskip, a-2245, a-2246,
 a-3386, a-4256, c-3702
 \newtoks, a-2210, a-4258,
 c-414, c-417
 \newwrite, a-4261, c-2870
 \next@ddc, c-509, c-512,
 c-520, c-524, c-529
 \nfss@text, c-1226
 \nieczer, c-2954
 \nlpercent, 21, a-7080
 \nobreakspace, c-3191
 nochanges, b-200, 108
 \nocite, c-1656
 \noeffect@info, a-7368,
 a-7386, a-7387,
 a-7388, a-7389,
 a-7534, a-7539,
 a-7540, a-7544
 \NoEOF, a-7729
 \nohy, c-3088
 \noindex, 10, a-2041, b-193, 108
 \nolimits, c-2603, c-2604
 \nolinkurl, c-3877
 nomarginpar, 11, a-2082
 NoNumSecs, c-1724
 \NonUniformSkips, 18,
 a-2295
 \nostanza, 18, a-2319
 \not@onlypreamble,
 c-1611, c-1615, c-1616,
 c-1617, c-1618, c-1619
 \NoValue, c-491, c-623,
 c-624, c-625, c-626, c-627
 \NoValueInIt, c-624
 \nu, c-2571
 \numexpr, a-2644, c-3409,
 c-3410, c-3805
 \oarg, c-1357
 \obeyspaces, a-2571,
 a-2585, a-5146, e-333,
 e-336, e-338, e-638
 \ocircum, b-339, b-360
 \oddsidemargin, a-6285,
 f-375
 \oe, b-364
 \old@MakeShortVerb,
 a-7592, e-699, e-721
 oldcomments, g-28
 \olddekclubs, e-669, 174
 \olddocIncludes, 9, 24,
 a-6637
 \OldDocInput, 8, 24,
 a-6638, a-7589
 \oldLaTeX, c-2229
 \oldLaTeXe, c-2230
 \OldMacrocodes, 24, a-7594
 \OldMakeShortVerb,
 e-669, e-720, 174
 \oldmc, a-5093, a-5101
 oldmc, 24, a-5093
 oldmc*, a-5096
 \oldmc@def, a-5151, a-5157
 \oldmc@end, a-5152, a-5158
 \omega, c-2578

File Key: a=gmdoc.sty, b=gmdocc.cls, c=gmutils.sty, d=gmiflink.sty, e=gmverb.sty,
 f=gmeometric.sty, g=gmoldcomm.sty

\OnlyDescription, 20,
a-7448
\opt, 23, a-7214
\oumlaut, b-340, b-358
outeroff, a-1882, b-182, 109

\PackageError, a-4079,
a-6368, a-6521, c-541,
c-1630, c-3347, c-3359
\PackageInfo, a-7360,
a-7368, e-436
\PackageWarning, c-1104,
c-1107
\PackageWarningNoLine,
a-5835
\pagebreak, c-1824,
c-1836, c-1840
\pagegoal, c-2882
\PageIndex, a-7410, a-7412
pageindex, 10, a-2046
pagella, b-254
\pagestyle, b-437
\pagetotal, c-2883
\paperheight, f-372
\paperwidth, f-371
\par, a-2308, a-2314,
a-2321, a-2412,
a-2461, a-2726,
a-2839, a-2989,
a-3010, a-3023,
a-3048, a-3099,
a-3378, a-5061,
a-5063, a-5072,
a-5074, a-5204,
a-5211, a-5424,
a-5633, a-5636,
a-6662, a-6698,
a-6703, a-6707,
a-6992, a-7007, a-7011
\paragraph, a-6571, c-3611
\ParanoidPostsec, b-310,
c-1989
\parg, c-1364
\parsep, e-533
\partial, c-2582
\partopsep, a-2280,
c-2195, c-2213, e-538
\PassOptionsToPackage,
b-194, b-259, b-270,
f-193
\pauza, c-3320
\pauza@skipcore, c-3298,
c-3309, c-3310
\pauzacore, c-3299,
c-3311, c-3316,
c-3324, c-3333,
c-3338, c-3369,
c-3372, c-3639, c-3640
\pauzadial, c-3326, c-3332
\pdef, a-2433, a-4224,
a-7123, a-7129,
a-7131, a-7512, c-182,
c-196, c-259, c-276,
c-295, c-296, c-317,
c-332, c-340, c-811,
c-847, c-888, c-1077,
c-1116, c-1117, c-1217,
c-1288, c-1295,
c-1310, c-1327,
c-1331, c-1394,
c-1405, c-1448,
c-1485, c-1507,
c-1529, c-1533,
c-1728, c-2250,
c-2435, c-2482,
c-2493, c-2504,
c-2546, c-2562,
c-2757, c-2771,
c-2779, c-2893,
c-2895, c-2897,
c-2987, c-3025,
c-3095, c-3196,
c-3305, c-3320,
c-3332, c-3337,
c-3346, c-3358,
c-3415, c-3542,
c-3581, c-3589,
c-3590, c-3591,
c-3605, c-3606,
c-3618, c-3621,
c-3730, c-3745,
c-3748, c-3751,
c-3767, c-3776,
c-3782, c-3834, c-3839
\pdfTeX, 22, c-2341
\pdfoutput, f-192
\pdfTeX, 22, c-2343
\perh@ps@grab@ms, c-498,
c-554
\Phi, c-2574
\phi, c-2573
\pi, c-2572
\pk, 21, a-1886, a-1887,
a-1917, a-6334, c-1310
\PlainTeX, 22, c-2329
\pm, c-2583, c-2584
\polskadata, c-3438, c-3479
\possfil, c-1336
\ppauza, c-3358
\ppauza@skipcore,
c-3301, c-3352, c-3353
\pprovide, a-4226, c-211,
c-2981
\predisplaypenalty,
e-481, e-490
prefix, a-3871
\prependtomacro, c-373
\prevhmodefalse,
a-2733, a-2782,
a-2876, a-3014, a-3043
\prevhmodetrue, a-2875
\PrintChanges, 16,
a-1927, a-6116,
a-6120, a-6480
\PrintDescribeEnv, 103
\PrintDescribeMacro, 103
\PrintEnvName, 103
\PrintFilesAuthors, 8,
a-6813
\PrintIndex, a-1931,
a-5645, a-6480
\printindex, a-5647,
a-5648, a-6480
\printlinenumber,
a-2761, a-2945,
a-3129, a-3135
\PrintMacroName, 103
\printsaces, c-1279, c-1288
\ProcessOptionsX, b-272
\protected, a-3270,
a-3397, a-3400,
c-182, c-211, c-271,
c-290, c-430, c-2753,
c-2755
\provide, a-4225, a-7210,
c-196, c-211
\providecolor, e-745
\ProvideFileInfo, 23,
a-6895, a-6911
\ProvidesClass, b-42
\ProvideSelfInfo, a-6911
\ps@plain, a-6688
\ps@titlepage, a-6688
\psi, c-2577

\quad, b-435, b-436, c-3424
\quantifierhook, c-2700,
c-2721
\QueerCharOne, a-3303,
a-3310, a-3312
\QueerCharTwo, a-3269,
a-3276, a-3278
\QueerEOL, 7, a-2381,
a-2504, a-3341,
a-5062, a-5073,

File Key: a=gmdoc.sty, b=gmdocc.cls, c=gmutils.sty, d=gmiflink.sty, e=gmverb.sty,
f=gmeometric.sty, g=gmoldcomm.sty

a-5647, a-6118,
 a-6643, a-7729, a-7730
 quotation, 22, a-6999
 \quote@char, a-3548,
 a-3569, a-3581,
 a-3605, a-4712
 \quote@charbychar,
 a-4706, a-4709, a-4714
 \quote@mname, a-4690,
 a-4704, a-4745, a-4830
 \quotecchar, 20, a-3479,
 a-3610, a-3625,
 a-4495, a-4538,
 a-4750, a-4833,
 a-5844, a-5904
 \quoted@eschar, a-4495,
 a-4538, a-4750,
 a-4751, a-4833, a-4834

 \raggedbottom, b-425
 \rdate, c-3605
 \real, c-3219, c-3221
 \RecordChanges, 16,
 a-5837, a-6028,
 a-6029, a-6480,
 b-443, b-446
 \reflectbox, c-2349, c-2356
 \relaxen, a-4245, a-4414,
 b-343, c-393, c-393,
 c-1867, c-2565,
 c-3389, c-3791, e-291,
 e-682
 \relsize, c-1077, c-1078,
 c-1112, c-1113, c-1114,
 c-1115, c-1116, c-1117, 127
 \rem@special, e-417,
 e-442, e-457
 \renewcommand, a-4268,
 a-5840
 \renewcommand*, a-3181,
 a-3183, b-430, c-2971
 \RequirePackage, a-2003,
 a-2005, a-2113,
 a-2116, a-2133,
 a-2145, a-2148,
 a-2154, a-5597, b-139,
 b-307, b-326, b-329,
 b-369, b-380, b-388,
 b-420, b-463, c-2416,
 c-2867, c-2964,
 c-3203, e-249, e-741,
 f-176
 \RequirePackageWithOptions,
 f-196
 \resetlinecountwith,
 a-3110

\resetMathstrut@, c-2632
 \resizebox, c-2536
 \resizegraphics, c-2533
 \Restore@Macro, c-1488,
 c-1491, c-1511, c-1521
 \Restore@Macros, c-1507,
 c-1509
 \Restore@MacroSt,
 c-1489, c-1497
 \RestoreEnvironment,
 c-1533
 \RestoreMacro, a-4247,
 a-4389, a-4390,
 a-4432, a-5547,
 a-6938, c-1485,
 c-2423, c-2425, f-199,
 g-64
 \RestoreMacro*, a-4456,
 a-4457, c-1535, c-2425
 \RestoreMacros, a-4922,
 c-1507, f-397
 \RestoringDo, a-6427, c-1570
 \ResumeAllDefining, 13,
 a-4387
 \ResumeDef, 13, a-4247
 \ResumeDefining, 13,
 a-4247, a-4450
 \reversemarginpar, a-5002
 \rightarrow, c-2612, c-2739
 \rightline, b-464, c-3605,
 c-3691
 \romannumeral, c-2193,
 c-2211
 \rotatebox, c-2601,
 c-2604, c-2609, c-2610
 \rs@size@warning,
 c-1096, c-1101, c-1104
 \rs@unknown@warning,
 c-1091, c-1107
 \runindef, c-3610

 \scan@macro, a-2897,
 a-3532, g-87
 \scantokens, a-6887, c-2620
 \scshape, c-2327, c-2733,
 c-3017
 \secondclass, c-3045
 \SecondClasstrue, c-3047
 \SelfInclude, 9, a-1909,
 a-6590
 \SetFileDiv, 22, a-6510,
 a-6513, a-6515,
 a-6523, a-6584, a-6606

\setkeys, a-3822, a-3829,
 a-3856
 \setmainfont, b-246
 \setmonofont, b-248
 \setsansfont, b-247
 \SetSectionFormatting,
 c-1867, c-1868,
 c-2032, c-2036,
 c-2044, c-2052,
 c-2059, c-2066, c-2071
 \settexcodehangi,
 a-2212, a-2217,
 a-2769, a-2777, a-3059
 \SetTOCIndents, b-435,
 b-436
 \SetTwoheadSkip, c-2016,
 c-2043, c-2051, c-2058
 \sfname, c-1288, c-1298
 \sgtleftxii, a-5721, a-5765
 \shortpauza, c-3371
 \shortthousep, c-3838
 \showboxbreadth, c-737
 \showboxdepth, c-737
 \ShowFont, c-2931
 \showlists, c-737
 \showthe, c-742
 \sigma, c-2575
 \sim, c-2585
 \SkipFilesAuthors, 9,
 a-6815
 \skipgmlonly, 22, a-1915,
 a-6957
 \skiplines, 25, a-2693
 \sl, b-249
 \SliTeX, 22, c-2326
 \smaller, c-1113, c-3062,
 c-3066, 127
 \smallerr, a-6772, c-1117,
 c-3234, 127
 \smallskipamount,
 a-2274, a-2275,
 c-2812, c-2813
 \smartunder, b-455, c-1156
 \SMglobal, a-3850, a-4378,
 a-4389, a-4390,
 a-4424, a-4432,
 a-4456, a-4457, c-1394
 \something@in, c-599,
 c-600, c-603, c-608
 \something@tmp, c-607,
 c-608, c-612
 \something@tmpb, c-611,
 c-612
 \SortIndex, a-7483
 \special, a-2907, a-2908

File Key: a=gmdoc.sty, b=gmdocc.cls, c=gmutils.sty, d=gmiflink.sty, e=gmverb.sty,
 f=gmeometric.sty, g=gmoldcomm.sty

\special@index, a-4500,
a-4902, a-4906, a-5039
\SpecialEnvIndex, a-7481
\SpecialEscapechar, a-7346
\SpecialIndex, a-7477
\SpecialMainEnvIndex,
a-7472
\SpecialMainIndex, a-7469
\SpecialUsageIndex, a-7479
\square, b-464
StandardModuleDepth,
a-7529
\stanza, 18, 22, a-1915,
a-1917, a-2311, a-6993
\stanzaskip, 18, a-2233,
a-2238, a-2239,
a-2264, a-2265,
a-2266, a-2271,
a-2272, a-2279,
a-2312, a-2728,
e-564, e-573, e-574
star, 13, a-3862
\step@checksum, a-3538,
a-6164
\StopEventually, 20,
a-7428, a-7448
\Store@Macro, c-1411,
c-1414, c-1451
\Store@Macros, c-1448,
c-1449
\Store@MacroSt, c-1412,
c-1421
\stored@code@delim, a-5110
\Stored@Macro, c-1520, c-1521
\storedcsname, a-7008,
a-7012, c-1524, c-3877
\StoredMacro, c-1520
\StoreEnvironment,
a-6997, c-1529
\StoreMacro, a-4241,
a-4378, a-4424,
a-5541, a-6933,
c-1405, c-2283,
c-2415, c-3639,
c-3876, f-187, g-36
\StoreMacro*, a-3850,
c-1531, c-2284
\StoreMacros, a-4920,
c-1448, f-185
\StoringAndRelaxingDo,
a-6410, c-1550
\StraightEOL, 7, a-2504,
a-3328, a-5647,
a-5831, a-6118,
a-6955, a-6968,
a-6991, a-7591
\strip@pt, c-3734, c-3739,
c-3756
\subdivision, 21, a-6570,
a-7153
\subitem, a-5634
\subs, c-1128, c-1158
\subsubdivision, 21,
a-6571, a-7156
\subsubitem, a-5635
\sum, c-2600
sysfonts, b-225, 109
\tableofcontents,
a-1905, a-2502,
a-2503, a-6479
\task, g-90
\TB, 22, c-2335
\TeXbook, 22, c-2334, c-2335
\Text@CommonIndex,
a-4758, a-4761
\Text@CommonIndexStar,
a-4758, a-4765
\text@indexenvir,
a-4732, a-4734,
a-4766, a-4985, a-7317
\text@indexmacro,
a-4688, a-4728,
a-4762, a-4977, a-7309
\Text@Marginize, a-3744,
a-4178, a-4806,
a-4975, a-4982,
a-4997, a-5018,
a-5282, a-7307, a-7314
\Text@MarginizeNext,
a-5274, a-5279, a-5281
\Text@UsgEnvir, a-4970,
a-4980
\Text@UsgIndex, a-4723,
a-4726
\Text@UsgIndexStar,
a-4723, a-4731
\Text@UsgMacro, a-4970,
a-4973
\textbullet, c-3415, c-3419
\textcolor, c-2954, e-748
\TextCommonIndex, 15,
a-4756
\textheight, f-374
\TextIndent, 18, a-2223,
a-3063, a-3215
\textlarger, c-1114
\textlit, c-3745
\TextMarginize, 15, a-4991
\textsl, b-249, c-2334
\textsmaller, c-1115
\textstyle, c-2281
\textsuperscript,
c-2541, c-2546
\texttilde, c-2779
\TextUsage, 14, a-4967
\TextUsgIndex, 15,
a-4721, a-7479
\textwidth, a-3086,
a-6274, a-6542,
c-3673, c-3674,
c-3692, c-3693, f-373
\thanks, a-6697, a-6714,
a-6754, a-6760, a-6917
\theCodelineNo, a-7361, 102
\thecodelinenum, a-2668,
a-3130, a-7363
\thedate, c-3615
\thefilediv, a-6443,
a-6535, a-6537,
a-6539, a-6556,
a-6559, a-6740
\theglossary, a-6481
\theglossary, a-6049
\theindex, a-5610
\thesection, b-430
\thfileinfo, 23, a-6917
\thickmuskip, c-2685
\thous, c-3839
\thous@inner, c-3841, c-3844
\thousep, c-3782, c-3834
\thr@@, c-2189, c-2207
\time, c-3409, c-3410
\tinycae, c-3201
\title, a-1886, a-6711
\titlesetup, a-6696,
a-6723, b-440
\TODO, c-2830
\toks, c-2003, c-2004,
c-2010, c-2011
\tolerance, a-2373,
c-3275, c-3286, c-3631
\traceoff, b-385
\traceon, b-384
\trimmed@everypar,
a-3246, a-3248
\trueittextsuperscript,
c-2543, c-2545

File Key: a=gmdoc.sty, b=gmdocc.cls, c=gmutils.sty, d=gmiflink.sty, e=gmverb.sty,
f=gmeometric.sty, g=gmoldcomm.sty

\ttverbatim, a-2463,
a-5112, a-7064, e-352,
e-571, e-599
\ttverbatim@hook, e-358,
e-365, e-368
\twocoltoc, a-1885,
c-2866, c-2875
\twopar, c-3669
\twoparinit, c-3667
type, 12, a-3919
\tytul, c-3621
tytulowa, c-3421

\udigits, c-2435, c-2438
\un@defentryze, a-4484,
a-4517
\un@usgentryze, a-4480,
a-4528
\UnDef, 13, a-4234, a-4241,
a-4245, a-4247,
a-4390, a-4414
\undeksmallskip, c-2813
\UndoDefaultIndexExclusions,
16, a-5540
\unexpanded, c-178, c-375,
c-708, c-709, c-713,
c-714, c-717, c-1908,
e-749
\ungag@index, a-4922, a-7418
\UniformSkips, 18,
a-2262, a-2283,
a-2288, a-2295
\unless, a-2322, a-3027,
a-3076, a-3091, c-413,
c-2408, c-2972
\upshape, c-3771
\resetlinecount, 10, a-2019
\Url, c-2690, c-2691
\url, c-3876, c-3877
\urladdstar, c-3873, c-3880
\usage, a-7521
\usc, c-3025, c-3027
\uscacro, c-3027
\usecounter, c-2200
\UsgEntry, 19, a-4598, a-7521
\uumlaut, b-341, b-359

\value, a-2643, c-1693
\varepsilon, c-2281,
c-2338, c-2569
\varnothing, c-2734
\varsigma, c-2576
\vartheta, c-2570
\vee, c-2609, c-2683

\verb, 21, a-3453, c-2415,
c-2423, e-384, e-408,
e-414, e-416, e-597, e-716
\verb*, e-383, e-597
\verb@balance@group,
e-618, e-621, e-652, e-654
\verb@egroup, a-3452,
e-621, e-652, e-655
\verb@eol@error, e-625
\verb@eolOK, e-639, e-647
\verb@verbatim, e-478
\verb@verbatim, e-478
\verb@verbatim*, e-490
\verb@verbatim@edef, e-512, e-516
\verb@verbatim@end, e-513, e-517
\verb@verbatim@nolig@list,
e-353, e-657
\verb@verbatimchar, 20,
a-3709, a-4470,
a-4694, a-5903,
a-5905, a-7503, 104
\verb@verbatimhangindent,
a-2213, e-567, e-578,
e-579
\verb@codecorr, a-3095
\verbbeolOK, 11, e-647, 173
\VerbHyphen, a-2193,
e-267, 173
\verbhyphen, a-7071,
e-261, e-269, e-279,
e-299
\VerbT, e-368
\VerbT₁, e-368
\visibleinspace, a-2827,
a-7066, c-1193,
c-1195, c-1273, e-330,
e-747, e-749
\VisSpacesGrey, 11,
a-2594, e-743, 174
\voffset, f-386
\Vs, c-2753
\vs, c-1273, c-1279, c-1283

\wd, c-2261, c-2264, c-2294,
c-2299, c-2307,
c-2308, c-2603,
c-2724, c-2725,
c-2736, c-2737
\Web, 22, c-2331
\Webern@Lieder@ChneOelze,
a-4315
\wedge, c-2610, c-2683
\whenonly, c-3158
\whern, c-3698

\wherncore, c-3690,
c-3699, c-3705
\whernskip, c-3699,
c-3702, c-3703
\whernup, c-3705
\widowpenalty, a-2368
\withmarginpar, 10, a-2080
\WPheadings, c-2031
\Ws, c-2755
\wyzejnizej, c-3242
\Wz, c-2757

\xathousep, c-3834, c-3848
\xdef@filekey, a-6417,
a-6421, a-6439
\Xedekfrac, c-2443
\XeLaTeX, c-2353
\XeTeX, 22, c-2346
\XeTeXdefaultencoding,
b-293, b-298
\XeTeXinputencoding, c-168
\XeTeXthree, b-347, c-2412
\XeTeXversion, c-157,
c-158, c-167, c-1133,
c-2407, c-2408,
c-3297, c-3379
\xiand, c-1179
\xiibackslash, c-1166, c-1170
\xiiclub, a-3480, a-5845,
e-343
\xiilbrace, a-7071,
a-7073, c-1141, e-279,
e-313
\xiipercent, a-5191,
a-5193, a-6234,
a-6238, a-7035,
a-7056, a-7066,
a-7068, a-7072,
a-7081, a-7090,
a-7116, c-1175, e-261
\xiirbrace, c-1142
\xiispace, a-3722, c-754,
c-755, c-1182, c-1195
\xiistring, a-3571,
a-4689, a-4735,
a-4953, a-4959,
a-4974, a-4981,
a-5019, c-753
\xiounder, c-1131, c-1134,
c-1135
\XKV@ifundefined,
b-292, b-297
\xparsed@args, c-417,
c-441, c-448

\xxt@visiblespace, c-1192, c-1193	\yeshy, <u>c-3089</u>	\zwrobcy, <u>c-3618</u>
	\z@skip, c-3658	\cdot, <u>c-2897</u>
	\zf@euencfalse, b-346	
\year, a-6236, a-6240	\zf@scale, <u>c-3209</u> , <u>c-3212</u> , c-3213	\-, <u>c-3346</u> , <u>c-3385</u> \-, <u>c-3305</u> , <u>c-3366</u> , <u>c-3384</u>