Grzegorz ‘Natror’ Murzynowski

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

August 2008

Contents

a. The gmdoc.sty Package 4
Readme 4
Installation 4
Contents of the gmdoc.zip Archive 5
Compiling the Documentation . . 5
Dependencies 5
Bonus: base Drivers 6
Introduction 6
The User Interface 6
Used Terms 6
Preparing the Source File 7
The Main Input Commands 8
Package Options 10
The Packages Required 11
Automatic marking of definitions . 12
Manual Marking the Macros and Environments 13
Index Ex/Inclusions 15
The DocStrip Directives 16
The Changes History 16
The Parameters 18
The Narration Macros 20
A Queerness of `\label` 22
doc-Compatibility 23
The Driver Part 23
The Code 25
The Package Options 26
The Dependencies and Preliminaries 27
The Core 30
Numbering (or Not) of the Lines . 39
Spacing with `\everypar` 40
Life Among Queer EOLS 41
Adjustment of `verbatim` and `\verb` 43
Macros for Marking The Macros . 44
Automatic detection of definitions 48
Indexing of css 58
Index Exclude List 71
Index Parameters 75
The DocStrip Directives 77
The Changes History 78
The Checksum 82
Macros from ltxdoc 84
`\DocInclude` and the ltxdoc-Like Setup 84
Redefinition of `\maketitle` 89
The File's Date and Version Information 92
Miscellanea 93
doc-Compatibility 97
gmdocing doc.dtx 102
Polishing, Development and Bugs . . 103
(No) ⟨*eof*⟩ 103

b. The gmdocc Class For gmdoc Driver Files 105
Intro 105
Usage 105
The Code 106

c. The gmutils Package 111
Intro 111
Contents of the gmutils.zip Archive 111
A couple of abbreviations 112
`\@ifnextcat`, `\@ifnextac` 114
`\afterfi` and Pals 116
Environments redefined 116
Almost an Environment or Redefinition of `\begin` 116
`\@ifenvir` and Improvement of `\end` 117
From relsize 118
`\firstofone` and the Queer `\catcodes` 119
Some 'other' stuff 119
Metasymbols 120
Macros for Printing Macros and Filenames 121
Storing and Restoring the Meanings of CSs 123
Not only preamble! 126
Third Person Pronouns 126
To Save Precious Count Registers . . 127
Improvements to mwcls Sectioning Commands 128
An improvement of MW's `\SetSectionFormatting` . . . 130
Negative `\addvspace` 131
My heading setup for mwcls 133
Compatibilising Standard and mwcls Sectionings 134
`enumerate*` and `itemize*` 135
The Logos 136
Expanding turning stuff all into 'other' 138

Brave New World of XƎTEX 139
Fractions 139
\resizegraphics 140
Varia 141
\@ifempty 145
\include not only .tex's 146
Faked small caps 147
See above/see below 148
luzniej and napapierki—environments used in page breaking for money . . . 148
Settings for mathematics in main font 151
Typesetting dates in my memoirs . . 154
Minion and Garamond Premier kerning and ligature fixes 157
d. The gmiflink Package 159
Introduction, usage 159
Contents of the gmiflink.zip archive 159
The Code 160
e. The gmverb Package 162
Intro, Usage 162
Contents of the gmverb.zip Archive 163
The Code 164
Preliminaries 164
The Breakables 164
Almost-Knuthian \ttverbatim . 165
The Core: From shortvrb 165
doc- And shortvrb-Compatibility . 170
Grey visible spaces 171
f. The gmeometric Package 172
Introduction, usage 172
Contents of the gmeometric.zip archive 172
Usage 173
The Code 173
g. The gmoldcomm Package 176
Change History 178
Index 182

a. The gmdoc.sty Package[1]

August 13, 2008

This is (a documentation of) file gmdoc.sty, intended to be used with LaTeX 2$_\varepsilon$ as a package for documenting (LA)TeX files and to be documented with itself.

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

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

```
\ifnum\catcode`\@=11␣% Why this test here—will come out in chapter The Driver.
\NeedsTeXFormat{LaTeX2e}
\ProvidesPackage{gmdoc}
          [2008/08/06␣v0.99m␣a␣documenting␣package␣(GM)]
\fi
```

Readme

This package is a tool for documenting of (LA)TeX 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 XeTeX (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 gmoldcomm.sty somewhere in the texmf/tex/latex branch on your own. (Creating a texmf/tex/latex/gm directory may be advisable if you consider using other packages written by me. And you *have* to use four of them to make gmdoc work.)

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 v0.99m dated 2008/08/06.

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 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 XƎLATEX 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 XƎLATEX on gmdoc.sty once more. (Using LATEX instead of XƎLATEX 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 LATEX 2ε 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_gmdoc.tex
docstrip_gmdoc.tex
doc_gmdoc.tex

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

These drivers typeset the respective files from the

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

directory of the TEXLive2007 distribution (they only read that directory).

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

Introduction

There are very sophisticated and effective tools for documenting LATEX 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 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 gmdoc makes use of hyperref advantages and makes a hyperlinking index and toc entries and the cross-references, too. (The css 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 it[2]. 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 TEXbook*'s meaning, i.e., a control sequence whose meaning is `\(e/g/x)defined`. By a **macro's parameter** I mean each of #⟨*digit*⟩s in its definition. When I write about a **macro's argument**, I mean the value (list of tokens) subsituting the corresponding parameter of this macro. (These understandings are according to *The TEXbook*, I hope: 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 `\itshape` or `\@onlypreamble{`⟨*cs*⟩`}`.

Talking of declarations, I'll use the **OCSR** acronym as a shorthand for 'observes/ing common 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 `\catcode`s, it's just the `%`. When the file is documented with gmdoc, such a char is re`\catcode`d and its rôle is else: it becomes the **code delimiter**.

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

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

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

Preparing the Source File

When (LA)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 (LA)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. `\^^M` `^^B`

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. `^^A`

Note that `^^A` behaves much like comment char although it's active in fact: it re`\catcode`s the special characters including `\`, `{` and `}` so you don't have to worry about unbalanced braces or `\if`s 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 queer[3] 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 `\StraightEOL` `\QueerEOL`

[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 LATEX's \include and very likely to ltxdoc's command of the same name. In gmdoc it has one mandatory argument that should be the file name *without extension*, just like for \include.

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 \PrintFilesAuthors be printed, you should write \PrintFilesAuthors in the preamble or before the rel-

\SkipFilesAuthors

evant `\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 filemarker 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.

\SelfInclude

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 `\maketitle`s of those files result in the respective headings.

\ltxLookSetup

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' `\maketitle`s 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)).

\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.

\AtBegInput

If you need to add something at the beginning of the input of file, there's the `\AtBegInput` 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.

\AtBegInputOnce

If you need to add something at the beginning of input of only one file, put before the respective input command an `\AtBegInputOnce{`⟨*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 succeding it (the argument gets wrapped in a macro that's `\relax`ed at the first use). `\AtBegInputOnce`s add up, too.

\IndexInput

One more input command is `\IndexInput` (the name and idea of effect comes from doc). It takes the same argument as `\DocInput`, the file's (path and) name with extension. (It *has* `\DocInput` inside). It works properly if the input file doesn't contain explicit ⟨*char1*⟩ (`^^A` is ok).

The effect of this command is typesetting of all the input file verbatim, with the code lines numbered and the css automatically indexed (gmdoc.sty options are in force).

Package Options

As many good packages, this also provides some options:

linesnotnum

Due to best TEX 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 css occurring in the code are automatically (hyper)indexed thanks to the hyperref package. If the user doesn't wish to index anything, she should use the `noindex` option.

pageindex

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

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

By default, gmdoc excludes some 300 css from being indexed. They are the most common css, LATEX internal macros and TEX primitives. To learn what css are excluded actually, see lines 5211–5337.

indexallmacros

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

If you have ambiguous feelings about whether to let the default exclusions or forbid them, see p. 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 `visspacesgrey` and by default it's defined as `{gray}{.5}`, you can change it with xcolor's `\definecolor`. There is also an ocsr declaration `\CodeSpacesGrey`.

\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

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

gmverb

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

\verbeolOK

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 `\verbeolOK` declaration. The plain `\verb` typesets spaces blank and `\verb*` makes them visible, as in the standard version(s).

\MakeShortVerb

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

```
\MakeShortVerb*\|
```

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

\dekclubs

\DeleteShortVerb

Be careful because such active chars may interfere with other things, e.g., the `|` with the vertical line marker in `tabular`s 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 `\active`ated by `\MakeShortVerb`, behave as if they were 'other' in math mode, so you may type e.g., `$k|n$` to get $k|n$ etc.

gmutils

The gmutils package provides a couple of macros similar to some basic (LA)TeX 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 *before* loading gmdoc.

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

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

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

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

Automatic marking of definitions

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

1. `\def`, `\newcount`, `\newdimen`, `\newskip`, `\newif`, `\newtoks`, `\newbox`, `\newread`, `\newwrite`, `\newlength`, `\newcommand(*)`, `\renewcommand(*)`, `\providecommand(*)`, `\DeclareRobustCommand(*)`, `\DeclareTextCommand(*)`, `\DeclareTextCommandDefault(*)`,
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 css uderlying those definitions, e.g. `\newcounter{foo}` in the code will result in indexing `foo` and `\c@foo`.

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

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

`dk` stands for `\define@key` and is the type of xkeyval definitions of keys (group 4 commands). When detected, it scans furher 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.

\DeclareKVOFam

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.

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 \@namedef to be detected (the original LaTeX version), declare

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

or

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

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

\HideDefining
\ResumeDefining

On the other hand, if you want some of the commands listed above *not* to be detected, write \HideDefining⟨*command*⟩ in the commentary. Later you can resume detection of it with \ResumeDefining⟨*command*⟩.

\HideAllDefining
\ResumeAllDefining

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

\UnDef

The basic definition command, \def, seems to me a bit controversial. 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, in the commentary before this \def write \UnDef. That will turn off the detection just for this one occurrence of \def.

\HideDef
\ResumeDef

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. 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 \Define declaration described in the next section.

Manual Marking 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 exluding some macros from being indexed.)

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

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

\Define \CodeUsage \TextUsage

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 ^^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 unstarred.

\MakePrivateLetters

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

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

right before the code line

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

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

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

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

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

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

\Describe

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

But what if you want just to make a marginpar with macro's or environment's name? Then you have \CodeMarginize to declare what to put into a marginpar in the TEX code (it's \outer) and \TextMarginize to do so in the commentary. According to the spirit of this part of the interface, these commands also take one argument and have their starred versions for strings other than control sequences.

\CodeMarginize \TextMarginize

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 declara-

\marginpartt

tion is \let to \tt but it may be advisable to choose a condensed font if there is any. Such a choice is made by gmdocc.cls if the Latin Modern fonts are available: in this case gmdocc.cls uses Latin Modern Typewriter Light Condensed.

\gmdmarginpar

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

\DefIndex
\CodeUsgIndex

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

\CodeCommonIndex*

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

\TextUsgIndex
\TextCommonIndex

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

macro
environment

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

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

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

Index Ex/Inclusions

\DoNotIndex

It's understandable[4] that you don't want some control sequences to be indexed in your documentation. The doc package gives a brilliant solution: the \DoNotIndex declaration. So do I (although here, TEXnically it's done another way). It OCSR. This declaration takes one argument consisting of a list of control sequences not to be indexed. The items 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 css 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.

[4] After reading doc's documentation ;-) .

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

\DoIndex

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 css given in the argument, e.g.,

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

\DefaultIndexExclusions
\UndoDefaultIndexExclusions

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 TeX 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{`⟨*version*⟩`}{`⟨*YYYY/MM/DD date*⟩`}{`⟨*text*⟩`}`

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

\RecordChanges
\PrintChanges

To cause the change information to be written out, include `\RecordChanges` in the driver['s preamble or just in the source file (gmdocc.cls does it for you)]. To read in and print the sorted change history (in two columns), just put the `\PrintChanges` 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. 19]. (The LaTeX 'glossary' mechanism is used for the change entries.)"

\GlossaryMin
\GlossaryPrologue
\GlossaryParms

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{v0.99e}{2007/04/29}{renamed from \cs{DodecaPhonic}}
```

results with a history (sub)entry:

```
v0.99e
   (...)
   \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

\MakeGlossaryControls

```
\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

ChangesStartDate
\ChangesStart

```
\ChangesStart{⟨version⟩}{⟨year⟩/⟨month⟩/⟨day⟩}
```

declaration.

If the ChangesStartDate is set to a date contemporary to TEX i.e., not earlier than September 1982[5], 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 TEX, no notification of ommitting shall be printed. This is intended for a rather tricky usage of the changes start date feature: you may establish two threads of the changes history: the one for the users, dated with four digit year, and the other for yourself only, dated with two or three digit year. If you declare

```
\ChangesStart{⟨version?⟩}{1000/00/00}
```

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

\CheckSum

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

```
% \chschange{v0.98j}{2006/10/19}{4372}
% \chschange{v0.98j}{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*⟩.

[5] DEK in *TEX The Program* mentions that month as of TEX Version 0 release.

The Parameters

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

\stanzaskip \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.

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), \CodeTopsep a \CodeTopsep glue is added. At the beginning and the end of a macro or environment environment a \MacroTopsep glue is added. By default, these two skips are set equal \stanzaskip.

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

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

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

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

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

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

\EOFMark given by the \EOFMark macro.

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

The crucial concept of gmdoc is to use the line end character as a verbatim group opener and the comment char, usually the %, as its delimiter. Therefore the 'knowledge' what char starts a commentary is for this package crucial and utterly important. The default assumption is that you use % as we all do. So, if you use another character, then \CodeDelim 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 deafult.)

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, \CodeEscapeChar 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.)

`\MakePrivateLetters`

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.

`\AddtoPrivateOthers`

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 `\MakeShortVerb`ed. (I don't see a need of removing chars from this list, but if you do, please let me know.)

`\LineNumFont` `codelinenum`

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

`\IndexPrefix` `\EntryPrefix` `\HLPrefix`

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.

`\UsgEntry` `\DefEntry`

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

`\CommonEntryCmd`

There's one more parameter macro, `\CommonEntryCmd` that stores the name of the encapsulation for the 'common' index entries (not special) i.e., a word that'll become a cs that will be put before an entry in the .ind file. By default it's defined as `{%relax}` and a nontrivial use of it you may see in the source of chapter 641, where `\def%\CommonEntryCmd{UsgEntry}` makes all the index entries of the driver formatted as 'usage'.

`IndexColumns` `\IndexMin`

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` = 133.0pt.

`\IndexPrologue` `\AtDIPrologue`

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 182. 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 `\AtDIPrologue` 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.

\IndexLinksBlack 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.

\IndexParms \gaddtomacro Other index parameters are set with the `\IndexParms` macro defined in line 5449 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 LaTeX's `\g@addto@macro` provided by gmutils.)

\actualchar \quotechar \levelchar \encapchar 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 TeX what characters of a scanned cs name it should quote before writing it to the .idx file.

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

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

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

The `\OnlyDescription` and `\AlsoImplementation` declarations are intended to exclude or include the code part from the documentation. The point between the description and the implementation part should be marked with `\StopEventually{%` ⟨*the stuff to be executed anyway*⟩`}` and `\Finale` should be typed at the end of file. Then `\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

\env

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.

\incs \inenv

For usage in the in-line comments there are \incs and \inenv commands that take analogous arguments and precede the typeset command and environment names with a % if at the beginning of a new line.

\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 \+.

\pk \file

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

\catletter \catother \catactive

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.

\division \subdivision \subsubdivision

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

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

It's resolved in 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).

\SetFileDiv

If you wish the \DocIncluded files make other sectionings than the default, you may declare \SetFileDiv{⟨*sec name without backslash*⟩}.

gmlonely \skipgmlonely

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

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

gmdoc (gmutils, to be precise) provides some TeX-related logos:

\AmSTeX typesets $\mathcal{AMS}$-TeX,
\BibTeX BibTeX,
\SliTeX SliTeX,
\PlainTeX Plain TeX,
\Web Web,
\TeXbook *The TeXbook*,
\TB *The TeXbook*
\eTeX ε-TeX,
\pdfeTeX pdfε-TeX
\pdfTeX pdfTeX

[6] See gmutils for some subtle details.

\XeTeX XƎTeX (the first E will be reversed if the graphics package is loaded or XƎTeX is at work) and
\LaTeXpar (La)TeX.
\ds DocStrip not quite a logo, but still convenient.

copyrnote The 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' \gmdmarginpar 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 \stanza and \chunkskip. The first adds \stanzaskip while the latter \MacroTopsep. \chunkskip Both of them take care of not cumulating the vspaces.

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

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

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

If the input file doesn't contain \Provides... in the code layer, there are com- \ProvideFileInfo mands \ProvideFileInfo{⟨*file name with extension*⟩}[⟨*info*⟩]. (⟨*info*⟩ should consist of: ⟨*year*⟩/⟨*month*⟩/⟨*day*⟩ ⟨*version number*⟩ ⟨*a short note*⟩.)

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 \FileInfo \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 \thfileinfo title (or heading, with \DocInclude at the default settings), there's \thfileinfo macro that puts \fileinfo in \thanks.

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

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

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

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 LaTeX.

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` `\AfterMacrocode{`⟨*mc number*⟩`}{`⟨*the stuff*⟩`}` defines control sequence `\gmd@mchook`⟨*mc number*⟩ with the meaning ⟨*the stuff*⟩ and every `oldmc` and, when

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.)

`\OldDocInput` `\DocInclude` `\olddocIncludes` `macrocode` 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`.)

If you are used to the traditional (doc's) `macrocode` and still wish to use gmdoc new `oldmc` 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). `\OldMacrocodes` The other is to write `\OldMacrocodes`. 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.

⟨*package⟩

The Driver Part

In case of a single package, such as gmutils, 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]{gmdocc}
\usepackage{eufrak}% for |\continuum| in the commentary.
\twocoltoc
```

```
\begin{document}

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

\fi% of driver pass
%\endskiplines
```

\skiplines
\endskiplines

The advantage of \skiplines...\endskiplines over \iffalse...\fi is that the latter has to contain balanced \ifs and \fis while the former hasn't because it 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.

```
\ifnum\catcode`\@=12

\documentclass[countalllines,␣codespacesgrey,␣outeroff,␣debug,␣
     mwrep,
pagella]{gmdocc}
\twocoltoc
\title{The␣\pk{gmdoc}␣Package\\␣i.e.,␣\pk{gmdoc.sty}␣and
  \pk{gmdocc.cls}}
\author{Grzegorz␣`Natror'␣Murzynowski}
\date{August␣2008}

%\includeonly{gmoldcomm}

\begin{document}

\maketitle

\setcounter{page}{2}% hyperref cries if it sees two pages numbered 1.
\tableofcontents
\DoIndex\maketitle

\SelfInclude
\DocInclude{gmdocc}
```

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

```
\skipgmlonely[\stanza␣The␣remarks␣about␣installation␣and␣
     compiling
  of␣the␣documentation␣are␣analogous␣to␣those␣in␣the␣chapter
  \pk{gmdoc.sty}␣and␣therefore␣ommitted.\stanza]
\DocInclude{gmutils}
\DocInclude{gmiflink}
```

```
\DocInclude{gmverb}
\DocInclude{gmeometric}
\DocInclude{gmoldcomm}
\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}
\PrintChanges
\typeout{%
  Produce␣index␣with^^J%
  makeindex␣-r␣\jobname^^J}
 \PrintIndex
```

```
\afterfi{%
\end{document}
```

MakeIndex shell commands:

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

(gmglo.ist should be put into some texmf/makeindex directory.)

And "That's all, folks" ;-) .

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

The Code

For debug

```
\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 css, 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 tomake a `\section` or `\emphasise`, he has to type respective css.

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

```
\RequirePackage{gmutils}% includes redefinition of \newif to make the switches
    % \protected
\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
```
\newif\if@linesnotnum
```

linesnotnum
```
\DeclareOption{linesnotnum}{\@linesnotnumtrue}
```

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

\if@uresetlinecount
```
\newif\if@uresetlinecount
```

uresetlinecount
```
\DeclareOption{uresetlinecount}{\@uresetlinecounttrue}
```

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

\if@countalllines
\if@printalllinenos
```
\newif\if@countalllines
\newif\if@printalllinenos
```

countalllines
```
\DeclareOption{countalllines}{%
  \@countalllinestrue
  \@printalllinenosfalse}
```

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

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

\if@noindex
```
\newif\if@noindex
```

noindex
```
\DeclareOption{noindex}{\@noindextrue}
```

\if@pageindex
```
\newif\if@pageindex
```

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

It would be a great honour to me if someone would like to document LaTeX 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
```
\newif\if@indexallmacros
```

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

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

\if@marginparsused
```
\@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`:

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

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

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

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

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

nomarginpar

```
\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

```
\DeclareOption{codespacesblank}{%
  \AtEndOfPackage{% to allow codespacesgrey,␣codespacesblank
  \AtBeginDocument{\CodeSpacesBlank}}}
```

codespacesgrey

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

```
\ProcessOptions
```

The Dependencies and Preliminaries

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

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

A standard package for defining colours,

```
\RequirePackage{xcolor}
```

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

```
\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.

```
\@ifpackageloaded{hyperref}{\hypersetup{colorlinks=true,
    linkcolor=deepblue,␣urlcolor=blue,␣filecolor=blue}}{%
  \RequirePackage[colorlinks=true,␣linkcolor=deepblue,␣
       urlcolor=blue,
  filecolor=blue,␣pdfstartview=FitH,␣pdfview=FitBH,
  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.

```
\RequirePackage{gmiflink}
```

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

```
\RequirePackage{gmverb}[2007/11/09]
```

```
\if@noindex
  \AtBeginDocument{\gag@index}% for the latter macro see line 4743.
\else
   \RequirePackage{makeidx}\makeindex
\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

```
\def\CodeDelim{\@ifstar\Code@Delim@St\Code@Delim}
```

\Code@Delim
\code@delim

```
\def\Code@Delim#1{%
  {\escapechar\m@ne
    \@xa\gdef\@xa\code@delim\@xa{\string#1}}}
```

(`\@xa` is `\expandafter`, see gmutils.)

\Code@Delim@St

```
\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:

```
\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

```
\newtoks\gmd@preverypar
```

\settexcodehangi

```
\newcommand*\settexcodehangi{%
  \hangindent=\verbatimhangindent␣\hangafter=\@ne}% we'll use it in the
          inline comment case. \verbatimhangindent is provided by the gmverb
          package and = 3 em by default.
\@ifdefinable\@@settexcodehangi{\let\@@settexcodehangi=%
      \settexcodehangi}
```

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

\TextIndent

```
\newlength\TextIndent
```

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

\CodeIndent

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

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

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

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

\stanzaskip

```
\stanzaskip=\medskipamount
\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

```
\newskip\CodeTopsep
\newskip\MacroTopsep
```

And let's set them. For æsthetic minimality[7] let's unify them and the other most important vertical spaces used in gmdoc. I think a macro that gathers all these assignments may be handy.

\UniformSkips \CodeTopsep \MacroTopsep

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

We make it the default,

```
\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.

```
\AtBeginDocument{\UniformSkips}
```

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

\NonUniformSkips

```
\newcommand*\NonUniformSkips{\@relaxen\UniformSkips}
```

Why do we launch \UniformSkips twice then? The first time is to set all the gmdoc-specific glues *somehow*, 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

```
\newcommand*\chunkskip{%
  \skipo=\MacroTopsep
  \if@codeskipput\advance\skipo␣by-\CodeTopsep\fi
  \par\addvspace{\skipo}\@codeskipputgtrue}
```

And, for a smaller part of text,

\stanza

```
\newcommand*\stanza{%
  \skipo=\stanzaskip
  \if@codeskipput\advance\skipo␣by-\CodeTopsep\fi
  \par\addvspace{\skipo}\@codeskipputgtrue}
```

Since the stanza skips are inserted automatically most often (cf. lines 2670, 3032, 2685, 2944, 3082), sometimes you may need to forbid them.

\nostanza

```
\newcommand*\nostanza{%
```

[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)

```
2266  \@codeskipputgtrue\@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

```
2273 \newgif\if@newline
```

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

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

```
\if@dsdir   2281 \newgif\if@dsdir
```

This switch will be falsified at the first char of a code line. (We need a switch independent of the one indicationg 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 \inputing command that'll change catcodes. The macros used by it are defined later.

```
\DocInput   2294 \newcommand*\DocInput{\bgroup\@makeother\_\Doc@Input}
            2296 \begingroup\catcode`\^^M=\active%
            2297 \firstofone{\endgroup%
\Doc@Input  2298   \newcommand*{\Doc@Input}[1]{\egroup\begingroup%
            2301     \edef\gmd@inputname{#1}% we'll use it in some notifications.
            2303     \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.
            2308     \gmd@setclubpenalty% we wrapped the assignment of \clubpenalty in a macro because we'll repeat it twice more.
            2310     \@clubpenalty\clubpenalty␣\widowpenalty=3333␣% Most paragraphs of the code will be one-line most probably and many of the narration, too.

            2315     \tolerance=1000␣% as in doc.
            2318     \@xa\@makeother\csname\code@delim\endcsname%
            2320     \gmd@resetlinecount% due to the option uresetlinecount we reset the linenumber counter or do nothing.
^^M         2323     \QueerEOL% It has to be before the begin-input-hook to allow change by that hook.
            2328     \@beginputhook% my first use of it is to redefine \maketitle just at this point not globally.
            2330     \everypar=\@xa{\@xa\@codetonarrskip\the\everypar}%
\gmd@guardedinput 2332 \edef\gmd@guardedinput{%
            2333       \@nx\@@input␣#1\relax% \@nx is \noexpand, see gmutils. \@@input is the true TEX's \input.
            2337       \gmd@iihook% cf. line 6764
            2338       \@nx\EOFMark% to pretty finish the input, see line 2498.
            2340       \@nx\CodeDelim\@xa\@nx\csname\code@delim\endcsname% to ensure the code delimiter is the same as at the beginning of input.
            2345       \@nx^^M\code@delim%
            2347     }% we add guardians after \inputing a file; somehow an error occurred without them.
```

```
    \catcode`\%=9␣% for doc-compatibility.
    \setcounter{CheckSum}{0}% we initialize the counter for the number of the escape chars (the assignment is \global).
    \everyeof{\relax}% \@nx moved not to spoil input of toc e.g.
    \@xa\@xa\@xa^^M\gmd@guardedinput%
    \par%
    \@endinputhook% It's a hook to let postpone some stuff till the end of input. We use it e.g. for the doc-(not)likeliness notifications.
    \glet\@currentlabel=\gmd@currentlabel@before% we restore value from before this group. In a very special case this could cause unexpected behaviour of crossrefs, but anyway we acted globally and so acts hyperref.
    \endgroup%
  }% end of \Doc@Input's definition.
}% 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 `^^M` will be let to. `\gmd@textEOL` will be used also for checking the `%^^M` case (`\@ifnextchar` does `\ifx`).

```
\gmd@textEOL  \protected\def\gmd@textEOL{␣% a space just like in normal TeX. We put it first to cooperate with \^^M's \expandafter\ignorespaces. It's no problem since a space $_{10}$ doesn't drive TeX out of the vmode.
  \ifhmode\@afternarrgtrue\@codeskipputgfalse\fi% being in the horizontal mode means we've just typeset some narration so we turn the respective switches: the one bringing the message 'we are after narration' to True (@afternarr) and the 'we have put the code-narration glue' to False (@codeskipput). Since we are in a verbatim group and the information should be brought outside it, we switch the switches globally (the letter g in both).
  \@newlinegtrue% to \refstep the lines' counter at the proper point.
  \@dsdirgtrue% to handle the DocStrip directives.
  \@xa\@trimandstore\the\everypar\@trimandstore% we store the previous value of \everypar register to restore it at a proper point. See line 3115 for the details.
  \begingroup%
  \gmd@setclubpenalty% Most paragraphs will be one-line most probably. Since some sectioning commands may change \clubpenalty, we set it again here and also after this group.
  \aftergroup\gmd@setclubpenalty%
  \let\par\@@par% inside the verbatim group we wish \par to be genuine.
  \ttverbatim% it does \tt and makes specials other or \active-and-breakable.
  \gmd@DoTeXCodeSpace%
  \@makeother\|% because \ttverbatim doesn't do that.
  \MakePrivateLetters% see line 3370.
  \@xa\@makeother\code@delim% we are almost sure the code comment char is among the chars having been $_{12}$ed already. For 'almost' see the \IndexInput 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.

```
  \@xa\@ifnextchar\@xa{\code@delim}{%
    \gmd@continuenarration}{%
```

```
                      2422     \gmd@dolspaces% it will launch \gmd@typesettexcode.
                      2423   }% end of \@ifnextchar's else.
                      2424 }% end of \gmd@textEOL's definition.
```

```
\gmd@setclubpenalty   2426 \def\gmd@setclubpenalty{\clubpenalty=3333␣}
```

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

```
\AtEndInput           2430 \def\AtEndInput{\g@addto@macro\@endinputhook}
\@endinputhook        2431 \def\@endinputhook{}
```

Simili modo

```
\AtBegInput           2434 \def\AtBegInput{\g@addto@macro\@beginputhook}
\@beginputhook        2435 \def\@beginputhook{}
```

For the index input hooking now declare a macro, we define it another way at line 6764.

```
                      2439 \emptify\gmd@iihook
```

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

```
                      2444 \AtBegInput{\let\gmd@@toc\tableofcontents
\tableofcontents      2445   \def\tableofcontents{%
                      2446     \@ifQueerEOL{\StraightEOL\gmd@@toc\QueerEOL}%
                      2447     {\gmd@@toc}}}
```

As you'll learn from lines 3211 and 3198, we use those two strange declarations to change and restore the very special meaning of the line end. Without such changes `\tableofcontents` would cause a disaster (it did indeed). And to check the catcode of `^^M` is the rôle of `\@ifEOLactive`:

```
\@ifEOLactive         2459 \long\def\@ifEOLactive#1#2{%
                      2460   \ifnum\catcode`\^^M=\active␣\afterfi{#1}\else\afterfi{#2}\fi}
```

```
                      2462 \foone\obeylines{%
\@ifQueerEOL          2463   \long\def\@ifQueerEOL#1#2{%
                      2464     \@ifEOLactive{\ifx^^M\gmd@textEOL\afterfi{#1}\else\afterfi{%
                                   #2}\fi}%
                      2465     {#2}}% of \@ifQueerEOL
                      2466 }% of \foone
```

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

```
\gmd@ABIOnce          2477 \@emptify\gmd@ABIOnce
                      2478 \AtBegInput\gmd@ABIOnce
```

```
\AtBegInputOnce       2480 \long\def\AtBegInputOnce#1{%
                      2493   \gaddtomacro\gmd@ABIOnce{\g@emptify\gmd@ABIOnce#1}}
```

Many tries of finishing the input cleanly led me to setting the guardians as in line 2345 and to

```
\EOFMark              2498 \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 7502.

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

```
                      2510 \begin{obeyspaces}%
```

```
                           2511 \gdef\CodeSpacesVisible{%
\gmd@DoTeXCodeSpace        2512 \def\gmd@DoTeXCodeSpace{%
                           2513 \obeyspaces\let␣=\breakablevisspace}}%
\CodeSpacesBlank           2520 \gdef\CodeSpacesBlank{%
                           2521 \let\gmd@DoTeXCodeSpace\gmobeyspaces%
                           2522 \let\gmd@texcodespace=\ }% the latter \let is for the \if...s.
\CodeSpacesSmall           2525 \gdef\CodeSpacesSmall{%
\gmd@DoTeXCodeSpace        2526 \def\gmd@DoTeXCodeSpace{%
                           2527 \obeyspaces\def␣{\,\hskip\z@}}%
\gmd@texcodespace          2528 \def\gmd@texcodespace{\,\hskip\z@}}%
                           2530 \end{obeyspaces}
\CodeSpacesGrey            2532 \def\CodeSpacesGrey{%
                           2535   \CodeSpacesVisible
                           2536   \VisSpacesGrey% defined in gmverb
                           2537 }%
```

Note that `\CodeSpacesVisible` doesn't revert `\CodeSpacesGrey`.

```
                           2542 \CodeSpacesVisible
```

How the continuing of the narration should look like?

```
\gmd@continuenarration     2546 \def\gmd@continuenarration{%
                           2547   \endgroup
                           2548   \gmd@cpnarrline% see below.
                           2549   \@xa\@trimandstore\the\everypar\@trimandstore
                           2550   \everypar=\@xa{\@xa\@codetonarrskip\the\everypar}%
                           2551   \@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 2419. And in line 2779 it's been read as $_{12}$. That's why it works in spite of that `%` is of category 'ignored'.)

```
                           2558 \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@        2568   \def\gmd@countnarrline@{%
                           2569       \gmd@grefstep{codelinenum}\@newlinegfalse
                           2570       \everypar=\@xa{%
                           2571         \@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.
                           2576     }% of \gmd@countnarrline@
\gmd@grefstep              2578     \def\gmd@grefstep#1{% instead of diligent redefining all possible commands
                                      and environments we just assign the current value of the respective TeX's
                                      primitive to the codelinenum counter. Note we decrease it by −1 to get
                                      the proper value for the next line. (Well, I don't quite know why, but it
                                      works.)
                           2585       \ifnum\value{#1}<\inputlineno
                           2586         \csname␣c@#1\endcsname\numexpr\inputlineno-1\relax
                           2587         \ifvmode\leavevmode\fi% this line is added 2008/08/10 after an all-
                                          night debuggery ;-) that showed that at one point \gmd@grefstep
```

was called in vmode which caused adding \penalty 10000 to the main vertical list and thus forbidding pagebreak during entire % oldmc.

```
    \grefstepcounter{#1}%
  \fi}% We wrap stepping the counter in an \ifnum to avoid repetition of the same ref-value (what would result in the “multiply defined labels” warning).
```

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

\gmd@cpnarrline

```
  \if@printalllinenos% Note that checking this swich makes only sense when countalllines is true.
    \def\gmd@cpnarrline{% count and print narration line
      \if@newline
        \gmd@countnarrline@
        \hyperlabel@line
        {\LineNumFont\thecodelinenum}\,\ignorespaces}%
      \fi}
    \else% not printalllinenos
      \emptify\gmd@cpnarrline
    \fi
```

\gmd@ctallsetup

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

\else% not countallines (and therefore we won't print the narration lines' numbers either)
  \@emptify\gmd@cpnarrline
  \let\gmd@grefstep\hgrefstepcounter% if we don't want to count all the lines, we only \ref-increase the counter in the code layer.
  \emptify\gmd@ctallsetup
\fi% of \if@countalllines
```

\skiplines

```
\def\skiplines{\bgroup
  \let\do\@makeother␣\dospecials␣% not \@sanitize because the latter doesn't recatcode braces and we want all to be quieten.
  \catcode`\^^M\active
  \gmd@skiplines}

  \edef\gmu@tempa{%
    \long\def\@nx\gmd@skiplines##1\bslash␣endskiplines{\egroup}}
  \gmu@tempa
```

And typesetting the TEX code?

\gmd@typesettexcode

```
\foone\obeylines{%
  \def\gmd@typesettexcode{%
    \gmd@parfixclosingspace% it's to eat a space closing the paragraph, see below. It contains \par. A verbatim group has already been opened by \ttverbatim and additional \catcode.
```

```
               \everypar={\@@settexcodehangi}% At first attempt we thought of giving
                     the user a \toks list to insert at the beginning of every code line, but
                     what for?
^^M            \def^^M{%
\@newlinegtrue   \@newlinegtrue% to \refstep the counter in proper place.
                 \@dsdirgtrue% to handle the DocStrip directives.
                 \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.
                 \ifhmode\par\@codeskipputgfalse\else%
                   \if@codeskipput%
                   \else\addvspace{\stanzaskip}\@codeskipputgtrue%
                   \fi% if we've just met a blank (code) line, we insert a \stanzaskip glue.
                 \fi%
                 \prevhmodegfalse% we want to know later that now we are in the vmode.
                 \@ifnextchar{\gmd@texcodespace}{%
                   \@dsdirgfalse\gmd@dolspaces}{\gmd@charbychar}%
               }% end of ^^M's definition.
               \let\gmd@texcodeEOL=^^M% for further checks inside \gmd@charbychar.
               \raggedright\leftskip=\CodeIndent%
               \if@aftercode\gmd@nocodeskip1{iaC}\else\if@afternarr%
                   \if@codeskipput\else\gmd@codeskip1\@codeskipputgtrue%
                       \@aftercodegfalse\fi%
               \else\gmd@nocodeskip1{naN}\fi\fi% if now we are switching from the
                     narration into the code, we insert a proper vertical space.
               \@aftercodegtrue\@afternarrgfalse%
               \ifdim\gmd@ldspaceswd>\z@% and here the leading spaces.
                 \leavevmode\@dsdirgfalse%
                 \if@newline\gmd@grefstep{codelinenum}\@newlinegfalse%
                 \fi%
                 \printlinenumber% if we don't want the lines to be numbered, the respective
                       option \lets this cs to \relax.
                 \hyperlabel@line%
                 \mark@envir% index and/or marginize an environment if there is some to
                       be done so, see line 4633.
                 \hskip\gmd@ldspaceswd%
                 \advance\hangindent␣by\gmd@ldspaceswd%
                 \xdef\settexcodehangi{%
                   \@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.
                   \@nx\hangafter=1\space}%
               \else%
                 \glet\settexcodehangi=\@@settexcodehangi%
                       % \printlinenumber here produced line numbers for blank lines
                       which is what we don't want.
               \fi% of \ifdim
               \gmd@ldspaceswd=\z@%
               \prevhmodegfalse% we have done \par so we are not in the hmode.
               \@aftercodegtrue% we want to know later that now we are typesetting a codeline.
               \gmd@charbychar% we'll eat the code char by char to scan all the macros and
```

thus to deal properly with the case \% in which the % will be scanned and won't launch closing of the verbatim group.

```
  }%
}% end of \gmd@typesettexcode's definitions's group's \firstofone.
```

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.)

\gmd@spacewd
\gmd@ldspaceswd

```
\newlength\gmd@spacewd% to store the width of a (leading) ␣12.
\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

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

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

\gmd@dolspaces

```
\def\gmd@dolspaces{%
  \ifx\gmd@texcodespace\@let@token
    \@dsdirgfalse
    \afterfi{\settowidth{\gmd@spacewd}{\visiblespace}%
    \gmd@ldspaceswd=\z@
    \gmd@eatlspace}%
  \else\afterfi{% about this smart macro and other of its family see gmutils sec. 3.
    \par
    \gmd@typesettexcode}%
  \fi}
```

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

\gmd@eatlspace

```
\def\gmd@eatlspace#1{%
  \ifx\gmd@texcodespace#1%
    \advance\gmd@ldspaceswd␣by\gmd@spacewd% we don't \advance it \globally
          because the current group may be closed iff we meet % and then we'll
          won't indent the line anyway.
    \afteriffifi\gmd@eatlspace
  \else
    \if\code@delim\@nx#1%
      \gmd@ldspaceswd=\z@
      \gmd@continuenarration#1%
    \else␣\afterfifi{\gmd@typesettexcode#1}%
    \fi
  \fi}%
```

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

```
\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

```
\newcommand*\gmd@charbychar[1]{%
  \ifhmode\prevhmodegtrue
  \else\prevhmodegfalse
  \fi
  \if\code@delim\@nx#1%
    \def\next{% occurs when next a \hskip4.875pt is to be put
      \gmd@percenthack% to typeset % if a comment continues the codeline.
    \endgroup%
    \gmd@checkifEOLmixd}% to see if next is ^^M and then do \par.
  \else% i.e., we've not met the code delimiter
    \ifx\relax#1\def\next{%
      \endgroup}% special case of end of file thanks to \everyeof.
    \else
      \if\code@escape@char\@nx#1%
        \@dsdirgfalse% yes, just here not before the whole \if because then we
            would discard checking for DocStrip directives doable by the active
            % at the 'old macrocode' setting.
        \def\next{%
          \gmd@counttheline#1\scan@macro}%
      \else
        \def\next{%
          \gmd@EOLorcharbychar#1}%
      \fi
    \fi
  \fi\next}
```

\debug@special

```
\def\debug@special#1{%
  \ifhmode\special{color␣push␣gray␣0.#1}%
  \else\special{color␣push␣gray␣0.#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 thorougly. 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

```
\def\gmd@EOLorcharbychar#1{%
  \ifx\gmd@texcodeEOL#1%
    \if@newline
      \@newlinegfalse
    \fi
    \afterfi{#1}% here we print #1.
  \else% i.e., #1 is not a (very active) line end,
    \afterfi
    {%
\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 ^^M with no commentary.
  \fi}
```

\gmd@counttheline

```
\def\gmd@counttheline{%
  \ifvmode
    \if@newline
      \leavevmode
      \gmd@grefstep{codelinenum}\@newlinegfalse
      \hyperlabel@line
    \fi
    \printlinenumber
2868       \mark@envir
2869     \else% not vmode
2870       \if@newline
2872         \gmd@grefstep{codelinenum}\@newlinegfalse
2873         \hyperlabel@line
2874       \fi
2875     \fi}
```

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

\gmd@percenthack

```
2880 \def\gmd@percenthack{%
2881   \ifprevhmode\code@delim\aftergroup\space% We add a space after %, because I think it looks better. It's done \aftergroup to make the spaces possible after the % not to be typeset.
2885   \else\aftergroup\gmd@narrcheckifds@ne% remember that \gmd@precenthack is only called when we've the code delimiter and soon we'll close the verbatim group and right after \endgroup there waits \gmd@checkifEOLmixd.
2889   \fi}
```

\gmd@narrcheckifds@ne

```
2891 \def\gmd@narrcheckifds@ne#1{%
2892   \@dsdirgfalse\@ifnextchar<{%
2893     \@xa\gmd@docstripdirective\@gobble}{#1}}
```

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

\gmd@checkifEOL

```
2899 \def\gmd@checkifEOL{%
2900   \gmd@cpnarrline
2901   \everypar=\@xa{\@xa\@codetonarrskip% we add the macro that'll insert a vertical space if we leave the code and enter the narration.
2904     \the\gmd@preverypar}%
2905   \@ifnextchar{\gmd@textEOL}{%
2906     \@dsdirgfalse\par\ignorespaces}{\gmd@narrcheckifds}}%
```

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

\gmd@narrcheckifds

```
2909 \def\gmd@narrcheckifds{%
2910   \@dsdirgfalse\@ifnextchar<{%
2911     \@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

```
2917 \def\gmd@checkifEOLmixd{%
2918   \gmd@cpnarrline
2919   \everypar=\@xa{\@xa\@codetonarrskip\the\gmd@preverypar}%
2922   \@afternarrgfalse\@aftercodegtrue
2923   \ifhmode\@codeskipputgfalse\fi
2924   \@ifnextchar{\gmd@textEOL}{%
2925     {\raggedright\gmd@endpe\par}% without \raggedright this \par would be justified which is not appropriate for a long codeline that should be broken, e.g., 2919.
2928     \prevhmodegfalse
2929     \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                      \def\par{%
                            \ifhmode% (I added this \ifhmode as a result of a heavy debug.)
                              \@@par
                              \if@afternarr
                                \if@aftercode
                                  \if@codeskipput\else\gmd@codeskip2\@aftercodegfalse%
                                        \@codeskipputgtrue\fi
                                \else\gmd@nocodeskip2{naC}%
                                \fi
                              \else\gmd@nocodeskip2{naN}%
                              \fi
                              \prevhmodegfalse\gmd@endpe% when taken out of \ifhmode, this line
                                   caused some codeline numbers were typeset with \leftskip = 0.
                              \everypar=\@xa{%
                                \@xa\@codetonarrskip\the\gmd@preverypar}%
                              \let\par\@@par%
                            \fi}%
                          \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            \def\gmd@endpe{%
                        \ifprevhmode
                          \settexcodehangi%ndent
                          \leftskip=\CodeIndent
                        \else
                          \leftskip=\TextIndent
                          \hangindent=\z@
                          \everypar=\@xa{%
                            \@xa\@codetonarrskip\the\gmd@preverypar}%
                        \fi}
```

Numbering (or Not) of the Lines

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

```
                      \if@uresetlinecount% with uresetlinecount option...
                        \@relaxen\gmd@resetlinecount% ... we turn resetting the counter by \DocIn-
                              % put off...
\resetlinecountwith     \newcommand*\resetlinecountwith[1]{%
codelinenum               \newcounter{codelinenum}[#1]}% ... and provide a new declaration of the
                                counter.
                      \else% With the option turned off...
DocInputsCount          \newcounter{DocInputsCount}%
codelinenum             \newcounter{codelinenum}[DocInputsCount]% ... we declare the \DocInputs'
                              number counter andthe codeline counter to be reset with stepping of it.
\gmd@resetlinecount     \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.
```

```
\fi
```

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

\printlinenumber

```
\newcommand*\printlinenumber{%
  \leavevmode\llap{\rlap{\LineNumFont$\phantom{999}$\llap{%
        \thecodelinenum}}%
    \hskip\leftskip}}
```

\LineNumFont

```
\def\LineNumFont{\normalfont\tiny}
```

```
\if@linesnotnum\@relaxen\printlinenumber\fi
```

\hyperlabel@line

```
\newcommand*\hyperlabel@line{%
  \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.
  \else
    \raisebox{2ex}[1ex][\z@]{\gmhypertarget[clnum.%
      \HLPrefix\arabic{codelinenum}]{}}%
  \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

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

@codeskipput

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

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

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

```
\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
\gmd@nocodeskip

```
\if1␣1
  \renewcommand*\gmd@codeskip[1]{%
    \hbox{\rule{1cm}{3pt}␣#1!!!}}
  \renewcommand*\gmd@nocodeskip[2]{%
    \hbox{\rule{1cm}{0.5pt}␣#1:␣#2␣}}
\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

```
\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

```
\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

```
\long\def\@codetonarrskip{%
  \if@codeskipput\else
    \if@afternarr\gmd@nocodeskip4{iaN}\else
      \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.

```
        {\setbox0=\lastbox}%
```

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

```
        \gmd@codeskip4\@codeskipputgtrue
        \leftskip\TextIndent% this line is a patch against a bug-or-feature that
              in certain cases the narration \leftskip is left equal the code left-
              skip. (It happens when there're subsequent code lines after an inline
              comment not ended with an explicit \par.)
        \else\gmd@nocodeskip4{naC}%
      \fi%
    \fi
  \fi\@aftercodegfalse}
```

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

\@trimandstore
\@trimandstore@hash

```
\long\def\@trimandstore#1\@trimandstore{%
  \def\@trimandstore@hash{#1}%
  \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.
    \gmd@preverypar={}%
  \else
    \afterfi{\@xa\@trimandstore@ne\the\everypar\@trimandstore}%
  \fi}
```

\@trimandstore@ne
\trimmed@everypar

```
\long\def\@trimandstore@ne#1#2\@trimandstore{%
  \def\trimmed@everypar{#2}%
  \ifx\@codetonarrskip#1%
    \gmd@preverypar=\@xa{\trimmed@everypar}%
  \else
    \gmd@preverypar=\@xa{\the\everypar}%
  \fi}
```

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

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:

```
                   3136 \catcode`\^^B=\active% note we re\catcode ⟨char2⟩ globally, for the entire document.
                   3138 \foone{\obeylines}%
^^B                3139   {\def\QueerCharTwo{%
\QueerCharTwo      3140       \protected\def^^B##1^^M{%
                   3142         \ifhmode\unskip\space\ignorespaces\fi}}% It shouldn't be \  not
                                      to drive TEX into hmode.
                   3144   }
                   3146 \QueerCharTwo
                   3148 \AtBegInput{\@ifEOLactive{\catcode`\^^B\active}{}\QueerCharTwo}% We
                           repeat redefinition of ⟨char2⟩ at begin of the documenting input, because
                           doc.dtx suggests that some packages (namely inputenc) may re\catcode
                           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 countalllines option is enabled.

I also liked the doc's idea of comment[2] 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.

```
                   3170 \catcode`\^^A=\active% note we re\catcode ⟨char1⟩ globally, for the entire document.
                   3172 \foone\obeylines{%
^^A                3173   \def\QueerCharOne{%
\QueerCharOne      3174     \def^^A{%
                   3176       \bgroup\let\do\@makeother\dospecials\gmd@gobbleuntilM}}%
\gmd@gobbleuntilM  3177   \def\gmd@gobbleuntilM#1^^M{\egroup\ignorespaces^^M}%
                   3178 }
                   3180 \QueerCharOne
                   3182 \AtBegInput{\@ifEOLactive{\catcode`\^^A\active}\QueerCharOne}% see note
                           after line 3148.
```

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       3198 \def\StraightEOL{%
                   3199   \catcode`\^^M=5
                   3200   \catcode`\^^A=14
                   3201   \catcode`\^^B=14
                   3202   \def\^^M{\ }}
                   3210 \foone\obeylines{%
\QueerEOL          3211   \def\QueerEOL{%
                   3212     \catcode`\^^M=\active%
3213      \let^^M\gmd@textEOL%
3214      \catcode`\^^A=\active%
3215      \catcode`\^^B=\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.)
3218      \let\^^M=\gmd@bslashEOL}%
3231 }
```

To make `^^M` behave more like a 'normal' lineend I command it to add a $_{10}$ at first. It works but has one uwelcome 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   3245 \def\gmd@parfixclosingspace{{%
                          3246      \advance\parfillskip␣by-\gmd@closingspacewd\par}}
```

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       3250 \newskip\gmd@closingspacewd
\gmd@setclosingspacewd    3251 \newcommand*\gmd@setclosingspacewd{%
                          3252   \global\gmd@closingspacewd=\fontdimen2\font%
                          3253      plus\fontdimen3\font␣minus\fontdimen4\font\relax}
```

See also line 2663 to see what we do in the codeline case when no closing space is added.

And one more detail:

```
                          3259 \foone\obeylines{%
                          3260   \if␣1␣1%
\gmd@bslashEOL            3261      \protected\def\gmd@bslashEOL{\ \@xa\ignorespaces^^M}%
                          3262   }% of \foone. Note we interlace here \if with a group.
                          3263 \else%
\gmd@bslashEOL            3264   \protected\def\gmd@bslashEOL{%
                          3265      \ifhmode\unskip\fi\ \ignorespaces}
                          3267   \fi
```

The `\QueerEOL` declaration will `\let` it to `\^^M` to make `\^^M` behave properly. If this definition was ommitted, `\^^M` would just expand to `\ ` and thus not gobble the leading `%` of the next line leave alone typesetting the TeX code. I type `\ ` etc. instead of just `^^M` which adds a space itself because I take account of a possibility of redefining the `\ ` 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 `verbatim(*)` typeset its contents with the TeX code's indentation:

```
\@verbatim                3290 \gaddtomacro\@verbatim{\leftskip=\CodeIndent}
```

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

```
\check@percent            3294 \AtBegInput{\long\def\check@percent#1{%
                          3295      \gmd@cpnarrline% to count the verbatim lines and possibly print their num-
                                        bers. This macro is used only by the verbatim end of line.
                          3297      \@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

```
\def\AddtoPrivateOthers#1{%
  \@xa\def\@xa\doprivateothers\@xa{%
    \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.

\gm@verb@eol \verb@egroup

```
\begingroup
\obeylines%
\AtBegInput{\def\gm@verb@eol{\obeylines%
     \def^^M{\verb@egroup\@latex@error{%
          \@nx\verb␣ended␣by␣end␣of␣line}%
        \@ifEOLactive{^^M}{\@ehc}}}}%
\endgroup
```

Macros for Marking 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 \quotechar \encapchar \levelchar

```
\edef\actualchar{\string␣@}
\edef\quotechar{\string␣"}
\edef\encapchar{\xiiclub}
\edef\levelchar{\string␣!}
```

However, for the glossary, i.e., the change history, a special style is required, e.g., gmglo.ist, and the above macros are redefined by the \changes command due to gmglo.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 \code@escape@char

```
\def\CodeEscapeChar#1{%
  \begingroup
  \escapechar\m@ne
  \xdef\code@escape@char{\string#1}%
  \endgroup}
```

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

```
\CodeEscapeChar\\
```

As mentioned in doc, someone may have some chars $_{11}$ed.

\MakePrivateLetters

```
\@ifundefined{MakePrivateLetters}{%
  \def\MakePrivateLetters{\makeatletter\catcode`\*=11␣}}{}
```

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

And you may wish some special chars to be $_{12}$.

```
\MakePrivateOthers   3379 \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 ^^M, see line 4797. So let's define the list:

```
\doprivateothers   3383 \def\doprivateothers{\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 ommit some lines irrelevant with my approach.

```
\scan@macro   3396 \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 $_{12}$ so we may just have it printed during main scan char by char, i.e., in the lines 2846 and 2850.
So, we step the checksum counter first,

```
3402   \step@checksum% (see line 5990 for details),
```

Then, unlike in doc, we do *not* check if the scanning is allowed, because here it's always allowed and required.
Of course, I can imagine horrible perversities, but I don't think they should really be taken into account. Giving the letter a \catcode other than $_{11}$ surely would be one of those perversities. Therefore I feel safe to take the character a as a benchmark letter.

```
3411   \ifcat␣a\@nx#1%
3412     \quote@char#1%
3413     \xdef\macro@iname{\gmd@maybequote#1}% global for symmetry with line
               3431.
3415     \xdef\macro@pname{\string#1}% we'll print entire name of the macro later.
```

We \string it here and in the lines 3435 and 3447 to be sure it is whole $_{12}$ for easy testing for special indexentry formats, see line 4303 etc. Here we are sure the result of \string is $_{12}$ since its argument is $_{11}$.

```
3422     \afterfi{\@ifnextcat{a}{\gmd@finishifstar#1}{%
               \finish@macroscan}}%
3423   \else% #1 is not a letter, so we have just scanned a one-char cs.
```

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

```
3430     {\uccode`9=`#1%
3431       \uppercase{\xdef\macro@iname{9}}%
3432     }%
3433     \quote@char#1%
3434     \xdef\macro@iname{\gmd@maybequote\macro@iname}%
3435     \xdef\macro@pname{\xiistring#1}%
3436     \afterfi␣\finish@macroscan
3437   \fi}
```

The \xiistring macro, provided by gmutils, is used instead of original \string because we wish to get ␣$_{12}$('other' space).

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

\continue@macroscan

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

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 LaTeX it should finish the scanning of a cs name—we want to avoid scanning `\command*argum` as one cs.

\gmd@finishifstar

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

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

\quote@char

```
\def\quote@char#1{{\uccode`9=`#1% at first I took digit 1 for this \uccodeing
      but then #1 meant #⟨#1⟩ in \uppercase's argument, of course.
    \uppercase{%
      \gmd@ifinmeaning␣9\of␣\indexcontrols
      {\glet\gmd@maybequote\quotechar}%
      {\g@emptify\gmd@maybequote}%
    }%
  }}
```

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

```
\AtBeginDocument{\xdef\indexcontrols{%
    \bslash\levelchar\encapchar\actualchar\quotechar}}
```

\gmd@ifinmeaning
\gmd@in@@

```
\long\def␣\gmd@ifinmeaning#1\of#2#3#4{% explained in the text paragraph
          below.
  \long\def\gmd@in@@##1#1##2\gmd@in@@{%
    \ifx^^A##2^^A\afterfi{#4}%
    \else\afterfi{#3}%
    \fi}%
  \@xa\gmd@in@@#2#1\gmd@in@@}%
```

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

`\quote@char` sort of re`\catcode`s 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 3694).

\ifgmd@glosscs

```
3559 \newif\ifgmd@glosscs% we use this switch to keep the information whether a history entry is a cs or not.
```

\finish@macroscan

```
3562 \newcommand*\finish@macroscan{%
```

First we check if the current cs is not just being defined. The switch may be set true in line 3593

```
3565   \ifgmd@adef@cshook% if so, we throw it into marginpar and index as a def entry...
3567   \@ifundefined{gmd/iexcl/\macro@pname}{% ... if it's not excluded from indexing.
3569       \@xa\Code@MarginizeMacro\@xa{\macro@pname}%
3570       \@xa\@defentryze\@xa{\macro@pname}{1}}{}% here we declare the kind of index entry and define \last@defmark used by \changes
3572     \global\gmd@adef@cshookfalse% we falsify the hook that was set true just for this cs.
3574   \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.

```
3579   \if\verbatimchar\macro@pname% it's important that \verbatimchar comes before the macro's name: when it was reverse, the \tt cs turned this test true and left the \verbatimchar what resulted with '\+tt' typeset. Note that this test should turn true iff the scanned macro name shows to be the default \verb's delimiter. In such a case we give \verb another delimiter, namely $:
```

```
\im@firstpar     3586     \def\im@firstpar{[$]}%
\im@firstpar     3587   \else\def\im@firstpar{}\fi
                 3588   \@xa␣\index@macro\im@firstpar\macro@iname\macro@pname
                 3590   \maybe@marginpar\macro@pname
                 3591   \macro@pname
                 3592   \let\next\gmd@charbychar
                 3593   \gmd@detectors% for automatic detection of definitions. Defined and explained
                             in the next section. It redefines \next if detects a definition command amd
                             thus sets the switch of line 3562 true.
                 3598   \next
                 3599 }
```

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 3605 \def\maybe@marginpar#1{%
                 3607   \@ifundefined{gmd/2marpar/#1}{}{%
                 3608     \@xa\Text@Marginize\@xa{\bslash#1}% \expandafters
                             because the \Text@Marginize command applies \string to its argument.
                             % \macro@pname, which will be the only possible argument to \maybe-
                             % @marginpar, contains the macro's name without the escapechar so we
                             added it here.
                 3616     \@xa\g@relaxen\csname␣gmd/2marpar/#1\endcsname% we reset the switch.
                 3617   }}
```

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

Automatic detection of definitions

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

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

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

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

```
                 3655 \foone{\catcode`\*=11␣}
\@ifstarl        3656   {\def\@ifstarl#1{\@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

```
\outer\def\DeclareDefining{\begingroup
  \MakePrivateLetters
  \@ifstarl
    {\gdef\gmd@adef@defaulttype{text}\Declare@Dfng}%
    {\gdef\gmd@adef@defaulttype{cs}\Declare@Dfng}%
}
```

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

\Declare@Dfng

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

\Declare@Dfng@inner

\gmd@adef@currdef

```
\def\Declare@Dfng@inner#1#2{%
  \edef\gmd@resa{%
    \@nx\setkeys[gmd]{adef}{type=\gmd@adef@defaulttype}}%
  \gmd@resa
  {\escapechar\m@ne
    \xdef\gmd@adef@currdef{\string#2}%
  }%
  \gmd@adef@setkeysdefault
  \setkeys[gmd]{adef}{#1}%
  \@xa\gmd@ifinmeaning
    \csname␣gmd@detect@\gmd@adef@currdef\endcsname
    \of\gmd@detectors{}{%
      \@xa\gaddtomacro\@xa\gmd@detectors\@xa{%
        \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.
  \@xa\xdef\csname␣gmd@detectname@\gmd@adef@currdef\endcsname{%
    \gmd@adef@currdef}%
  \edef\gmu@tempa{% this \edef is to expand \gmd@adef@TYPE.
    \global\@nx\@namedef{gmd@detect@\gmd@adef@currdef}{%
      \@nx\ifx
        \@xa\@nx\csname␣gmd@detectname@\gmd@adef@currdef%
            \endcsname
        \@nx\macro@pname
        \@nx\n@melet{next}{gmd@adef@\gmd@adef@TYPE}%
        \@nx\n@melet{gmd@adef@currdef}{gmd@detectname@%
            \gmd@adef@currdef}%
      \@nx\fi}}%
  \gmu@tempa
3713   \SMglobal\StoreMacro*{gmd@detect@\gmd@adef@currdef}% we store the cs
          to allow its temporary discarding later.
3715 }
```

```
\gmd@adef@setkeysdefault   3718 \def\gmd@adef@setkeysdefault{%
3719   \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 3892.

```
star   3725 \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 4142 and therefore \newcounter{foo} occurring in the code will index both foo and \c@foo (as definition entries).

```
prefix   3734 \define@key[gmd]{adef}{prefix}[]{%
3735   \edef\gmd@resa{%
3736     \def\@xa\@nx\csname␣gmd@adef@prefix@\gmd@adef@currdef␣%
          \endcsname{%
3737       #1}}%
3738   \gmd@resa}
```

```
\gmd@KVprefdefault   3741 \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   3749 \define@key[gmd]{adef}{KVpref}[\gmd@KVprefdefault]{%
3750   \edef\gmd@resa{#1}%
3751   \ifx\gmd@resa\gmd@KVprefdefault
3752   \else
3753     \@namedef{gmd@adef@KVprefixset@\gmd@adef@currdef}{1}%
3754     \gmd@adef@setKV% whenever the KVpreffix is set (not default), the declared
          command is assumed to be keyvalish.
3756   \fi
3757   \edef\gmd@resa{#1}% because \gmd@adef@setKV redefined it.
3758   \edef\gmd@resa{%
3759     \def\@xa\@nx\csname␣gmd@adef@KVpref@\gmd@adef@currdef%
          \endcsname{%
3760       \ifx\gmd@resa\empty
3761       \else#1@\fi}}% as in xkeyval, if the KV prefix is not empty, we add @ to it.
3763   \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   3770 \define@key[gmd]{adef}{KVfam}[]{%
3771   \edef\gmd@resa{#1}%
3772   \@namedef{gmd@adef@KVfamset@\gmd@adef@currdef}{1}%
3773   \edef\gmd@resa{%
3774     \def\@xa\@nx\csname␣gmd@adef@KVfam@\gmd@adef@currdef%
          \endcsname{%
3775       \ifx\gmd@resa\empty
```

```
      \else#1@\fi}}%
  \gmd@resa
  \gmd@adef@setKV}% whenever the KVfamily is set, the declared command is assumed to be keyvalish.
```

type

```
\define@choicekey[gmd]{adef}{type}
  [\gmd@adef@typevals\gmd@adef@typenr]
  {% the list of possible types of defining commands
    def,
    newcommand,
    cs,% equivalent to the two above, covers all the cases of defining a cs, including the Plain TeX \new... and LaTeX \newlength.
    newenvironment,
    text,% equivalent to the one above, covers all the commands defining its first mandatory argument that should be text, \DeclareOption e.g.
    define@key,% special case of more arguments important; covers the xkeyval defining commands.
    dk,% a shorthand for the one above.
    DeclareOptionX,% another case of special arguments configuration, covers the xkeyval homonym.
    dox,% a shorthand for the one above.
    kvo% one of option defining commands of the kvoptions package by Heiko Oberdiek (a package available o CTAN in the oberdiek bundle).
  }
  {% In fact we collapse all the types just to four so far:
    \ifcase\gmd@adef@typenr% if def
      \gmd@adef@settype{cs}{0}%
    \or% when newcommand
      \gmd@adef@settype{cs}{0}%
    \or% when cs
      \gmd@adef@settype{cs}{0}%
    \or% when newenvironment
      \gmd@adef@settype{text}{0}%
    \or% when text
      \gmd@adef@settype{text}{0}%
    \or% when define@key
      \gmd@adef@settype{dk}{1}%
    \or% when dk
      \gmd@adef@settype{dk}{1}%
    \or% when DeclareOptionX
      \gmd@adef@settype{dox}{1}%
    \or% when dox
      \gmd@adef@settype{dox}{1}%
    \or% when kvo
      \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⟩].
    \fi}
```

\gmd@adef@settype
\gmd@adef@TYPE

```
\def\gmd@adef@settype#1#2{%
  \def\gmd@adef@TYPE{#1}%
  \ifnum1=#2␣% now we define (or not) a quasi-switch that fires for the keyvalish definition commands.
    \gmd@adef@setKV
  \fi}
```

\gmd@adef@setKV

```
\def\gmd@adef@setKV{%
  \edef\gmd@resa{%
    \def\@xa\@nx\csname␣gmd@adef@KV@\gmd@adef@currdef\endcsname{%
      1}%
  }%
  \gmd@resa}
```

We initialize the carrier of detectors:

```
\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

```
\newif\ifgmd@adef@cshook
```

\gmd@adef@cs

```
\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 280 in gmverb) and **??**nd here we switch it according to the type of detected definition.

\gmd@adef@text

```
\def\gmd@adef@text{\gdef\gmd@lbracecase{1}\gmd@charbychar}

\foone{%
  \catcode`\[\active
  \catcode`\<\active}
{%
```

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

\gmd@adef@dk

```
  \def\gmd@adef@dk{%
    \let[\gmd@adef@scanKVpref
    \catcode`\[\active
    \gdef\gmd@lbracecase{2}%
    \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.
    \gmd@adef@checklbracket}
```

The detector of xkeyval \DeclareOptionX:

\gmd@adef@dox

```
  \def\gmd@adef@dox{%
    \let[\gmd@adef@scanKVpref
    \let<\gmd@adef@scanDOXfam
    \catcode`[\active
    \catcode`<\active
    \gdef\gmd@lbracecase{1}%
    \gmd@adef@dfKVpref\gmd@KVprefdefault% We set the default values of the xkeyval prefix...
    \edef\gmd@adef@fam{\gmd@inputname}% ... and family.
    \gmd@adef@dofam
    \gmd@adef@checkDOXopts}%
}
```

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

\gmd@adef@checklbracket

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

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

\gmd@adef@checkDOXopts

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

\gmd@adef@scanKVpref

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

\gmd@adef@dfKVpref

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

\gmd@adef@scanDOXfam

```
\def\gmd@adef@scanDOXfam{%
  \ifnum12=\catcode`\>\relax
    \let\next\gmd@adef@scanfamoth
  \else
    \ifnum13=\catcode`\>\relax
      \let\next\gmd@adef@scanfamact
    \else
      \PackageError{gmdoc}{>␣neither␣`other'␣nor␣`active'!␣Make␣
          it
        `other'␣with␣\bslash␣AddtoPrivateOthers\bslash\>.}%
    \fi
  \fi
  \next}
```

\gmd@adef@scanfamoth

```
\def\gmd@adef@scanfamoth#1>{%
    \edef\gmd@adef@fam{\@gobble#1}% there is always \gmd@charbychar first.
    \gmd@adef@dofam
    <\gmd@adef@fam>%
  \gmd@charbychar}
```

```
\foone{\catcode`\>\active}
```

\gmd@adef@scanfamact

```
  {\def\gmd@adef@scanfamact#1>{%
```

```
      \edef\gmd@adef@fam{\@gobble#1}% there is always \gmd@charbychar
          first.
      \gmd@adef@dofam
      <\gmd@adef@fam>%
      \gmd@charbychar}%
  }
```

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

```
\def\gm@lbracehook{%
  \ifcase\gmd@lbracecase\relax
  \or% when 1
    \afterfi{%
      \gdef\gmd@lbracecase{0}%
      \gmd@adef@scanname}%
  \or% when 2—the first mandatory argument of two (\define@(...)key)
    \afterfi{%
    \gdef\gmd@lbracecase{3}%
     \gmd@adef@scanDKfam}%
  \or% when 3—the second mandatory argument of two (the key name).
    \afterfi{%
      \gdef\gmd@lbracecase{0}%
      \gmd@adef@scanname}%
  \fi}
```

\gmd@lbracecase

```
\def\gmd@lbracecase{0}% we initialize the hook caser.
```

And we define the inner left brace macros:

```
\foone{\catcode`\[1␣\catcode`\]2␣\catcode`\}12␣}
  [% 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

```
  \def\gmd@adef@scanDKfam#1}[%
    \edef\gmd@adef@fam[\@gobble#1]% there is always \gmd@charbychar first.
    \gmd@adef@dofam
    \gmd@adef@fam}%
  \gmd@charbychar]
```

\gmd@adef@scanname

```
  \def\gmd@adef@scanname#1}[%
    \@makeother\[%
    \@makeother\<%
```

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

```
    \gmd@adef@deftext[#1]%
    \@gobble#1}%
    \gmd@charbychar]
  ]
```

\gmd@adef@dofam

```
\def\gmd@adef@dofam{%
  \ifnum1=0\csname␣gmd@adef@KVfamset@\gmd@adef@currdef\endcsname
    \relax% a family declared with \DeclareDefining overrides the one currently scanned.
  \else
    \edef\gmu@resa{%
      \gdef\@xa\@nx
      \csname␣gmd@adef@KVfam@\gmd@adef@currdef\endcsname
      {\ifx\gmd@adef@fam\empty
        \else\gmd@adef@fam␣@%
        \fi}}%
    \gmu@resa
  \fi}
```

\gmd@adef@deftext

```
\def\gmd@adef@deftext#1{%
  \edef\macro@pname{\@gobble#1}% we gobble \gmd@charbychar, cf. above.
  \@xa\Text@Marginize\@xa{\macro@pname}%
  \gmd@adef@indextext
  \edef\gmd@adef@altindex{%
    \csname␣gmd@adef@prefix@\gmd@adef@currdef␣\endcsname}%
```

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

```
  \ifnum1=0\csname␣gmd@adef@KV@\gmd@adef@currdef␣\endcsname\relax% The
        CS \gmd@adef@KV@⟨def. command⟩ is defined {1} (so \ifnum gets 1=01%
        \relax—true) iff ⟨def. command⟩ is a keyval definition. In that case we check
        for the KVprefix and KVfamily. (Otherwise \gmd@adef@KV@⟨def. command⟩
        is undefined so \ifnum gets 1=0\relax—false.)
    \edef\gmd@adef@altindex{%
      \gmd@adef@altindex
      \csname␣gmd@adef@KVpref@\gmd@adef@currdef␣\endcsname}%
    \edef\gmd@adef@altindex{%
      \gmd@adef@altindex
      \csname␣gmd@adef@KVfam@\gmd@adef@currdef␣\endcsname}%
  \fi
  \ifx\gmd@adef@altindex\empty
  \else% we make another index entry of the definiendum with prefix/KVheader.
    \edef\macro@pname{\gmd@adef@altindex\macro@pname}%
    \gmd@adef@indextext
  \fi}
```

\gmd@adef@indextext

```
\def\gmd@adef@indextext{%
  \@xa\@defentryze\@xa{\macro@pname}{0}% declare the definiendum has to have a definition entry and in the changes history should appear without backslash.
  \gmd@doindexingtext% redefine \do to an indexing macro.
  \@xa\do\@xa{\macro@pname}}
```

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

Default defining commands

Some commands are easy to declare as defining:

```
\DeclareDefining[star=false]\def
```

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 occurence of \def off (only the next one).

```
\UnDef                      \def\UnDef{%
                              \gdef\gmd@detect@def{%
                                \ifx\gmd@detectname@def\macro@pname
                                  \def\next{\SMglobal\RestoreMacro\gmd@detect@def}%
                                \fi}%
                              }

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

\HideDef                    \def\HideDef{\HideDefining\def\relaxen\UnDef}
\relaxen
\ResumeDef                  \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                   \DeclareDefining[star=false]\newcount
\newdimen                   \DeclareDefining[star=false]\newdimen
\newskip                    \DeclareDefining[star=false]\newskip
                            \DeclareDefining[star=false]\newif
\newtoks                    \DeclareDefining[star=false]\newtoks
\newbox                     \DeclareDefining[star=false]\newbox
\newread                    \DeclareDefining[star=false]\newread
\newwrite                   \DeclareDefining[star=false]\newwrite
\newlength                  \DeclareDefining[star=false]\newlength

                            \DeclareDefining\newcommand
\renewcommand               \DeclareDefining\renewcommand
                            \DeclareDefining\providecommand
\DeclareRobustCommand       \DeclareDefining\DeclareRobustCommand
\DeclareTextCommand         \DeclareDefining\DeclareTextCommand
\DeclareTextCommandDefault  \DeclareDefining\DeclareTextCommandDefault

                            \DeclareDefining*\newenvironment
                            \DeclareDefining*\renewenvironment
\DeclareOption              \DeclareDefining*\DeclareOption

                            %\DeclareDefining*\@namedef

\newcounter                 \DeclareDefining*[prefix=\bslash␣c@]\newcounter% this prefix provides indexing also \c@⟨counter⟩.

\define@key                 \DeclareDefining[type=dk,␣prefix=\bslash]\define@key
\define@boolkey             \DeclareDefining[type=dk,␣prefix=\bslash␣if]\define@boolkey% the alternate index entry will be \if⟨KVpref⟩@⟨KVfam⟩@⟨key name⟩
\define@choicekey           \DeclareDefining[type=dk,␣prefix=\bslash]\define@choicekey

\DeclareOptionX             \DeclareDefining[type=dox,␣prefix=\bslash]\DeclareOptionX% the alternate 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             \newcommand*\DeclareDOXHead[2][\gmd@KVprefdefault]{%
                              \csname␣DeclareDefining\endcsname
                              [type=dox,␣prefix=\bslash,␣KVpref=#1,␣KVfam=#2]%
```

\DeclareOptionX

```
  \DeclareOptionX
}
```

An example:

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

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

ChneOelze

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

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

Now the Heiko Oberdiek's kvoptions package option definitions:

\DeclareStringOption
\DeclareBoolOption
DeclareComplementaryOption
\DeclareVoidOption

```
\DeclareDefining[type=kvo,␣prefix=\bslash,␣KVpref=]%
     \DeclareStringOption
\DeclareDefining[type=kvo,␣prefix=\bslash,␣KVpref=]%
     \DeclareBoolOption
\DeclareDefining[type=kvo,␣prefix=\bslash,␣KVpref=]%
     \DeclareComplementaryOption
\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:

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

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

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

Suspending ('hiding') and resuming detection

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

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

\HideAllDefining

```
\def\HideAllDefining{%
  \ifnum0=0\csname␣gmd@adef@allstored\endcsname
    \SMglobal\StoreMacro\gmd@detectors
    \global\@namedef{gmd@adef@allstored}{1}%
  \fi
  \global\emptify\gmd@detectors}% we make the carrier \empty not \relax
        to be able to declare new defining command in the scope of \HideAll...
```

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

\ResumeAllDefining

```
\def\ResumeAllDefining{%
  \ifnum1=0\csname␣gmd@adef@allstored\endcsname\relax
    \SMglobal\RestoreMacro\gmd@detectors
    \SMglobal\RestoreMacro\UnDef
    \global\@namedef{gmd@adef@allstored}{0}%
  \fi}
```

Note that \ResumeAllDefining discards the effect of any \DeclareDefining that could have occured 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

```
\def\HideDefining{\begingroup
  \MakePrivateLetters
  \Hide@Dfng}
```

\Hide@Dfng

```
\def\Hide@Dfng#1{%
  \escapechar\m@ne
  \gn@melet{gmd@detect@\string#1}{relax}%
  \gn@melet{gmd@detect@\string#1*}{relax}%
  \ifx\def#1\global\relaxen\UnDef\fi
  \endgroup}
```

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 css.

\ResumeDefining

```
\def\ResumeDefining{\begingroup
  \MakePrivateLetters
  \gmd@ResumeDfng}
```

\gmd@ResumeDfng

```
\def\gmd@ResumeDfng#1{%
  \escapechar\m@ne
  \SMglobal\RestoreMacro*{gmd@detect@\string#1}%
  \SMglobal\RestoreMacro*{gmd@detect@\string#1*}%
  \endgroup}
```

Indexing of css

The inner macro indexing macro. #1 is the \verb's delimiter; #2 is assumed to be the macro's name with MakeIndex-control chars quoted. #3 is a macro storing the $_{12}$ macro's name, usually \macro@pname, built with \stringing every char in lines 3415, 3435 and 3447. #3 is used only to test if the entry should be specially formatted.

\index@macro

```
\newcommand*\index@macro[3][\verbatimchar]{{%
    \@ifundefined{gmd/iexcl/#3}%
    {% #3 is not excluded from index
      \@ifundefined{gmd/defentry/#3}%
      {% #3 is not def entry
```

```
                        \@ifundefined{gmd/usgentry/#3}%
                        {% #3 is not usg entry
                          \edef\kind@fentry{\CommonEntryCmd}}%
                        {% #3 is usg entry
\kind@fentry              \def\kind@fentry{UsgEntry}%
                          \un@usgentryze{#3}}%
                      }%
                      {% #3 is def entry
\kind@fentry            \def\kind@fentry{DefEntry}%
                        \un@defentryze{#3}%
                      }% of gmd/defentry/ test's 'else'
                      \if@pageindex\@pageinclindexfalse\fi% should it be here or there?
                          Definitely here because we'll wish to switch the switch with a declaration.
                      \if@pageinclindex
                        \edef\gmu@tempa{gmdindexpagecs{\HLPrefix}{\kind@fentry}{%
                            \EntryPrefix}}%
                      \else
                        \edef\gmu@tempa{gmdindexrefcs{\HLPrefix}{\kind@fentry}{%
                            \EntryPrefix}}%
                      \fi
                      \edef\gmu@tempa{\IndexPrefix#2\actualchar%
                        \quotechar\bslash␣verb*#1\quoted@eschar#2#1% The last macro in
                            this line usually means the first two, but in some cases it's redefined
                            to be empty (when we use \index@macro to index not a cs).
                        \encapchar\gmu@tempa}%
                      \@xa\special@index\@xa{\gmu@tempa}% We give the indexing macro the
                          argument expanded so that hyperref may see the explicit encapchar
                          in order not to add its own encapsulation of |hyperpage when the
                          (default) hyperindex=true option is in force. (After this setting the
                          \edefs in the above may be changed to \defs.)
                    }{}% closing of gmd/iexcl/ test.
                    }}
\un@defentryze  \def\un@defentryze#1{%
                  \@xa\g@relaxen\csname␣gmd/defentry/#1\endcsname
                  \ifx\gmd@detectors\empty
                    \g@relaxen\last@defmark
                  \fi}% the last macro (assuming \fi is not a macro :-) is only used by \changes. If
                      we are in the scope of automatic detection of definitions, we want to be able
                      not to use \Define but write \changes after a definition and get proper entry.
                      Note that in case of automatic detection of definitions \last@defmark's
                      value keeps until the next definition.
\un@usgentryze  \def\un@usgentryze#1{%
                  \@xa\g@relaxen\csname␣gmd/usgentry/#1\endcsname}
                \@emptify\EntryPrefix% this macro seems to be obsolete now (v0.98d).
```

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

```
\if@pageinclindex  \newif\if@pageinclindex
```

```
\quoted@eschar  \newcommand*\quoted@eschar{\quotechar\bslash}% we'll redefine it when in-
                    dexing an environment.
```

Let's initialize \IndexPrefix

```
\IndexPrefix     4372 \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.

```
                 4381 \if@linesnotnum\@pageindextrue\fi
                 4382 \AtBeginDocument{%
                 4383   \if@pageindex
\gmdindexrefcs   4384     \def\gmdindexrefcs#1#2#3#4{\csname#2\endcsname{\hyperpage{#4}}}%
                               in the page case we gobble the third argument that is supposed to be the
                               entry prefix.
                 4387     \let\gmdindexpagecs=\gmdindexrefcs
                 4388   \else
\gmdindexrefcs   4391     \def␣\gmdindexrefcs#1#2#3#4{\gmiflink[clnum.#4]{%
                 4392         \csname#2\endcsname{#4}}}%
\gmdindexpagecs  4393     \def␣\gmdindexpagecs#1#2#3#4{\hyperlink{page.#4}{%
                 4394         \csname#2\endcsname{\gmd@revprefix{#3}#4}}}%

\gmd@revprefix   4396     \def\gmd@revprefix#1{%
\gmu@tempa       4397       \def\gmu@tempa{#1}%
                 4398       \ifx\gmu@tempa\@empty␣p.\,\fi}

\HLPrefix        4400     \providecommand*\HLPrefix{}% it'll be the hypertargets names' prefix in
                               multi-docs. Moreover, it showed that if it was empty, hyperref saw du-
                               plicates of the hyper destinations, which was perfectly understandable
                               (codelinenum.123 made by \refstepcounter and codelinenum.123
                               made by \gmhypertarget). But since v0.98 it is not a problem any-
                               more because during the automatic \hypertargeting the lines are la-
                               beled clnum.⟨number⟩. When \HLPrefix was defined as dot, MakeIndex
                               rejected the entries as 'illegal page number'.
                 4412   \fi}
```

The definition is postponed till \begin{document} because of the \PageIndex declaration (added for doc-compatibility), see line 7182.

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        4427 \def\DefEntry#1{\underline{#1}}
\UsgEntry        4428 \def\UsgEntry#1{\textit{#1}}
```

The third option will be just \relax by default:

```
\CommonEntryCmd  4430 \def\CommonEntryCmd{relax}
```

In line 4307 it's \edefed 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 $_{12}$ed name of the macro given it as the argument into proper macro to be \ifxed in lines 4303 and 4305 respectively.

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

```
\DefIndex          4446 \outer\def\DefIndex{\begingroup
                   4447   \MakePrivateLetters
                   4448   \@ifstarl{\MakePrivateOthers\Code@DefIndexStar}{%
                               \Code@DefIndex}}
```

```
\Code@DefIndex     4453 \long\def\Code@DefIndex#1{\endgroup{%
                   4454     \escapechar\m@ne% because we will compare the macro's name with a string
                                 without the backslash.
                   4456     \@defentryze{#1}{1}}}
```

```
\Code@DefIndexStar 4460 \long\def\Code@DefIndexStar#1{%
                   4461   \endgroup
                   4462   \addto@estoindex{#1}%
                   4463   \@defentryze{#1}{0}}
```

```
\gmd@justadot      4465 \def\gmd@justadot{.}
```

```
\@defentryze       4467 \long\def\@defentryze#1#2{%
                   4468   \@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      4471   \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 \\).
                   4476   \@xa\gdef\csname␣gmd/isaCS/\last@defmark\endcsname{#2}}% #2 is ei-
                               ther 0 or 1. It is the information whether this entry is a cs or not.
```

```
\@usgentryze       4480 \long\def\@usgentryze#1{%
                   4481   \@xa\let\csname␣gmd/usgentry/\string#1\endcsname\gmd@justadot}
```

Initialize \envirs@toindex

```
                   4484 \@emptify\envirs@toindex
```

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

```
\CodeUsgIndex      4487 \outer\def\CodeUsgIndex{\begingroup
                   4488   \MakePrivateLetters
                   4489   \@ifstarl{\MakePrivateOthers\Code@UsgIndexStar}{%
                               \Code@UsgIndex}}
```

The * possibility is for marking environments etc.

```
\Code@UsgIndex     4492 \long\def\Code@UsgIndex#1{\endgroup{%
                   4493     \escapechar\m@ne
                   4494     \global\@usgentryze{#1}}}
```

```
\Code@UsgIndexStar 4497 \long\def\Code@UsgIndexStar#1{%
                   4498   \endgroup
                   4499   \addto@estoindex{#1}%
                   4500   \@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   4504 \outer\def\CodeCommonIndex{\begingroup
```

```
                       4505   \MakePrivateLetters
                       4506   \@ifstarl{\MakePrivateOthers\Code@CommonIndexStar}{%
                                   \Code@CommonIndex}}
```

```
\Code@CommonIndex      4509 \long\def\Code@CommonIndex#1{\endgroup}
```

```
\Code@CommonIndexStar  4512 \long\def\Code@CommonIndexStar#1{%
                       4513   \endgroup\addto@estoindex{#1}}
```

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

```
\text@indexmacro       4518 \long\def\text@indexmacro#1{%
                       4519   {\escapechar\m@ne␣\xdef\macro@pname{\xiistring#1}}%
                       4520   \@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.
                       4524   \if\verbatimchar\macro@pname
\im@firstpar           4525     \def\im@firstpar{[$]}%
\im@firstpar           4526   \else\def\im@firstpar{}%
                       4527   \fi
                       4528   {\do@properindex% see line 4866.
                       4529     \@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           4534 \def\quote@mname{%
\macro@iname           4535   \def\macro@iname{}%
                       4536   \quote@charbychar}
```

```
\quote@charbychar      4539 \def\quote@charbychar#1{%
                       4540   \if\relax#1% finish quoting when you meet \relax or:
                       4541   \else
                       4542     \quote@char#1%
                       4543     \xdef\macro@iname{\macro@iname␣\gmd@maybequote#1}%
                       4544     \afterfi\quote@charbychar
                       4545   \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          4551 \def\TextUsgIndex{\begingroup
                       4552   \MakePrivateLetters
                       4553   \@ifstarl{\MakePrivateOthers\Text@UsgIndexStar}{%
                                   \Text@UsgIndex}}
```

```
\Text@UsgIndex         4556 \long\def\Text@UsgIndex#1{%
                       4557   \endgroup\@usgentryze#1%
                       4558   \text@indexmacro#1}
```

```
\Text@UsgIndexStar     4561 \long\def\Text@UsgIndexStar#1{\endgroup\@usgentryze{#1}%
                       4562   \text@indexenvir{#1}}
```

```
\text@indexenvir       4564 \long\def␣\text@indexenvir#1{%
                       4565   \edef\macro@pname{\xiistring#1}%
                       4566   \if\bslash\@xa\@firstofmany\macro@pname\@@nil% if \stringed #1 be-
                                   gins with a backslash, we will gobble it to make MakeIndex not see it.
4569      \edef\gmu@tempa{\@xa\@gobble\macro@pname}%
4570      \@tempswatrue
4571    \else
4572      \let\gmu@tempa\macro@pname
4573      \@tempswafalse
4574    \fi
4575    \@xa\quote@mname\gmu@tempa\relax% we process \stinged #1 char by char
            and quote MakeIndex controls. \relax is the iterating macro's stopper. The
            quoted \stringed #1 shall be the meaning of \macro@iname.
4579    {\if@tempswa
\quoted@eschar
4580      \def\quoted@eschar{\quotechar\bslash}%
4581      \else\@emptify\quoted@eschar\fi% we won't print any backslash before
              an environment's name, but we will before a cs's name.
4583      \do@properindex% see line 4866.
4584      \index@macro\macro@iname\macro@pname}}
```

```
\TextCommonIndex
4586  \def\TextCommonIndex{\begingroup
4587    \MakePrivateLetters
4588    \@ifstarl{\MakePrivateOthers\Text@CommonIndexStar}{%
            \Text@CommonIndex}}
```

```
\Text@CommonIndex
4591  \long\def\Text@CommonIndex#1{\endgroup
4592    \text@indexmacro#1}
```

```
\Text@CommonIndexStar
4595  \long\def\Text@CommonIndexStar#1{\endgroup
4596    \text@indexenvir{#1}}
```

As you see in the lines 4314 and 4310, the markers of special formatting are reset after first use.

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

```
\CodeMarginize
4603  \outer\def\CodeMarginize{\begingroup
4604    \MakePrivateLetters
4605    \@ifstarl
4606      {\MakePrivateOthers\egCode@MarginizeEnvir}
4607      {\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
4613  \long\def\egCode@MarginizeMacro#1{\endgroup
4614    \Code@MarginizeMacro#1}
```

```
\Code@MarginizeMacro
4617  \long\def\Code@MarginizeMacro#1{{\escapechar\m@ne
4618      \@xa\glet\csname␣gmd/2marpar/\string#1\endcsname\gmd@justadot
4620    }}
```

```
\egCode@MarginizeEnvir
4623  \long\def\egCode@MarginizeEnvir#1{\endgroup
4624    \Code@MarginizeEnvir{#1}}
```

```
\Code@MarginizeEnvir
4627  \long\def\Code@MarginizeEnvir#1{\addto@estomarginpar{#1}}
```

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

Here it is:

```
\mark@envir
4633  \def\mark@envir{%
4634    \ifx\envirs@tomarginpar\@empty
4635   \else
4636     \let\do\Text@Marginize
4637     \envirs@tomarginpar%
4638     \g@emptify\envirs@tomarginpar%
4639   \fi
4640   \ifx\envirs@toindex\@empty
4641   \else
4642     \gmd@doindexingtext
4643     \envirs@toindex
4644     \g@emptify\envirs@toindex%
4645   \fi}
```

\gmd@doindexingtext

```
4647 \def\gmd@doindexingtext{%
4648   \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 4069.
4652     \if\bslash\@firstofmany##1\@@nil% if ##1 begins with a backslash, we will gobble it for MakeIndex not see it.
4655     \edef\gmd@resa{\@gobble##1}%
4656     \@tempswatrue
4657     \else
4658     \edef\gmd@resa{##1}\@tempswafalse
4659     \fi
4660     \@xa\quote@mname\gmd@resa\relax% see line 4575 & subs. for commentary.
4662     {\if@tempswa
```

\quoted@eschar

```
4663       \def\quoted@eschar{\quotechar\bslash}%
4664       \else\@emptify\quoted@eschar\fi
4665       \index@macro\macro@iname{##1}}}%
4666 }
```

One very important thing: initialisation of the list macros:

```
4670 \@emptify\envirs@tomarginpar
4671 \@emptify\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

```
4678 \outer\def\Define{\begingroup
4679   \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 `\catcode`s for the next token):

```
4684   \@ifstarl{\MakePrivateOthers\Code@DefEnvir}{\Code@DefMacro}}
```

\CodeUsage

```
4686 \outer\def\CodeUsage{\begingroup
4687   \MakePrivateLetters
4688   \@ifstarl{\MakePrivateOthers\Code@UsgEnvir}{\Code@UsgMacro}}
```

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

\Code@DefMacro

```
4691 \long\def\Code@DefMacro#1{%
4692   \Code@DefIndex#1% we use the internal macro; it'll close the group.
4693   \Code@MarginizeMacro#1}
```

\Code@UsgMacro

```
4696 \long\def\Code@UsgMacro#1{%
4697   \Code@UsgIndex#1% here also the internal macro; it'll close the group
```

```
  \Code@MarginizeMacro#1}
```

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

\codeline@wrindex

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

\codeline@glossary

```
\def\codeline@glossary#1{% It doesn't need to establish a group since it is always called in a group.
  \if@pageinclindex
    \edef\gmu@tempa{gmdindexpagecs{\HLPrefix}{relax}{%
        \EntryPrefix}}%
  \else
    \edef\gmu@tempa{gmdindexrefcs{\HLPrefix}{relax}{\EntryPrefix}}%
        % relax stands for the formatting command. But we don't want to do
        anything special with the change history entries.
  \fi
  \protected@edef\gmu@tempa{%
    \@nx\protected@write\@nx\@glossaryfile{}%
    {\string\glossaryentry{#1\encapchar\gmu@tempa}%
    {\HLPrefix\number\c@codelinenum}}}%
  \gmu@tempa
}
```

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

```
\AtBeginDocument{%
  \if@pageindex
    \let\special@index=\index
    \let\gmd@glossary\glossary
  \else
    \let\special@index=\codeline@wrindex
    \let\gmd@glossary\codeline@glossary
  \fi}% postponed till \begin{document} with respect of doc-like declarations.
```

And in case we don't want to index:

\gag@index

```
\def\gag@index{\let\index=\@gobble
  \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:

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

\ungag@index

```
\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 containig the cs to be marked. We separate indexing from marginizing to leave a possibility of doing only one of those things.

\Code@DefEnvir

```
\long\def\Code@DefEnvir#1{%
  \endgroup
  \addto@estomarginpar{#1}%
  \addto@estoindex{#1}%
  \@defentryze{#1}{0}}
```

\Code@UsgEnvir

```
\long\def\Code@UsgEnvir#1{%
  \endgroup
  \addto@estomarginpar{#1}%
  \addto@estoindex{#1}%
  \@usgentryze{#1}}
```

\addto@estomarginpar

```
\long\def\addto@estomarginpar#1{%
  \edef\gmu@tempa{\@nx\do{\xiistring#1}}% we \string the argument to allow it to be a control sequence.
  \@xa\addtomacro\@xa\envirs@tomarginpar\@xa{\gmu@tempa}}
```

\addto@estoindex

```
\long\def\addto@estoindex#1{%
  \edef\gmu@tempa{\@nx\do{\xiistring#1}}
  \@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 css appearing explicitly and the latter for the strings and css appearing implicitly.

\TextUsage

```
\def\TextUsage{\begingroup
  \MakePrivateLetters
  \@ifstarl{\MakePrivateOthers\Text@UsgEnvir}{\Text@UsgMacro}}
```

\Text@UsgMacro

```
\long\def\Text@UsgMacro#1{%
  \endgroup{\tt\xiistring#1}%
  \Text@Marginize#1%
  \begingroup\Code@UsgIndex#1% we declare the kind of formatting of the entry.
  \text@indexmacro#1}
```

\Text@UsgEnvir

```
\long\def\Text@UsgEnvir#1{%
  \endgroup{\tt\xiistring#1}%
  \Text@Marginize{#1}%
  \@usgentryze{#1}% we declare the 'usage' kind of formatting of the entry and index the sequence #1.
  \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

```
\def\TextMarginize{\begingroup
  \MakePrivateLetters
  \@ifstarl{\MakePrivateOthers\egText@Marginize}{%
      \egText@Marginize}}
```

\egText@Marginize

```
\long\def\egText@Marginize#1{\endgroup
  \Text@Marginize#1}
```

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

```
\if@marginparsused
  \reversemarginpar
  \marginparpush\z@
  \marginparwidth8pc\relax
```

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

\gmdmarginpar

```
 \long\def\gmdmarginpar#1{%
   \marginpar{\raggedleft\strut
     \hskip0ptplus100ptminus100pt%
     #1}}%
```

\gmdmarginpar

```
\else
 \long\def\gmdmarginpar#1{}%
\fi
```

\Text@Marginize

```
\long\def\Text@Marginize#1{%
 \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 gmdocc class.)

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

If we pront also the narration lines' numbers, then the index entries for css and environments marked in the commentary should have codeline numbers not page numbers and that is \let in line 4737. 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

```
\def\do@properindex{%
 \if@printalllinenos\else
   \@pageinclindextrue
   \let\special@index=\index
 \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*

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

```
\newenvironment*{macrocode}{%
 \if@codeskipput\else\par\addvspace\CodeTopsep%
      \@codeskipputgtrue\fi
 \QueerEOL}%
 {\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                  4923 \let\oldmc\macrocode
                       4924 \let\endoldmc\endmacrocode
oldmc*                 4926 \n@melet{oldmc*}{macrocode*}
                       4927 \n@melet{endoldmc*}{endmacrocode*}
```

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

```
                       4931 \addtomacro\oldmc{\@oldmacrocode@launch}%
                       4932 \@xa\addtomacro\csname␣oldmc*\endcsname{%
                       4933   \@oldmacrocode@launch}

\@oldmacrocode@launch  4936 \def\@oldmacrocode@launch{%
                       4937   \emptify\gmd@textEOL% to disable it in \gmd@docstripdirective launched
                                   within the code.
                       4939   \gmd@ctallsetup
                       4940   \glet\stored@code@delim\code@delim
                       4941   \@makeother\^^B\CodeDelim\^^B%
                       4942   \ttverbatim␣\gmd@DoTeXCodeSpace%
                       4943   \@makeother\|% because \ttverbatim doesn't do that.
                       4944   \MakePrivateLetters% see line 3370.
                       4946   \docstrips@percent␣\@makeother\>%
```

sine qua non of the automatic delimiting is replacing possible $*_{12}$in 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.

```
                       4953   \@xa\gmd@currenvxistar\@currenvir*\relax
                       4954   \@oldmacrocode}

                       4956 \foone{\catcode`*11␣}
\gm@xistar             4957 {\def\gm@xistar{*}}

\gmd@currenvxistar     4959 \def\gmd@currenvxistar#1*#2\relax{%
                       4960   \edef\@currenvir{#1\if*#2\gm@xistar\fi}}
```

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

```
                       4970 \foone{%
                       4971 \catcode`[=1␣\catcode`]=2
                       4972 \catcode`\{=\active␣\@makeother\}
                       4973 \@makeother\^^B
                       4974 \catcode`/=0␣\catcode`\\=\active
```

[8] Richard Strauss after Oscar Wilde, *Salome*.

```
\catcode`&=14␣\catcode`*=11
\catcode`\%=\active␣\obeyspaces}&␣%
[& here the \foone's second pseudo-argument begins
```

\@oldmacrocode

```
/def/@oldmacrocode[&
/bgroup/let␣=/relax& to avoid writing /@nx   four times.
/xdef/oldmc@def[&
/def/@nx/oldmc@end####1/@nx%␣␣␣␣/@nx\end&
/@nx{/@currenvir}[&
####1^^B/@nx/end[/@currenvir]/@nx/gmd@oldmcfinis]]&
/egroup& now \oldmc@edef is defined to have one parameter delimited with
      \end{⟨current env.'s name⟩}
/oldmc@def&
/oldmc@end]&
]
```

```
\def\gmd@oldmcfinis{%
  \@xa\CodeDelim\stored@code@delim
  \gmd@mchook}% see line 6967
```

```
\def\OldMacrocodes{%
  \let\macrocode\oldmc
  \n@melet{macrocode*}{oldmc*}}
```

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

```
\foone{\catcode`\%\active}
{\def\docstrips@percent{\catcode`\%\active
    \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

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

```
\newenvironment{macro}{%
  \@tempskipa=\MacroTopsep
  \if@codeskipput\advance\@tempskipa␣by-\CodeTopsep\fi
  \par\addvspace{\@tempskipa}\@codeskipputgtrue
  \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.)
  \gmd@ifonetoken\Hybrid@DefMacro\Hybrid@DefEnvir}%
  {\par\addvspace\MacroTopsep\@codeskipputgtrue}
```

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 `macro`s 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 to use this environment for marking implicit definitions, define the star version:

```
\@namedef{macro*}{\let\gmd@ifonetoken\@secondoftwo\macro}
\@xa\let\csname␣endmacro*\endcsname\endmacro
```

Note that `macro` and `macro*` have the same effect for more-than-one-token arguments thanks to `\gmd@ifonetoken`'s meaning inside unstarred `macro` (it checks whether the argument is one-token and if it isn't, `\gmd@ifonetoken` 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`
`\gmu@tempa`

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

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

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

`\gmd@ifonetoken`
`\gmu@tempb`

```
\def\gmd@ifonetoken#1#2#3{%
  \def\gmu@tempb{#3}% We hide #3 from TEX in case it's \if... or so. \gmu@tempa
        is used in \gmd@ifsingle.
  \gmd@ifsingle#3\@@nil
    \afterfi{\@xa#1\gmu@tempb}%
  \else
    \edef\gmu@tempa{\@xa\string\gmu@tempb}%
    \afterfi{\@xa#2\@xa{\gmu@tempa}}%
  \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`

```
\long\def\Hybrid@DefMacro#1{%
  \Code@DefIndex{#1}% this macro closes the group opened by \macro.
  \Text@MarginizeNext{#1}}
```

`\Hybrid@DefEnvir`

```
\long\def\Hybrid@DefEnvir#1{%
  \Code@DefIndexStar{#1}% this macro also closes the group begun by \macro.
  \Text@MarginizeNext{#1}}
```

`\Text@MarginizeNext`

```
\long\def\Text@MarginizeNext#1{%
```

```
  \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 `\relax`es itself after first use.

\gmd@evpaddonce

```
\long\def\gmd@evpaddonce#1{%
  \stepnummacro\gmd@oncenum
  \@xa\long\@xa\edef%
    \csname␣gmd/evp/NeuroOncer\gmd@oncenum\endcsname{%
      \@nx\g@relaxen
      \csname␣gmd/evp/NeuroOncer\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 un-
          expandable. That's why it passes \edef and is able to be assigned.
  \@xa\addtomacro\csname␣gmd/evp/NeuroOncer\gmd@oncenum%
      \endcsname{#1}%
  \@xa\addto@hook\@xa\everypar\@xa{%
    \csname␣gmd/evp/NeuroOncer\gmd@oncenum\endcsname}%
}
```

```
\nummacro\gmd@oncenum% We store the number uniquifying the auxiliary macro in
      a macro to save count registers (cf. gmutils sec. To Save Precious Count Regis-
      ters).
```

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.

```
\@xa\let\@xa\environment\csname␣macro*\endcsname
\@xa\let\@xa\endenvironment\csname␣endmacro*\endcsname
```

Index Exclude List

We want some css not to be indexed, e.g., the LATEX internals and TEX primitives.

doc takes `\index@excludelist` to be a `\toks` register to store the list of expelled css. 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

```
\def\DoNotIndex{\bgroup\MakePrivateLetters\DoNot@Index}
```

\DoNot@Index

```
\long\def\DoNot@Index#1{\egroup% we close the group,
  \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.
  \dont@index#1.}
```

\dont@index
\gmu@tempa

```
\long\def\dont@index#1{%
  \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.
  \if\gmu@tempa.% a dot finishes expelling
  \else
    \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
    \else% what is else shall off the Index be expelled.
```

[9] This idea comes from Marcin Woliński.

```
5184        {\escapechar\m@ne
5185          \xdef\gmu@tempa{\string#1}}%
5186        \@xa\let%
5187        \csname␣gmd/iexcl/\gmu@tempa\endcsname=\gmd@iedir% In the default
                case explained e.g. by the macro's name, the last macro's meaning is
                such that the test in line 4301 will turn false and the subject cs shall not
                be indexed. We \let not \def to spare TEX's memory.
5192        \afterfifi\dont@index
5193      \fi
5194    \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 LATEX source but normal packages.

```
5203 \DoNotIndex\{ \DoNotIndex\}% the index entries of these two css would be re-
          jected by MakeIndex anyway.
```

```
5206 \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

```
5210   \gdef\DefaultIndexExclusions{%
5211     \DoNotIndex{\@ \@@par \@beginparpenalty \@empty}%
5212     \DoNotIndex{\@flushglue \@gobble \@input}%
5213     \DoNotIndex{\@makefnmark \@makeother \@maketitle}%
5214     \DoNotIndex{\@namedef \@ne \@spaces \@tempa}%
5215     \DoNotIndex{\@tempb \@tempswafalse \@tempswatrue}%
5216     \DoNotIndex{\@thanks \@thefnmark \@topnum}%
5217     \DoNotIndex{\@@ \@elt \@forloop \@fortmp \@gtempa
           \@totalleftmargin}%
5218     \DoNotIndex{\" \/ \@ifundefined \@nil \@verbatim \@vobeyspaces}%
5219     \DoNotIndex{\| \~ \  \active \advance \aftergroup \begingroup
           \bgroup}%
5220     \DoNotIndex{\mathcal \csname \def \documentstyle \dospecials
           \edef}%
5221     \DoNotIndex{\egroup}%
5222     \DoNotIndex{\else \endcsname \endgroup \endinput \endtrivlist}%
5223     \DoNotIndex{\expandafter \fi \fnsymbol \futurelet \gdef \global}%
5224     \DoNotIndex{\hbox \hss \if \if@inlabel \if@tempswa
           \if@twocolumn}%
5225     \DoNotIndex{\ifcase}%
5226     \DoNotIndex{\ifcat \iffalse \ifx \ignorespaces \index \input
           \item}%
5227     \DoNotIndex{\jobname \kern \leavevmode \leftskip \let \llap
           \lower}%
5228     \DoNotIndex{\m@ne \next \newpage \nobreak \noexpand
           \nonfrenchspacing}%
5229     \DoNotIndex{\obeylines \or \protect \raggedleft \rightskip \rm
           \sc}%
5230     \DoNotIndex{\setbox \setcounter \small \space \string \strut}%
5231     \DoNotIndex{\strutbox}%
5232     \DoNotIndex{\thefootnote \thispagestyle \topmargin \trivlist
           \tt}%
5233     \DoNotIndex{\twocolumn \typeout \vss \vtop \xdef \z@}%
5234     \DoNotIndex{\, \@bsphack \@esphack \@noligs \@vobeyspaces
           \@xverbatim}%
```

```
5235 \DoNotIndex{\` \catcode \end \escapechar \frenchspacing \glossary}%
5236 \DoNotIndex{\hangindent \hfil \hfill \hskip \hspace \ht \it \langle}%
5237 \DoNotIndex{\leaders \long \makelabel \marginpar \markboth \mathcode}%
5238 \DoNotIndex{\mathsurround \mbox}% % \newcount \newdimen \newskip
5239 \DoNotIndex{\nopagebreak}%
5240 \DoNotIndex{\parfillskip \parindent \parskip \penalty \raise \rangle}%
5241 \DoNotIndex{\section \setlength \TeX \topsep \underline \unskip}%
5242 \DoNotIndex{\vskip \vspace \widetilde \\ \% \@date \@defpar}%
5243 \DoNotIndex{\[ \]}% see line 5203.
5244 \DoNotIndex{\count@ \ifnum \loop \today \uppercase \uccode}%
5245 \DoNotIndex{\baselineskip \begin \tw@}%
5246 \DoNotIndex{\a \b \c \d \e \f \g \h \i \j \k \l \m \n \o \p \q}%
5247 \DoNotIndex{\r \s \t \u \v \w \x \y \z \A \B \C \D \E \F \G \H}%
5248 \DoNotIndex{\I \J \K \L \M \N \O \P \Q \R \S \T \U \V \W \X \Y \Z}%
5249 \DoNotIndex{\1 \2 \3 \4 \5 \6 \7 \8 \9 \0}%
5250 \DoNotIndex{\! \# \$ \& \' \( \) \. \: \; \< \= \> \? \_}% \+ seems to be so rarely used that it may be advisable to index it.
5252 \DoNotIndex{\discretionary \immediate \makeatletter \makeatother}%
5253 \DoNotIndex{\meaning \newenvironment \par \relax \renewenvironment}%
5254 \DoNotIndex{\repeat \scriptsize \selectfont \the \undefined}%
5255 \DoNotIndex{\arabic \do \makeindex \null \number \show \write \@ehc}%
5256 \DoNotIndex{\@author \@ehc \@ifstar \@sanitize \@title}%
5257 \DoNotIndex{\if@minipage \if@restonecol \ifeof \ifmmode}%
5258 \DoNotIndex{\lccode % %\newtoks
5259   \onecolumn \openin \p@ \SelfDocumenting}%
5260 \DoNotIndex{\settowidth \@resetonecoltrue \@resetonecolfalse \bf}%
5261 \DoNotIndex{\clearpage \closein \lowercase \@inlabelfalse}%
5262 \DoNotIndex{\selectfont \mathcode \newmathalphabet \rmdefault}%
5263 \DoNotIndex{\bfdefault}%
```

From the above list I removed some `\new...` declarations because I think it may be useful to see gathered the special `\new...`s of each kind. For the same reason I would not recommend excluding from the index such declarations as `\AtBeginDocument`, `\AtEndDocument`, `\AtEndOfPackage`, `\DeclareOption`, `\DeclareRobustCommand` etc. But the common definitions, such as `\new/providecommand` and `\(e/g/x)defs`, as the most common, in my opinion excluded should be.

And some my exclusions:

```
5276 \DoNotIndex{\@@input \@auxout \@currentlabel \@dblarg}%
5277 \DoNotIndex{\@ifdefinable \@ifnextchar \@ifpackageloaded}%
5278 \DoNotIndex{\@indexfile \@let@token \@sptoken \^}% the latter comes from css like \^^M, see sec. 668.
5280 \DoNotIndex{\addto@hook \addvspace}%
5281 \DoNotIndex{\CurrentOption}%
5282 \DoNotIndex{\emph \empty \firstofone}%
5283 \DoNotIndex{\font \fontdimen \hangindent \hangafter}%
```

```
  \DoNotIndex{\hyperpage \hyperlink \hypertarget}%
  \DoNotIndex{\ifdim \ifhmode \iftrue \ifvmode \medskipamount}%
  \DoNotIndex{\message}%
  \DoNotIndex{\NeedsTeXFormat \newcommand \newif}%
  \DoNotIndex{\newlabel}%
  \DoNotIndex{\of}%
  \DoNotIndex{\phantom \ProcessOptions \protected@edef}%
  \DoNotIndex{\protected@xdef \protected@write}%
  \DoNotIndex{\ProvidesPackage \providecommand}%
  \DoNotIndex{\raggedright}%
  \DoNotIndex{\raisebox \refstepcounter \ref \rlap}%
  \DoNotIndex{\reserved@a \reserved@b \reserved@c \reserved@d}%
  \DoNotIndex{\stepcounter \subsection \textit \textsf \thepage \tiny}%
  \DoNotIndex{\copyright \footnote \label \LaTeX}%
  \DoNotIndex{\@eha \@endparenv \if@endpe \@endpefalse \@endpetrue}%
  \DoNotIndex{\@evenfoot \@oddfoot \@firstoftwo \@secondoftwo}%
  \DoNotIndex{\@for \@gobbletwo \@idxitem \@ifclassloaded}%
  \DoNotIndex{\@ignorefalse \@ignoretrue \if@ignore}%
  \DoNotIndex{\@input@ \@input}%
  \DoNotIndex{\@latex@error \@mainaux \@nameuse}%
  \DoNotIndex{\@nomath \@oddfoot}% %\@onlypreamble should be indexed IMO.
  \DoNotIndex{\@outerparskip \@partaux \@partlist \@plus}%
  \DoNotIndex{\@sverb \@sxverbatim}%
  \DoNotIndex{\@tempcnta \@tempcntb \@tempskipa \@tempskipb}%
```

I think the layout parameters even the kernel, should not be excluded: % \@topsep \@topsepadd \abovedisplayskip \clubpenalty etc.

```
  \DoNotIndex{\@writeckpt}%
  \DoNotIndex{\bfseries \chapter \part \section \subsection}%
  \DoNotIndex{\subsubsection}%
  \DoNotIndex{\char \check@mathfonts \closeout}%
  \DoNotIndex{\fontsize \footnotemark \footnotetext \footnotesize}%
  \DoNotIndex{\g@addto@macro \hfilneg \Huge \huge}%
  \DoNotIndex{\hyphenchar \if@partsw \IfFileExists }%
  \DoNotIndex{\include \includeonly \indexspace}%
  \DoNotIndex{\itshape \language \LARGE \Large \large}%
  \DoNotIndex{\lastbox \lastskip \m@th \makeglossary}%
  \DoNotIndex{\maketitle \math@fontsfalse \math@fontstrue \mathsf}%
  \DoNotIndex{\MessageBreak \noindent \normalfont \normalsize}%
  \DoNotIndex{\on@line \openout \outer}%
  \DoNotIndex{\parbox \part \rmfamily \rule \sbox}%
  \DoNotIndex{\sf@size \sffamily \skip}%
  \DoNotIndex{\textsc \textup \toks@ \ttfamily \vbox}%
%%\DoNotIndex{\begin*} maybe in the future, if the idea gets popular…
  \DoNotIndex{\hspace* \newcommand* \newenvironment* \providecommand*}%
  \DoNotIndex{\renewenvironment* \section* \chapter*}%
}% of \DefaultIndexExclusions.
```

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

```
\end{MakePrivateLetters}
```

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

```
\if@indexallmacros\else
  \DefaultIndexExclusions
\fi
```

If we expelled so many css, someone may like it in general but he/she may need one or two expelled to be indexed back. So

\DoIndex

```
\def\DoIndex{\bgroup\MakePrivateLetters\Do@Index}
```

\Do@Index

```
\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

```
\def\UndoDefaultIndexExclusions{%
  \StoreMacro\DoNotIndex
  \let\DoNotIndex\DoIndex
  \DefaultIndexExclusions
  \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

```
\long\def\IndexPrologue#1{\@bsphack\def\index@prologue{#1}%
     \@esphack}
```

\indexdiv

```
\def\indexdiv{\@ifundefined{chapter}{\section*}{\chapter*}}
```

\index@prologue

```
\@ifundefined{index@prologue}␣{\def\index@prologue{\indexdiv{%
     Index}%
    \markboth{Index}{Index}%
    Numbers␣written␣in␣italic␣refer␣to␣the␣\if@pageindex␣pages␣%
         \else
    code␣lines␣\fi␣where␣the
    corresponding␣entry␣is␣described;␣numbers␣underlined␣refer␣
         to␣the
    \if@pageindex\else␣code␣line␣of␣the␣\fi␣definition;␣numbers␣
         in
    roman␣refer␣to␣the␣\if@pageindex␣pages\else␣code␣lines␣\fi␣
         where
    the␣entry␣is␣used.
    \if@pageindex\else
      \ifx\HLPrefix\@empty
        The␣numbers␣preceded␣with␣`p.'␣are␣page␣numbers.
      \else␣The␣numbers␣with␣no␣prefix␣are␣page␣numbers.
    \fi\fi
    \ifx\IndexLinksBlack\relax\else
      All␣the␣numbers␣are␣hyperlinks.
    \fi
    \gmd@dip@hook% this hook is intended to let a user add something without redefining the entire prologue, see below.
  }}{}
```

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

```
               \@emptify\gmd@dip@hook
\AtDIPrologue  \long\def\AtDIPrologue#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

```
\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         \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   \newcount\c@IndexColumns␣\c@IndexColumns␣=␣3
theindex          \renewenvironment{theindex}
                    {\begin{multicols}\c@IndexColumns[\index@prologue][\IndexMin]%
                      \IndexLinksBlack
                      \IndexParms␣\let\item\@idxitem␣\ignorespaces}%
                    {\end{multicols}}
```

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

```
                  \@ifundefined{IndexParms}
\IndexParms         {\def\IndexParms{%
                      \parindent␣\z@
                      \columnsep␣15pt
                      \parskip␣0pt␣plus␣1pt
                      \rightskip␣15pt
                      \mathsurround␣\z@
                      \parfillskip=-15pt␣plus␣1␣fil␣% doc defines this parameter rigid but that's because of the stretchable space (more precisely, a \dotfill) between the item and the entries. But in gmdoc we define no such special delimiters, so we add an ifinite stretch.
                      \small
                      \def\@idxitem{\par\hangindent␣30pt}%
\subitem              \def\subitem{\@idxitem\hspace*{15pt}}%
\subsubitem           \def\subsubitem{\@idxitem\hspace*{25pt}}%
                      \def\indexspace{\par\vspace{10pt␣plus␣2pt␣minus␣3pt}}%
                      \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.
                      \ifnum\c@IndexColumns>\tw@\raggedright\fi% the numbers in narrow columns look better when they are \raggedright in my opinion.
                    }}{}
```

```
\PrintIndex       \def\PrintIndex{% we ensure the standard meaning of the line end character not
```

to cause a disaster.

```
\@ifQueerEOL{\StraightEOL\printindex\QueerEOL}%
{\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 LaTeX command \g@addto@macro (it has \global effect, also with an apeless name (\gaddtomacro) provided by gmutils. (It adds its second argument at the end of definition of its first argument provided the first argument is a no-argument macro.) Moreover, gmutils provides also \addtomacro that has the same effect except it's not \global.

The DocStrip Directives

```
\foone{\@makeother\<\@makeother\>
  \glet\sgtleftxii=<}
{
```

\gmd@docstripdirective

```
  \def\gmd@docstripdirective{%
    \begingroup\let\do=\@makeother
    \do\*\do\/\do\+\do\-\do\,\do\&\do\|\do\!\do\(\do\)\do\>\do\<%
    \@ifnextchar{<}{%
      \let\do=\@makeother␣\dospecials
      \gmd@docstripverb}
    {\gmd@docstripinner}}%
```

\gmd@docstripinner

```
  \def\gmd@docstripinner#1>{%
    \endgroup
```

\gmd@modulehashone

```
    \def\gmd@modulehashone{%
      \Module{#1}\space
      \@afternarrgfalse\@aftercodegtrue\@codeskipputgfalse}%
    \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.

```
}
```

```
\foone{\@makeother\<\@makeother\>
  \glet\sgtleftxii=<
  \catcode`\^^M=\active}%
{
```

\gmd@docstripverb

```
  \def\gmd@docstripverb<#1^^M{%
    \endgroup%
```

\gmd@modulehashone

```
    \def\gmd@modulehashone{%
      \ModuleVerb{#1}\@afternarrgfalse\@aftercodegtrue%
      \@codeskipputgfalse}%
    \gmd@docstripshook%
    \gmd@textEOL\gmd@modulehashone^^M}%
}
```

(~Verbatim ;-) from doc:)

```
\Module                \providecommand*\Module[1]{{\mod@math@codes$\langle\mathsf{#1}%
                            \rangle$}}
\ModuleVerb            \providecommand*\ModuleVerb[1]{{\mod@math@codes$\langle\langle%
                            \mathsf{#1}$}}
\mod@math@codes        \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     \def\gmd@DefineChanges{%
\changes                 \outer\long\def\changes{\@bsphack\begingroup\@sanitize
                           \catcode`\\\z@␣\catcode`\ 10␣\MakePercentIgnore
                           \MakePrivateLetters␣\StraightEOL
                           \MakeGlossaryControls
                           \changes@}}
\changes               \newcommand\changes[4][]{\PackageWarningNoLine{gmdoc}{%
                           ^^JThe␣\bslash␣changes␣command␣used␣\on@line
                           ^^Jwith␣no␣\string\RecordChanges\space␣declared.
                           ^^JI␣shall␣not␣warn␣you␣again␣about␣it}%
\changes                 \renewcommand\changes[4][]{%
                         }}
\MakeGlossaryControls  \def\MakeGlossaryControls{%
                         \edef\actualchar{\string=}\edef\quotechar{\string!}%
                         \edef\levelchar{\string>}\edef\encapchar{\xiiclub}}% 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@              \newcommand\changes@[4][\generalname]{%
```

```
5683  \if@RecentChange{#3}% if the date is later than the one stored in \c@ChangesStartDate,
5685    \@tempswafalse
5686    \ifx\generalname#1% then we check whether a cs-entry is given in the optional first argument or is it unchanged.
5688      \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.
5691        \@tempswatrue
5692        \edef\gmu@tempb{% it's a bug fix: while typesetting traditional .dtxes, \last@defmark came out with \ at the beginning (which resulted 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).
5700          \if\bslash\last@defmark\else\last@defmark\fi}%
5701        \ifx\last@defmark\bslash\let\gmu@tempb\last@defmark\fi%
5702        \n@melet{gmd@glossCStest}{gmd/isaCS/\last@defmark}%
5703      \fi
5704    \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]...)
5708      \@tempswatrue
5709      {\escapechar\m@ne
5710        \xdef\gmu@tempb{\string#1}}%
5711      \if\bslash\@xa\@firstofmany\string#1\relax\@@nil% we check whether #1 is a cs...
\gmd@glossCStest  5713        \def\gmd@glossCStest{1}% ... and tell the glossary if so.
5714      \fi
5716    \fi
\gmd@glossCStest  5717    \@ifundefined{gmd@glossCStest}{\def\gmd@glossCStest{0}}{}%
5718    \protected@edef\gmu@tempa{\@nx\gmd@glossary{%
5719        \if\relax\GeneralName\relax\else
5720          \GeneralName% it's for the \DocInclude case to precede every \changes of the same file with the file name, cf. line 6163.
5723        \fi
5724        #2\levelchar%
5725        \if@tempswa% If the macro \last@defmark doesn't contain any cs name (i.e., is empty) nor #1 specifies a cs, the current changes entry was done at top-level. In this case we precede it by \generalname.
5730          \gmu@tempb
5731          \actualchar\bslash␣verb*%
5732          \if\verbatimchar\gmu@tempb$\else\verbatimchar\fi
5733          \if1\gmd@glossCStest\quotechar\bslash\fi␣\gmu@tempb
5734          \if\verbatimchar\gmu@tempb$\else\verbatimchar\fi
5735        \else
5736          \space\actualchar\generalname
5737        \fi
5738        :\levelchar%
5739        #4%
5740      }}%
5741    \gmu@tempa
```

```
    \grelaxen\gmd@glossCStest
  \fi% of \if@recentchange
  \endgroup\@esphack}
```

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

```
\@relaxen\last@defmark
\@emptify\GeneralName
```

\ChangesGeneral

```
\def\ChangesGeneral{\grelaxen\last@defmark}% If automatic detection of definitions is on, the default entry of \changes is the meaning of \last@defmark, the last detected definiendum that is. The declaration defined here serves to start a scope of 'general' \changes' entries.
```

```
\AtBegInput{\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 0 so if no date is declared there'll be no problem. The date will have the ⟨*YYYYMMDD*⟩ shape both to be easily compared and readable.

\c@ChangesStartDate

```
\newcount\c@ChangesStartDate
```

\if@RecentChange

```
\def\if@RecentChange#1{%
  \gmd@setChDate#1\@@nil\@tempcnta
  \ifnum\@tempcnta>\c@ChangesStartDate}
```

\gmd@setChDate

```
\def\gmd@setChDate#1/#2/#3\@@nil#4{% the last parameter will be a \count register.
  #4=#1\relax
  \multiply#4␣by\@M
  \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 \count0 (\@tempcnta) and some higher (not a scratch) one.
  \multiply\count8␣by100␣%
  \advance#4␣by\count8␣\count8=\z@
  \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

```
\def\ChangesStart#1#2{%
  \gmd@setChDate#2\@@nil\c@ChangesStartDate
  \typeout{^^JPackage␣gmdoc␣info:␣^^JChanges'␣start␣date␣#1␣
      memorized
    as␣\string<\the\c@ChangesStartDate\string>␣\on@line.^^J}
  \advance\c@ChangesStartDate\m@ne% we shall show the changes at the specified day and later.
  \ifnum\c@ChangesStartDate>19820900␣%[10] see below.
    \edef\gmu@tempa{%
      \@nx\g@addto@macro\@nx\glossary@prologue{%
        The␣changes
        \if\relax\GeneralName\relax\else␣of␣\GeneralName\space\fi
        earlier␣than
        #1␣\if\relax#1\relax␣#2\else(#2)\fi\space␣are␣not␣
            shown.}}%
```

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

```
5806     \gmu@tempa
5807   \fi}
```

(Explanation to line 5796.) 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 5796 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   5845 \def\generalname{General}
```

"To cause the changes to be written (to a .glo) file, we define `\RecordChanges` to invoke LATEX's usual `\makeglossary` command."

I add to it also the `\writeing` definition of the `\changes` macro to ensure no changes are written out without `\RecordChanges`.

```
\RecordChanges   5857 \def\RecordChanges{\makeglossary\gmd@DefineChanges
                 5858   \@relaxen\RecordChanges}
```

"The remaining macros are all analogues of those used for the `theindex` environment. When the glossary is started we compute the space which remains at the bottom of the current page; if this is greater than `\GlossaryMin` then the first part of the glossary will be placed in the available space. The number of columns set [is] controlled by the counter `\c@GlossaryColumns` which can be changed with a `\setcounter` declaration."

```
\GlossaryMin         5870 \newdimen\GlossaryMin          \GlossaryMin        = 80pt
\c@GlossaryColumns   5872 \newcount\c@GlossaryColumns    \c@GlossaryColumns = 2
```

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

```
theglossary   5878 \newenvironment{theglossary}{%
              5880   \begin{multicols}\c@GlossaryColumns
              5881     [\glossary@prologue][\GlossaryMin]%
              5882     \GlossaryParms␣\IndexLinksBlack
              5883     \let\item\@idxitem␣\ignorespaces}%
              5884   {\end{multicols}}
```

Here is the MakeIndex style definition:

```
5889 ⟨/package⟩
5890 ⟨+gmglo⟩ preamble
5891 ⟨+gmglo⟩ "\n␣\\begin{theglossary}␣\n
5892 ⟨+gmglo⟩ \\makeatletter\n"
5893 ⟨+gmglo⟩ postamble
```

```
⟨+gmglo⟩ "\n\n␣\\end{theglossary}\n"
⟨+gmglo⟩ keyword␣"\\glossaryentry"
⟨+gmglo⟩ actual␣'='
⟨+gmglo⟩ quote␣'!'
⟨+gmglo⟩ level␣'>'
⟨*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
\glossary@prologue

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

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

\glossary@prologue

```
\@ifundefined{glossary@prologue}
  {\def\glossary@prologue{\indexdiv{{Change␣History}}%
      \markboth{{Change␣History}}{{Change␣History}}%
    }}{}
```

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

\GlossaryParms

```
\AtBeginDocument{%
  \@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

```
\def\PrintChanges{% to avoid a disaster among queer EOLs:
  \@ifQueerEOL
    {\StraightEOL\@input@{\jobname.gls}\QueerEOL}%
    {\@input@{\jobname.gls}}%
    \g@emptify\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      5986 \newcount\check@sum
\CheckSum       5988 \def\CheckSum#1{\@bsphack\global\check@sum#1\relax\@esphack}
CheckSum        5990 \newcounter{CheckSum}
\step@checksum  5993 \newcommand*\step@checksum{\stepcounter{CheckSum}}
```

And we'll use it in the line 3402 (\stepcounter is \global). See also the \chschange declaration, l. 6074.

However, the check sum mechanism in gmdoc behaves slightly different than in doc which is nicely visible while gmdocing doc: doc states its check sum to be 2171 and our count counts 2126. The mystery lies in the fact that doc's CheckSum mechanism counts the code's backslashes no matter what they mean and the gmdoc's the escape chars so, among others, \\ at the default settings increases doc's CheckSum by 2 while the gmdoc's by 1. (There are 38 occurrences of \\ in doc.dtx macrocodes, I counted myself.)[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.

```
6020 \AtEndInput{\check@checksum}
```

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

```
\check@checksum 6023 \def\check@checksum{\relax
                6024   \ifnum\check@sum=\z@
                6025     \edef\gmu@tempa{% why \edef—see line 6053
                6026       \@nx\typeout{**********************************^^J%
                6027         *␣The␣input␣file␣\gmd@inputname\space␣has␣no␣Checksum
                6028         stated.^^J%
                6029         *␣The␣current␣checksum␣is␣\the\c@CheckSum.^^J%
                6030         \gmd@chschangeline% a check sum changes history entry, see below.
                6031         *␣(package␣gmdoc␣info.)^^J%
                6032         **********************************^^J}}
                6033     \else
                6034     \ifnum\check@sum=\c@CheckSum
                6035       \edef\gmu@tempa{%
                6036         \@nx\typeout{*********************^^J%
                6037           *␣The␣input␣file␣\gmd@inputname:␣Checksum␣passed.^^J%
                6038           \gmd@chschangeline
                6039           *␣(package␣gmdoc␣info.)^^J%
                6040           *********************^^J}}
                6041     \else
                6042       \edef\gmu@tempa{%
                6043         \@nx\typeout{*******!*!*!*!*!*!*!*!*!*!*!*!^^J%
                6044           *!␣The␣input␣file␣\gmd@inputname:^^J%
                6045           *!␣The␣CheckSum␣stated:␣\the\check@sum\space<>␣my
                6046           count:␣\the\c@CheckSum.^^J%
                6047           \gmd@chschangeline
                6048           *!␣(package␣gmdoc␣info.)^^J%
                6049           *******!*!*!*!*!*!*!*!*!*!*!*!^^J}}%
                6050     \fi
                6051   \fi
```

[11] My opinion is that nowadays a check sum is not necessary for checking the completness of a file but I like it as a marker of file development and this more than that is its rôle in gmdoc.

```
  \gmu@tempa
  \@xa\AtEndDocument\@xa{\gmu@tempa}% we print the checksum notification
        on the terminal immediately and at end of TEXing not to have to scroll the
        output far nor search the log.
  \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

```
\def\gmd@chschangeline{%
  \xiipercent\space\string\chschange
  {\csname␣fileversion\endcsname}%
  {\the\year/\the\month/\the\day}%
  {\the\c@CheckSum}^^J%
  \xiipercent\space\string\chschange
  {\csname␣fileversion\endcsname}%
  {\@xa\@gobbletwo\the\year/\the\month/\the\day}%
  {% with two digit year in case you use \ChangesStart.
    \the\c@CheckSum}^^J}
```

And here the meaning of such a line is defined:

\chschange

```
\newcommand*\chschange[3]{%
  \csname␣changes\endcsname{#1}{#2}{CheckSum␣#3}% \csname... because
        % \changes is \outer.
  \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

```
\newcommand*\ltxPageLayout{%
```

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

```
  \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 `\addtolength`s with `\setlength`s.

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

                 \egroup% having the arguments read, we close the group opened by the previous
                           macro for _12.
\HLPrefix        \gdef\HLPrefix{\filesep}%
                 \gdef\EntryPrefix{\filesep}% we define two rather kernel parameters to ex-
                           pand to the file marker. The first will bring the information to one of the
                           default \IndexPrologue's \ifs. Therefore the definition is global. The lat-
                           ter is such for symmetry.
\GeneralName     \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.
                   \edef\gmd@jobname{\jobname}%
                   \edef\gmd@difilename{% we want the filename all 'other', just as in \jobname.
                     \@xa\@xa\@xa\@gobble\@xa\string\csname#2\endcsname}%
                   \ifx\gmd@jobname\gmd@difilename
\gmd@auxext          \def\gmd@auxext{auxx}%
                   \else
\gmd@auxext          \def\gmd@auxext{aux}%
                   \fi
                 \relax
                 \clearpage
                 \gmd@docincludeaux
\currentfile     \def\currentfile{gmdoc-IncludeFileNotFound.000}%
                 \let\fullcurrentfile\currentfile
                 \IfFileExists{#1#2.fdd}{\edef\currentfile{#2.fdd}}{% it's not .fdd,
                   \IfFileExists{#1#2.dtx}{\edef\currentfile{#2.dtx}}{% it's not .dtx
                           either,
                     \IfFileExists{#1#2.sty}{\edef\currentfile{#2.sty}}{% it's not .sty,
                       \IfFileExists{#1#2.cls}{\edef\currentfile{#2.cls}}{% it's not
                           .cls,
                         \IfFileExists{#1#2.tex}{\edef\currentfile{#2.tex}}{% it's not
                           .tex,
                           \IfFileExists{#1#2.fd}{\edef\currentfile{#2.fd}}{% so it
                           must be .fd or error.
                             \PackageError{gmdoc}{\string\DocInclude\space␣file
                               #1#2.fdd/dtx/sty/cls/tex/fd␣not␣found.}}}}}}}%
                 \edef\fullcurrentfile{#1\currentfile}%
6202  \ifnum\@auxout=\@partaux
6203    \@latexerr{\string\DocInclude\space␣cannot␣be␣nested}\@eha
6204  \else␣\@docinclude{#1}#2␣\fi}% Why is #2 delimited with  not braced as
          we are used to, one may ask.
```

\@docinclude

```
6210 \def\@docinclude#1#2␣{% To match the macro's parameter string, is an answer.
        But why is \@docinclude defined so? Originally, in ltxdoc it takes one argument
        and it's delimited with a space probably in resemblance to the true
        \input (\@@input in LaTeX).
6215  \clearpage
6217  \if@filesw␣\gmd@writemauxinpaux{#2.\gmd@auxext}\fi% this strange macro
          with a long name is another thing to allow _ in the filenames (see line 6278).
6220  \@tempswatrue
6221  \if@partsw␣\@tempswafalse\edef\gmu@tempb{#2}%
6222    \@for␣\gmu@tempa:=\@partlist\do{\ifx\gmu@tempa\gmu@tempb%
            \@tempswatrue\fi}%
6223  \fi
6224  \if@tempswa␣\let\@auxout\@partaux
6225    \if@filesw
6226      \immediate\openout\@partaux␣#2.\gmd@auxext\relax% Yes, only #2.
              It's to create and process the partial .aux(x) files always in the main
              document's (driver's) directory.
6231      \immediate\write\@partaux{\relax}%
6232    \fi
```

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

```
6239    \StoringAndRelaxingDo\gmd@doIndexRelated
6240    \if@ltxDocInclude\part{\currentfile}% In the ltxdoc-like setup we make
            a part title page with only the filename and the file's \maketitle will
            typeset an article-like title.
6243    \else\let\maketitle=\InclMaketitle
6244    \fi% In the default setup we redefine \maketitle to typeset a common chapter
            or part heading.
6246    \if@ltxDocInclude\xdef@filekey\fi
6247    \GetFileInfo{\currentfile}% it's my (GM) addition with the account of
            using file info in the included files' title/heading etc.
6249    \incl@DocInput{\fullcurrentfile}% originally just \currentfile.
6250    \if@ltxDocInclude\else\xdef@filekey\fi% in the default case we add
            new file to the file key *after* the input because in this case it's the files
            own \maketitle what launches the sectioning command that increases
            the counter.
```

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

```
6256    \RestoringDo\gmd@doIndexRelated
6258    \clearpage
6260    \gmd@writeckpt{#1#2}%
6261    \if@filesw␣\immediate\closeout\@partaux␣\fi
6262  \else\@nameuse{cp@#1#2}%
6263  \fi
6264  \let\@auxout\@mainaux}% end of \@docinclude.
```

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

\xdef@filekey

```
6268 \def\xdef@filekey{{\@relaxen\ttfamily% This assignment is very trickly crafted:
```

it makes *all* \ttfamilys present in the \filekey's expansion unexpandable not only the one added in this step.

```
    \xdef\filekey{\filekey,␣\thefilediv={\ttfamily%
        \currentfile}}}}
```

To allow _ in the filenames we must assure _ will be $_{12}$ while reading the filename. Therefore define

\gmd@writemauxinpaux

```
\def\gmd@writemauxinpaux#1{% this name comes from 'write outto main .aux to
        input partial .aux'.
```

We wrap \@input{⟨*partial* .aux⟩} in a $\_{12}$ hacked scope. This hack is especially recommended here since the .aux file may contain a non-\global stuff that should not be 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.)

```
  \immediate\write\@mainaux{%
    \bgroup\string\@makeother\string\_%
    \string\firstofone{\egroup
    \string\@input{#1}}}}
```

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

\gmd@writeckpt

```
\def\gmd@writeckpt#1{%
  \immediate\write\@partaux{%
    \string\bgroup\string\@makeother\string\_%
    \string\firstofone\@charlb\string\egroup}
  \@writeckpt{#1}%
  \immediate\write\@partaux{\@charrb}}
```

\gmd@doIndexRelated

```
\def\gmd@doIndexRelated{%
  \do\tableofcontents␣\do\makeindex␣\do\EnableCrossrefs
  \do\PrintIndex␣\do\printindex␣\do\RecordChanges␣\do%
       \PrintChanges
  \do\theglossary␣\do\endtheglossary}
```

```
\@emptify\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

\aalph
\@aalph

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

A macro that initialises things for \DocInclude.

\gmd@docincludeaux

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

We set the things for including the files only once.

```
  \global\@relaxen\gmd@docincludeaux
```

By default, we will include multiple files into one document as chapters in the classes that provide \chapter and as parts elsewhere.

```
6333  \ifx\filediv\relax
6334    \ifx\filedivname\relax% (nor \filediv neither \filedivname is defined
            by the user)
6338      \@ifundefined{chapter}{%
6339        \SetFileDiv{part}}%
6342      {\SetFileDiv{chapter}}%
6343    \else% (\filedivname is defined by the user, \filediv is not)
6344      \SetFileDiv{\filedivname}% why not? Inside is \edef so it'll work.
6345    \fi
6346  \else% (\filediv is defined by the user
6347    \ifx\filedivname\relax% and \filedivname is not)
6350      \PackageError{gmdoc}{You've␣redefined␣\string\filediv\space
6351        without␣redefining␣\string\filedivname.}{Please␣redefine␣
              the
6352        two␣macros␣accordingly.␣You␣may␣use␣\string\SetFileDiv{%
              name
6353          without␣bslash}.}%
6354    \fi
6355  \fi
\thefilediv  6364  \def\thefilediv{\aalph{\filedivname}}% The files will be numbered with
          letters, lowercase first.
6366  \@xa\let\csname␣the\filedivname\endcsname=\thefilediv% This line lets
          \the⟨chapter⟩ etc. equal \thefilediv.
\filesep  6368  \def\filesep{\thefilediv-}% File separator (identifier) for the index.
6369  \let\filekey=\@gobble
6370  \g@addto@macro\index@prologue{%
6371    \gdef\@oddfoot{\parbox{\textwidth}{\strut\footnotesize
6372        \raggedright{\bfseries␣File␣Key:}␣\filekey}}% The footer for the
              pages of index.
6374    \glet\@evenfoot\@oddfoot}% anyway, it's intended to be oneside.
6376  \g@addto@macro\glossary@prologue{%
6377    \gdef\@oddfoot{\strut␣Change␣History\hfill\thepage}% The footer for
            the changes history.
6379    \glet\@evenfoot\@oddfoot}%
6382  \gdef\@oddfoot{% The footer of the file pages will be its name and, if there is
          a file info, also the date and version.
6384    \@xa\ifx\csname␣ver@\currentfile\endcsname\relax
6385      File␣\thefilediv:␣{\ttfamily\currentfile}␣%
6386    \else
6387      \GetFileInfo{\currentfile}%
6388      File␣\thefilediv:␣{\ttfamily\filename}␣%
6389      Date:␣\filedate\ %
6390      Version␣\fileversion
6391    \fi
6392    \hfill\thepage}%
6393  \glet\@evenfoot\@oddfoot% see line 6374.
6395  \@xa\def\csname\filedivname␣name\endcsname{File}% we redefine the name
          of the proper division to 'File'.
6397  \ifx\filediv\section
6398    \let\division=\subsection
6399    \let\subdivision=\subsubsection
6400    \let\subsubdivision=\paragraph
```

If \filediv is higher than \section we don't change the three divisions (they are \section, \subsection and \subsubsection by default). \section seems to me the lowest reasonable sectioning command for the file. If \filediv is lower you should rather rethink the level of a file in your documentation not redefine the two divisions.

```
  \fi}% end of \gmd@docincludeaux.
```

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

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

\SelfInclude

```
\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

```
\newif\if@ltxDocInclude
```

\ltxLookSetup

```
\newcommand*\ltxLookSetup{%
  \SetFileDiv{part}%
  \ltxPageLayout
  \@ltxDocIncludetrue
}
```

```
\@onlypreamble\ltxLookSetup
```

The default is that we \DocInclude the files due to the original gmdoc input settings.

```
\let\incl@DocInput=\DocInput
```

```
\@emptify\currentfile% for the pages outside the \DocInclude's scope. In force
        for all includes.
```

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

\olddocIncludes

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

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

\gmdocIncludes

```
\newcommand*\gmdocIncludes{%
  \let\incl@DocInput=\DocInput
  \AtBegInput{\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.

\maketitle

```
\AtBegInput{%
  \providecommand*\maketitle{\par
```

```
              \begingroup␣\def␣\thefootnote␣{\fnsymbol␣{footnote}}%
              \setcounter␣{footnote}\z@
              \def\@makefnmark{\hbox␣to\z@{$\m@th^{\@thefnmark}$\hss}}%
\@makefntext  \long\def\@makefntext##1{\parindent␣1em\noindent
                \hbox␣to1.8em{\hss$\m@th^{\@thefnmark}$}##1}%
              \if@twocolumn␣\twocolumn␣[\@maketitle␣]%
              \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."

```
              \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:"

```
              \setcounter␣{footnote}\z@
              \gdef\@date{\today}\g@emptify\@thanks%
              \g@emptify\@author\g@emptify\@title%
            }%
```

"When a number of articles are concatenated into a journal, for example, it is not usual for the title pages of such documents to be formatted differently. Therefore, a class such as ltugboat can define this macro in advance. However, if no such definition exists, we use pagestyle plain for title pages."

```
            \@ifundefined{ps@titlepage}{\let\ps@titlepage=\ps@plain}{}%
```

And let's provide `\@maketitle` just in case: an error occurred without it at TEXing 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.

```
            \providecommand*\@maketitle{%
              \newpage\null␣\vskip␣2em\relax%
              \begin{center}%
                \titlesetup
                \let␣\footnote␣\thanks
                {\LARGE␣\@title␣\par}%
                \vskip␣1.5em%
                {\large␣\lineskip␣.5em%
                  \begin{tabular}[t]{c}%
                    \strut␣\@author
                  \end{tabular}\par}%
                \vskip␣1em%
                {\large␣\@date}%
              \end{center}%
              \par␣\vskip␣1.5em\relax}%
```

We'd better restore the primary meanings of the macros making a title. (LATEX 2ε source, File F: ltsect.dtx Date: 1996/12/20 Version v1.0z, lines 3.5.7.9–12.14–17.)

```
\title      \providecommand*\title[1]{\gdef\@title{#1}}
\author     \providecommand*\author[1]{\gdef\@author{#1}}
\date       \providecommand*\date[1]{\gdef\@date{#1}}
\thanks     \providecommand*\thanks[1]{\footnotemark
              \protected@xdef\@thanks{\@thanks
                \protect\footnotetext[\the\c@footnote]{#1}}%
            }%
```

```
\and         6547   \providecommand*\and{%  % \begin{tabular}
             6548     \end{tabular}%
             6549     \hskip␣1em␣\@plus.17fil%
             6550     \begin{tabular}[t]{c}}%  %  \end{tabular} And finally, let's initialize
                           \titlesetup if it is not yet.
\titlesetup  6552     \providecommand*\titlesetup{}%
             6553 }% end of \AtBegInput.
```

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 hte `\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.

```
6567 \@relaxen\filediv
6568 \@relaxen\filedivname
6569 \@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  6579 \def\InclMaketitle{%
\and            6582   {\def\and{,␣}% we make \and just a comma.
                6583     {\let\thanks=\@gobble% for the toc version of the heading we discard \thanks.
                6585       \protected@xdef\incl@titletotoc{\@title\if@fshda\protect%
                               \space
                6586         (\@author)\fi}% we add the author iff the 'files have different authors'
                                 % (@fshda)
                6588     }%
\thanks         6589     \def\thanks##1{\footnotemark
                6590       \protected@xdef\@thanks{\@thanks% to keep the previous \thanks if
                               there were any.
                6592         \protect\footnotetext[\the\c@footnote]{##1}}}% for some mys-
                                 terious reasons so defined \thanks do typeset the footnote mark
                                 and text but they don't hyperlink it properly. A hyperref bug?
                6596     \@emptify\@thanks
                6597     \protected@xdef\incl@filedivtitle{%
                6598       [{\incl@titletotoc}]% braces to allow [ and ] in the title to toc.
                6600       {\protect\@title
                6601         {\smallerr% this macro is provided by the gmutils package after the rel-
                                 size package.
                6603           \if@fshda\\[0.15em]\protect\@author
                6604             \if\relax\@date\relax\else,␣\fi
                6605           \else
                6606             \if\relax\@date\relax\else\\[0.15em]\fi
                6607           \fi
```

The default is that all the included files have the same author(s). In this case we won't print the author(s) in the headings. Otherwise we wish to print them. The information which case are we in is brought by the `\if@fshda` switch defined in line 6638.

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.

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

```
  }% end of the comma-\and's group.
  \@xa\filediv\incl@filedivtitle
  \@thanks
  \g@relaxen\@author␣\g@relaxen\@title␣\g@relaxen\@date
  \g@emptify\@thanks
}% 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             \newif\if@fshda
```

(its name comes from *f*ile*s* *h*ave *d*ifferent *a*uthors).

```
\PrintFilesAuthors    \newcommand*\PrintFilesAuthors{\@fshdatrue}
```

And the counterpart, if you change your mind:

```
\SkipFilesAuthors     \newcommand*\SkipFilesAuthors{\@fshdafalse}
```

The File's Date and Version Information

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

```
\GetFileInfo          \def\GetFileInfo#1{%
\filename               \def\filename{#1}%
\gmu@tempb              \def\gmu@tempb##1␣##2␣##3\relax##4\relax{%
\filedate                 \def\filedate{##1}%
\fileversion              \def\fileversion{##2}%
\fileinfo                 \def\fileinfo{##3}}%
                        \edef\gmu@tempa{\csname␣ver@#1\endcsname}%
                        \@xa\gmu@tempb\gmu@tempa\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 TEXnical and do it with writing to a file and inputting that file.

```
\FileInfo             \newcommand*\FileInfo{%
                        \bgroup
                        \gmd@ctallsetup
                        \bgroup% yes, we open two groups because we want to rescan tokens in 'usual'
                              catcodes. We cannot put \gmd@ctallsetup into the inner macro because
                              when that will be executed, the \inputlineno will be too large (the last not
                              the first line).
                        \let\do\@makeother
                        \do\ \do\{\do\}\do\^^M\do\\%
                        \gmd@fileinfo}

                      \foone{%
```

```
  \catcode`!\z@
  \catcode`(\@ne
  \catcode`)\tw@
  \let\do\@makeother
  \do\  % we make space ‘other’ to keep it for scanning the code where it may be
        leading.
  \do\{\do\}\do\^^M\do\\}%
(%
```

\gmd@fileinfo

```
!def!gmd@fileinfo#1Provides#2{#3}#4[#5]#6^^M%
(!egroup% we close the group of changed catcodes, the catcodes of the arguments
        are set. And we are still in the group for \gmd@ctallsetup.
!gmd@writeFI(#2)(#3)(#5)%
!gmd@FIrescan(#1Provides#2{#3}#4[#5]#6)% this macro will close the group.
)%
)
```

\gmd@writeFI

```
\def\gmd@writeFI#1#2#3{%
  \immediate\write\@auxout{%
    \global\@nx\@namedef{%
      ver@#2.\if␣P\@firstofmany#1\@@nil␣sty\else␣cls\fi}{#3}}}
```

```
\foone\obeylines{%
```

\gmd@FIrescan

```
  \def\gmd@FIrescan#1{%
 {\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_ϵ, file L v1.1g, l. 102.

\ProvideFileInfo

```
\def\ProvideFileInfo#1{%
  \begingroup
    \catcode`\ 10␣\catcode\endlinechar␣10␣%
    \@makeother\/\@makeother\&%
    \kernel@ifnextchar[{\gmd@providefii{#1}}{\gmd@providefii{#1}[]}%
  }
```

\gmd@providefii

```
\def\gmd@providefii#1[#2]{%
```

(we *don't* write the file info to .log)

```
  \@xa\xdef\csname␣ver@#1\endcsname{#2}%
  \endgroup}
```

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

\ProvideSelfInfo

```
\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

```
\newcommand*\filenote{This␣file␣has␣version␣number␣\fileversion{%
      }␣dated␣\filedate{}.}
```

And exactly as \thanks:

\thfileinfo

```
\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:

```
\foone{\obeylines}%
```

```
{%
  \def\IndexInput#1{%
    \StoreMacro\code@delim%
    \CodeDelim\^^Z%
    \def\gmd@iihook{% this hook is \edefed!
      \@nx^^M%
      \code@delim\relax\@nx\let\@nx\EOFMark\relax}%
    \DocInput{#1}\RestoreMacro\code@delim}%
}
```

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 `\skipgmlonely` declaration's scope it gobbles its contents.

```
\newenvironment{gmlonely}{\StraightEOL}{}
```

```
\newcommand\skipgmlonely[1][]{%
  \def\gmu@tempa{%
    \def\gmd@skipgmltext{%
      \g@emptify\gmd@skipgmltext
      #1%
    }}% not to count the lines of the substituting text but only of the text omitted
  \gmu@tempa
  \@xa\AtBegInput\@xa{\gmu@tempa}%
  \renewenvironment{gmlonely}{%
    \StraightEOL
    \@fileswfalse% to forbid writing to .toc, .idx etc.
    \setbox0=\vbox\bgroup}{\egroup\gmd@skipgmltext}}
```

Sometimes in the commentary of this package, so maybe also others, I need to say some char is of category 12 ('other sign'). This I'll mark just as $_{12}$ got by `\catother`.

```
\foone{\catcode`\_=8␣}% we ensure the standard \catcode of _.
{
  \newcommand*\catother{${}_{12}$}%
```

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

```
  \newcommand*\catactive{${}_{13}$}%
```

and a letter, $_{11}$

```
  \newcommand*\catletter{${}_{11}$}%.
}
```

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

```
\newenvironment*{copyrnote}{%
  \StraightEOL\everypar{\hangindent3em\relax\hangafter1␣}%
  \par\addvspace\medskipamount\parindent\z@\obeylines}{%
  \@codeskipputgfalse\stanza}
```

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

```
\StoreEnvironment{quotation}
```

```
\gmd@quotationname  \def\gmd@quotationname{quotation}
         quotation  \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.

```
  \ifx\@currenvir\gmd@quotationname
  \afterfi{\par``\ignorespaces}%
  \else\afterfi{\storedcsname{quotation}}%
  \fi}
{\ifx\@currenvir\gmd@quotationname
  \afterfi{\ifhmode\unskip\fi''\par}%
  \else\afterfi{\storedcsname{endquotation}}%
  \fi}
```

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

```
\gmdnoindent  \newcommand*\gmdnoindent{\leavevmode\hskip-\parindent}
```

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  \newcommand*\inverb{%
                     \@ifstar{%
       \gmu@tempa      \def\gmu@tempa{{\tt\xiipercent}}%
                       \@emptify\gmu@tempb% here and in the paralell points of the other case and
                            % \nlpercent I considered an \ifhmode test but it's not possible to be
                            in vertical mode while in an inline comment. If there happens vertical
                            mode, the commentary begins to be 'outline' (main text).
                       \gmd@inverb}%
                     {\@emptify\gmu@tempa
       \gmu@tempb      \def\gmu@tempb{\gmboxedspace}%
                       \gmd@inverb}}

    \gmboxedspace  \newcommand*\gmboxedspace{\hbox{\normalfont{␣}}}

     \gmd@nlperc   \newcommand*\gmd@nlperc[1][]{%
                     \ifhmode\unskip\fi
                     \discretionary{\gmu@tempa}{{\tt\xiipercent\gmboxedspace}}{%
                          \gmu@tempb}%
                     \penalty10000\hskip0sp\relax}

     \gmd@inverb   \newcommand*\gmd@inverb[1][]{%
                     \gmd@nlperc
                     \ifmmode\hbox\else\leavevmode\null\fi
                     \bgroup
                     \ttverbatim
\breakablevisspace   \def\breakablevisspace{%
                       \discretionary{\visiblespace}{\xiipercent\gmboxedspace}{%
                            \visiblespace}}%
     \breakbslash    \def\breakbslash{%
                       \discretionary{}{\xiipercent\gmboxedspace\bslash}{\bslash}}%
     \breaklbrace    \def\breaklbrace{%
                       \discretionary
```

```
                    {\xiilbrace\verbhyphen}%
                    {\xiipercent\gmboxedspace}%
                    {\xiilbrace}}%
                \gm@verb@eol
                \@sverb@chbsl% It's always with visible spaces.
                }
```

```
\nlpercent      \newcommand*\nlpercent{%
\gmu@tempa        \@ifstar{\def\gmu@tempa{{\tt\xiipercent}}%
                    \@emptify\gmu@tempb
                    \gmd@nlperc}%
                  {\@emptify\gmu@tempa
\gmu@tempb          \def\gmu@tempb{\gmboxedspace}%
                    \gmd@nlperc}}
```

```
\incs           \newcommand*\incs{% an inline \cs
\gmu@tempa        \@ifstar{\def\gmu@tempa{{\tt\xiipercent}}%
                    \@emptify\gmu@tempb
                    \gmd@nlperc\cs}%
                  {\@emptify\gmu@tempa
\gmu@tempb          \def\gmu@tempb{\gmboxedspace}%
                    \gmd@nlperc\cs}}
```

```
\inenv          \def\inenv{\incs[]}% 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 css in the inline comment:

```
                \@ifundefined{+}{}{\typeout{^^Jgmdoc.sty:␣redefining␣\bslash+.}}
\+              \def\+{\discre{{\normalfont-}}{{\tt\xiipercent\gmboxedspace}}{}}
```

```
\ds             \@ifundefined{ds}{\def\ds{DocStrip}}{}
```

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          \newcommand*\CDAnd{\CodeDelim\&}
```

```
\CDPerc         \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       \let\division=\section
```

```
\subdivision    \let\subdivision=\subsection
```

```
\subsubdivision \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          \newcounter{gmd@mc}
```

Note it is after the `macrocode` group

\gmd@mchook

```
\def\gmd@mchook{\stepcounter{gmd@mc}%
  \gmd@mcdiag
  \ifcsname␣gmd@mchook\the\c@gmd@mc\endcsname
  \afterfi{\csname␣gmd@mchook\the\c@gmd@mc\endcsname}%
  \fi}
```

\AfterMacrocode

```
\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 expalin 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).

```
\@emptify\gmd@mcdiag
```

\mcdiagOn
\gmd@mcdiag

```
\def\mcdiagOn{\def\gmd@mcdiag{%
    \typeout{^^J\bslash␣end{\@currenvir}␣No.\the\c@gmd@mc
    \space\on@line,␣cln.\the\c@codelinenum.}}}
```

\mcdiagOff

```
\def\mcdiagOff{\@emptify\gmd@mcdiag}
```

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

enumargs

```
\newenvironment*{enumargs}
  {\enumerate
    \@namedef{label\@enumctr}{%
      \cs[]{\#\csname␣the\@enumctr\endcsname␣}}}
  {\endenumerate}
```

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 `\let`s 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 2349.

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

\CharacterTable

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

\foone{%
  \catcode`\[=1␣\catcode`\]=2␣%
  \@makeother\{\@makeother\}}%
[
```

\Character@Table

```
  \def\Character@Table#1{#2}[\endgroup
     \message[^^J^^J␣gmdoc.sty␣package:^^J
     ====␣The␣input␣file␣contains␣the␣\bslash␣CharacterTable.^^J
```

```
    ====␣If␣you␣really␣need␣to␣check␣the␣correctness␣of␣the␣
         chars,^^J
    ====␣please␣notify␣the␣author␣of␣gmdoc.sty␣at␣the␣email␣
         address^^J
    ====␣given␣in␣the␣legal␣notice␣in␣gmdoc.sty.^^J^^J]%
]]
```

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 7402.

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

\DescribeMacro
\DescribeEnv

The \DescribeMacro and \DescribeEnv commands seem to correspond with my \TextUsage macro in its plain and starred version respectively except they don't typeset their arguments in the text i.e., they do two things of the three. So let's \def them to do these two things in this package, too:

\DescribeMacro

```
\outer\def\DescribeMacro{%
  \begingroup\MakePrivateLetters
  \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

```
\outer\def\DescribeEnv{%
  \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

```
\outer\def\Describe{% It doesn't typeset its argument in the point of occurrence.
  \begingroup\MakePrivateLetters
  \@ifstarl{\MakePrivateOthers\Describe@Env}{\Describe@Macro}}
```

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

\Describe@Macro

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

\Describe@Env

```
\def\Describe@Env#1{%
  \endgroup
  \strut\Text@Marginize{#1}%
  \@usgentryze{#1}% we declare the 'usage' kind of formatting the entry and in-
          dex the sequence #1.
  \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

\MacroFont. In my opinion there's a noble tradition of typesetting the TEX code in \tt font nad this tradition sustained should be. If one wants to change the tradition, let him redefine \tt, in TEX 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:

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

\CodeIndent
\MacroIndent

We have provided \CodeIndent in line 2172. And it corresponds with doc's \MacroIndent so

\MacroIndent

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

And similarly the other skips:

\MacrocodeTopsep

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

\MacroTopsep

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

\SpecialEscapechar

```
\let\SpecialEscapechar\CodeEscapeChar
```

\theCodelineNo
\LineNumFont

\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.

```
\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

```
\def\noeffect@info#1{\@ifundefined{#1}{}{\PackageInfo{gmdoc}{^^J%
      The␣\bslash#1␣macro␣is␣not␣supported␣by␣this␣package^^J
      and␣therefore␣has␣no␣effect␣but␣this␣notification.^^J
      If␣you␣think␣it␣should␣have,␣please␣contact␣the␣maintainer^^J
      indicated␣in␣the␣package's␣legal␣note.^^J}}}
```

\PrintDescribeMacro
\PrintMacroName
\PrintDescribeEnv
\PrintEnvName

The four macros formatting the macro and environment names, namely \PrintDescribeMacro, \PrintMacroName, \PrintDescribeEnv and \PrintEnvName are not supported by gmdoc. They seem to me to be too internal to take care of them. Note that in the name of (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.

```
\AtEndInput{%
  \noeffect@info{PrintDescribeMacro}%
  \noeffect@info{PrintMacroName}%
  \noeffect@info{PrintDescribeEnv}%
  \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

```
\def\CodelineNumbered{\AtBeginDocument{\gag@index}}
\@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`

```
\let\CodelineIndex\@pageindexfalse
\@onlypreamble\CodelineIndex
```

```
\let\PageIndex\@pageindextrue
\@onlypreamble\PageIndex
```

The next two declarations I find useful and smart:

\DisableCrossrefs

```
\def\DisableCrossrefs{\@bsphack\gag@index\@esphack}
```

\EnableCrossrefs
\DisableCrossrefs

```
\def\EnableCrossrefs{\@bsphack\ungag@index
  \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

```
\newcommand*\AlsoImplementation{\@bsphack%
  \long\def\StopEventually##1{\gdef\Finale{##1}}% we define \Finale
        just to expand to the argument of \StopEventually not to 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).
    \@esphack}
```

```
\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
\StopEventually

```
\def\OnlyDescription{\@bsphack\long\def\StopEventually##1{%
```

"In this case the argument of `\StopEventually` should be set and afterwards TEX should stop reading from this file. Therefore we finish this macro with"

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

"If no `\StopEventually` command is given we silently ignore a `\Finale` issued."

```
\@relaxen\Finale
```

\meta
\<...>

The `\meta` macro is so beautifully crafted in doc that I couldn't resist copying it into gmutils. It's also available in Knuthian (*The TEXbook* format's) disguise `\<`⟨*the argument*⟩`>`.

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:

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

\SpecialMainEnvIndex

```
\def\SpecialMainEnvIndex{\csname␣CodeDefIndex\endcsname*}% we don't
```

type \DefIndex explicitly here because it's \outer, remember?

\SpecialIndex
```
\let\SpecialIndex=\CodeCommonIndex
```
\SpecialUsageIndex
```
\let\SpecialUsageIndex=\TextUsgIndex
```
\SpecialEnvIndex
```
\def\SpecialEnvIndex{\csname␣TextUsgIndex\endcsname*}
```
\SortIndex
```
\def\SortIndex#1#2{\index{#1\actualchar#2}}
```

"All these macros are usually used by other macros; you will need them only in an emergency."

Therefore I made the assumption(s) that 'Main' indexing macros are used in my 'Code' context and the 'Usage' ones in my 'Text' context.

\verbatimchar Frank Mittelbach in doc provides the \verbatimchar macro to (re)define the \verb(*)'s delimiter for the index entries. The gmdoc package uses the same macro and its default definition is {&}. When you use doc you may have to redefine \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
```
\def\verbatimchar{&}
```

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

*
```
\DeclareRobustCommand*\*{\leavevmode\lower.8ex\hbox{$\,%
    \widetilde{\ }\,$}}
```

\IndexPrologue \IndexPrologue is defined in line 5388. And other doc index commands too.

```
\@ifundefined{main}{}{\let\DefEntry=\main}
```
```
\@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 dont want an error while gmdocumenting doc-prepared files, better let's define doc's counter for the modules' depths.

StandardModuleDepth
```
\newcounter{StandardModuleDepth}
```

For now let's just mark the macro for further development

\DocstyleParms
```
\noeffect@info{DocstyleParms}
```

For possible further development or to notify the user once and forever:

\DontCheckModules
\CheckModules
```
\@emptify\DontCheckModules␣\noeffect@info{DontCheckModules}
\@emptify\CheckModules␣\noeffect@info{CheckModules}
```

\Module The \Module macro *is* provided exactly as in doc.

\AltMacroFont
```
\@emptify\AltMacroFont␣\noeffect@info{AltMacroFont}
```

"And finally the most important bit: we change the \catcode of % so that it is ignored (which is how we are able to produce this document!). We provide two commands to do the actual switching."

\MakePercentIgnore
\MakePercentComment
```
\def\MakePercentIgnore{\catcode`\%9\relax}
\def\MakePercentComment{\catcode`\%14\relax}
```

gmdocing doc.dtx

The author(s) of doc suggest(s):

“For examples of the use of most—if not all—of the features described above consult the doc.dtx source itself.”

Therefore I hope that after doc.dtx has been gmdoc-ed, one can say gmdoc is doc-compatible “at most—if not at all”.

TEXing the original doc with my humble[12] 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 7364).

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

```
7361 \def\OldDocInput{%
7363   \AtBegInputOnce{\StraightEOL
7364     \let\@MakeShortVerb=\old@MakeShortVerb
7366     \OldMacrocodes}%
7367   \bgroup\@makeother\_% it's to allow _ in the filenames. The next macro will
           close the group.
7369   \Doc@Input}
```

We don’t swith 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 ‘Succesfully 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 adjustments too because some crucial differences came out. I’d like to write a word about them now.

The first but not least is that the author(s) of doc give the cs marking commands non-macro arguments sometimes, e.g., `\DescribeMacro{StandardModuleDepth}`. Therefore we should launch the *starred* versions of corresponding gmdoc commands. This means the doc-like commands will not look for the cs’s occurrence in the code but will mark the first codeline met.

Another crucial difference is that in gmdoc the narrative and the code layers are separated with only the code delimiter and therefore may be much more mixed than in doc. among others, the `macro` environment is *not* a typical `verbatim` like: the texts commented out within `macrocode` are considered a normal commentary i.e., not verbatim. Therefore some macros ‘commented out’ to be shown verbatim as an example source must have been ‘additionally’ verbatimized for gmdoc with the shortverb chars e.g. You may also change the code delimiter for a while, e.g., the line

```
7402 %␣\AVerySpecialMacro␣%␣delete␣the␣first␣%␣when...
```

was got with

[12] What a *false* modesty! ;-)

```
\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 2016–2021. 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 2445).

• Some lines of non-printing stuff such as `\Define...` and `\changes` connecting the narration with the code resulted with unexpected large vertical space. Adding a fully blank line between the printed narration text and not printed stuff helped.

• Specifying `codespacesgrey, codespacesblank` results in typesetting all the spaces grey including the leading ones.

• About the DocStrip verbatim mode directive see above.

(No) ⟨*eof*⟩

Until version 0.99i a file that is `\DocInput` had to be ended with a comment line with an `\EOF` or `\NoEOF` cs that suppressed the end-of-file character to make input end properly. Since version 0.99i however the proper ending of input is acheved 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 below[14]:

```
\foone{\catcode`\^^M\active␣}{%
  \def\@NoEOF#1^^M{%
```

\@NoEOF

[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.

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

```
          \@relaxen\EOFMark\endinput}%
\@EOF     \def\@EOF#1^^M{\endinput}}

\NoEOF  \def\NoEOF{\QueerEOL\@NoEOF}
\EOF    \def\EOF{\QueerEOL\@EOF}
```

As you probably see, `\(No)EOF` have the 'immediate' `\endinput` effect: the file ends even in the middle of a line, the stuff after `\(No)EOF` will be gobbled unlike with a bare `\endinput`.

```
\endinput
⟨/package⟩
```

b. The gmdocc Class For gmdoc Driver Files[1]

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

```
\NeedsTeXFormat{LaTeX2e}
\ProvidesClass{gmdocc}
            [2008/08/03␣v0.79␣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 (LA)TEX 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 lmodern 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 lmodern 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 (LA)TEX 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 lmodern 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.

noindex

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

nochanges

[1] This file has version number v0.79 dated 2008/08/03.

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 LaTeX writers don't use it in general and gmdoc.sty *does* make some use of it.

debug This class provides also the debug option. It turns the \if@debug Boolean switch True and loads the trace package that was a great help to me while debugging gmdoc.sty.

The default base document class loaded by gmdocc.cls is Marcin Woliński mwart. If you have not installed it on your computer, the standard article will be used.

Moreover, if you like MW's classes (as I do) and need \chapter (for multiple files' input e.g.), you may declare another mwcls with the option homonimic with the class'es
mwrep name: mwrep for mwrep and mwbk for mwbk. For the symmetry there's also mwart option
mwbk (equivalent to the default setting).
mwart The existence test is done for any MW class option as it is in the default case.

Since version 0.99g (November 2007) the bundle goes XeTeX and that means you can
sysfonts use the system fonts if you wish, just specify the sysfonts option and the three basic XeTeX-related packages (fontspec, xunicode and xltxtra) will be loaded and then you can specify fonts with the fontspec declarations. For use of them check the driver of this documentation where the TeX 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

```
\RequirePackage{xkeyval}
```

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

\gm@DOX
\gm@EOX

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

```
\newif\ifgmcc@mwcls
```

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

class
\gmcc@CLASS

```
\gm@DOX{class}{% the default will be Marcin Woliński class (mwcls) analogous to
        article, see line 257.
  \def\gmcc@CLASS{#1}%
  \@for\gmcc@resa:=mwart,mwrep,mwbk\do␣{%
    \ifx\gmcc@CLASS\gmcc@resa\gmcc@mwclstrue\fi}%
}
```

mwart

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

mwrep
```
165 \gm@DOX{mwrep}{\gmcc@class{mwrep}}% If you need chapters, this option chooses
        an MW class that corresponds to report,
```
mwbk
```
169 \gm@DOX{mwbk}{\gmcc@class{mwbk}}% and this MW class corresponds to book.
```
article
```
172 \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 occured: 'undefined control sequence
        % \XKV@resa␣->␣\@nil'.
```
outeroff
```
180 \gm@DOX{outeroff}{\let\outer\relax}% This option allows \outer-prefixed
        macros to be gmdoc-processed with all the bells and whistles.
```
\if@debug
```
184 \newif\if@debug
```
debug
```
186 \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
```
191 \gm@DOX{noindex}{%
192   \PassOptionsToPackage{noindex}{gmdoc}}% This option turns the writing
          outto .idx file off.
```
\if@gmccnochanges
```
196 \newif\if@gmccnochanges
```
nochanges
```
198 \gm@DOX{nochanges}{\@gmccnochangestrue}% This option turns the writing outto
        .glo file off.
```
gmeometric
```
202 \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 XƎTEX 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 XƎTEX. 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 XƎTEX. That means that if XƎTEX is detected, we may load the fontspec package and the other two of basic three XƎTEX-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
sysfonts
```
221 \newif\ifgmcc@oldfonts
222 \gmcc@oldfontstrue
223 \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
\mpttversion
```
232 \gm@DOX{mptt}[17]{\def\mpttversion{#1}}% the default value (17) works if the
        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
```
237 \def\gmcc@setfont#1{%
238   \gmcc@oldfontsfalse% note that if we are not in XƎTEX, this switch will be turned
          true in line 304
240   \AtBeginDocument{%
```

```
    \@ifXeTeX{%
      \defaultfontfeatures{Numbers={OldStyle,Proportional}}%
      \setmainfont[Mapping=tex-text]{#1}%
      \setsansfont[Mapping=tex-text,␣Scale=MatchLowercase]{Latin␣Modern␣Sans}%
        \setmonofont[Scale=MatchLowercase]{Latin␣Modern␣Mono}%
      \let\sl\it␣\let\textsl\textit
    }{}}%
}
```

minion

```
\gm@DOX{minion}{\gmcc@setfont{Minion␣Pro}}
```

pagella

```
\gm@DOX{pagella}{\gmcc@setfont{TeX␣Gyre␣Pagella}%
```

\gmcc@PAGELLA

```
  \def\gmcc@PAGELLA{1}%
}
```

```
\gm@EOX{class=mwart}% We set the default basic class to be mwart.
```

```
\gm@EOX{mptt=0}% We default to set the marginpar typewriter font to OpenType LMTT LC.
```

```
\DeclareOptionX*{\PassOptionsToPackage{\CurrentOption}{gmdoc}}
```

```
\ProcessOptionsX[gmcc]<>
```

```
\ifgmcc@mwcls
  \IfFileExists{\gmcc@CLASS.cls}{}{\gmcc@mwclsfalse}% As announced, we do the ontological test to any mwcls.
\fi
\ifgmcc@mwcls
  \XKV@ifundefined{XeTeXdefaultencoding}{}{%
    \XeTeXdefaultencoding␣"cp1250"}% mwcls are encoding-sensitive because MW uses Polish diacritics in the commentaries.
  \LoadClass[fleqn,␣oneside,␣noindentfirst,␣11pt,␣withmarginpar,
  sfheadings]{\gmcc@CLASS}%
  \XKV@ifundefined{XeTeXdefaultencoding}{}{%
    \XeTeXdefaultencoding␣"utf-8"}%
\else
  \LoadClass[fleqn,␣11pt]{article}% Otherwise the standard article is loaded.
\fi
```

```
\RequirePackage{gmutils}[2008/08/09]% earlier to provide \@ifXeTeX.
```

```
\ifgmcc@mwcls\afterfi\ParanoidPostsec\fi
```

```
\@ifXeTeX{}{\gmcc@oldfontstrue}
```

```
\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 `\[...\]`:

`\[|\verbatim\stuff|\].`

```
\ifgmcc@oldfonts
  \IfFileExists{lmodern.sty}{% We also examine the ontological status of this package
    \RequirePackage{lmodern}% and if it shows to be satisfactory (the package shows to be), we load it and set the proper font encoding.
    \RequirePackage[T1]{fontenc}%
```

```
                 }{}%
```

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

```
\agrave            \def\agrave␣␣{\`a}%
\cacute            \def\cacute␣␣{\'c}%
\eacute            \def\eacute␣␣{\'e}%
\idiaeres          \def\idiaeres{\"\i}%
\nacute            \def\nacute␣␣{\'n}%
\ocircum           \def\ocircum␣{\^o}%
\oumlaut           \def\oumlaut␣{\"o}%
\uumlaut           \def\uumlaut␣{\"u}%
                 \else% this case happens only with XeTeX.
                   \let\do\relaxen
                   \do\Finv\do\Game\do\beth\do\gimel\do\daleth% these five caused the 'already defined' error.
                   \let\@zf@euenctrue\zf@euencfalse
                   \XeTeXthree
\agrave            \def\agrave␣␣{\char"00E0␣}%
\cacute            \def\cacute␣␣{\char"0107␣}% Note the space to be sure the number ends here.
\eacute            \def\eacute␣␣{\char"00E9␣}%
\idiaeres          \def\idiaeres{\char"00EF␣}%
\nacute            \def\nacute␣␣{\char"0144␣}%
\oumlaut           \def\oumlaut␣{\char"00F6␣}%
\uumlaut           \def\uumlaut␣{\char"00FC␣}%
\ocircum           \def\ocircum␣{\char"00F4␣}%
                   \AtBeginDocument{%
\ae                  \def\ae{\char"00E6␣}%
                     \def\l␣{\char"0142␣}%
\oe                  \def\oe{\char"0153␣}%
                   }%
                 \fi
```

Now we set the page layout.

```
                 \RequirePackage{gmeometric}
\gmdoccMargins   \def\gmdoccMargins{%
                   \geometry{top=77pt,␣height=687pt,␣% =53 lines but the lines option seems not to work 2007/11/15 with TeX Live 2007 and XeTeX 0.996-patch1
                     left=4cm,␣right=2.2cm}}
                 \gmdoccMargins
```

```
                 \if@debug% For debugging we load also the trace package that was very helpful to me.
                   \RequirePackage{trace}%
                   \errorcontextlines=100␣% And we set an error info parameter.
                 \fi
```

```
\ifdtraceon      \newcommand*\ifdtraceon{\if@debug\afterfi\traceon\fi}
\ifdtraceoff     \newcommand*\ifdtraceoff{\if@debug\traceoff\fi}
```

We load the core package:

```
                 \RequirePackage{gmdoc}
```

```
                 \ifgmcc@oldfonts
                 \@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       \def\marginpartt{\normalfont\fontseries{lc}\ttfamily}}{}%
               \else
\marginpartt     \def\marginpartt{\fontspec{LMTypewriter10␣LightCondensed}}%
               \fi

               \ifnum1=0\csname␣gmcc@PAGELLA\endcsname\relax
                 \RequirePackage{pxfonts,tgpagella,qpxmath}%
               \fi

               \raggedbottom

               \setcounter{secnumdepth}{0}% We wish only the parts and chapters to be num-
                    bered.

\thesection    \renewcommand*\thesection{\arabic{section}}% isn't it redundant at the above
                    setting?

               \@ifnotmw{}{%
                 \@ifclassloaded{mwart}{% We set the indentation of Contents:
                   \SetTOCIndents{{}{\quad}{\quad}{\quad}{\quad}{\quad}{\quad}}}{%
                        % for mwart
                   \SetTOCIndents{{}{\bf9.\enspace}{\quad}{\quad}{\quad}{\quad}{%
                        \quad}}}% and for the two other mwclss.
                 \pagestyle{outer}}% We set the page numbers to be printed in the outer and
                      bottom corner of the page.

\titlesetup    \def\titlesetup{\bfseries\sffamily}% We set the title(s) to be boldface and
                    sans serif.

               \if@gmccnochanges\let\RecordChanges\relax\fi% If the nochanges option is
                    on, we discard writing outto the .glo file.

               \RecordChanges% We turn the writing the \changes outto the .glo file if not the
                    above.

               \dekclubs% We declare the club sign | to be a shorthand for \verb*.
               \edverbs% to redefine \[ so that it puts a shortverb in a \hbox.
               \smartunder% and we declare the _ char to behave as usual in the math mode and
                    outside math to be just an uderscore.

               \exhyphenpenalty\hyphenpenalty% 'cause mwcls set it =10000 due to Polish cus-
                    toms.

               \RequirePackage{amssymb}
\EOFMark       \def\EOFMark{\rightline{\ensuremath{\square}}}

               \endinput
```

c. The gmutils Package[1]

Written by Grzegorz Murzynowski,
natror at o2 dot pl

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

```
\NeedsTeXFormat{LaTeX2e}
\ProvidesPackage{gmutils}
    [2008/08/07␣v0.92␣some␣rather␣TeXnical␣macros,␣some␣of␣them␣
        tricky␣(GM)]
```

Intro

The gmutils.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 (LA)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 gmutils.zip Archive

The distribution of the gmutils package consists of the following four files and a TDS-compliant archive.

gmutils.sty
README
gmutilsDoc.tex
gmutilsDoc.pdf
gmutils.tds.zip

```
\ifx\XeTeXversion\relax
\let\XeTeXversion\@undefined%
```

If someone earlier used the `\@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

[1] This file has version number v0.92 dated 2008/08/07.

and it eats its arguments, freezing their catcodes, which is not what we want in line 2788

```
\fi
```

```
\ifdefined\XeTeXversion
\XeTeXinputencoding␣utf-8␣% we use Unicode dashes later in this file.
\fi% and if we are not in XƎTEX, we skip them thanks to XƎTEX-test.
```

A couple of abbreviations

\@xa \@nx

```
\let\@xa\expandafter
\let\@nx\noexpand
```

The \newgif declaration's effect is used even in the LATEX 2$_\varepsilon$ 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 LATEX'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

```
\protected\def\newgif#1{%
  {\escapechar\m@ne
    \global\let#1\iffalse
    \@gif#1\iftrue
    \@gif#1\iffalse
  }}
```

'Almost' is also in the detail that in this case, which deals with \global assignments, we don't have to bother with storing and restoring the value of \escapechar: we can do all the work inside a group.

\@gif

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

```
\protected\def\newif#1{% We not only make \newif \protected but also make
          it to define \protected assignments so that premature expansion doesn't
          affect \if…\fi nesting.
  \count@\escapechar␣\escapechar\m@ne
  \let#1\iffalse
  \@if#1\iftrue
  \@if#1\iffalse
  \escapechar\count@}
```

\@if

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

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 LaTeX approach.

\grefstepcounter

```
\protected\def\grefstepcounter#1{%
  {\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:

\hgrefstepcounter

```
\protected\def\hgrefstepcounter#1{%
  \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

```
\protected\def\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 `\expandafter`s e.g.

\glet

```
\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

```
\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

```
\long\def\addto@macro#1#2{%
  \toks@\@xa{#1#2}%
  \edef#1{\the\toks@}%
}% (\toks@ is a scratch register, namely \toks0.)
```

And for use in the very document,

\addtomacro

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

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

\prependtomacro

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

Note that `\prependtomacro` can be prefixed.

\addtotoks

```
\long\def\addtotoks#1#2{%
  #1=\@xa{\the#1#2}}
```

\@emptify
\emptify

```
\newcommand*\@emptify[1]{\let#1=\@empty}
\@ifdefinable\emptify{\let\emptify\@emptify}
```

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

\g@emptify \gemptify

```
\newcommand*\g@emptify{\global\@emptify}
\@ifdefinable\gemptify{\let\gemptify\g@emptify}
```

\@relaxen \relaxen

```
\newcommand\@relaxen[1]{\let#1=\relax}
\@ifdefinable\relaxen{\let\relaxen\@relaxen}
```

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

\g@relaxen \grelaxen

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

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

```
\def\gmshowlists{\showboxdepth=1000␣\showboxbreadth=1000␣%
      \showlists}
```

\nameshow \nameshowthe

```
\newcommand\nameshow[1]{\@xa\show\csname#1\endcsname}
\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 $_{10}$. In gmdoc I needed the spaces in macros' and environments' names to be always $_{12}$, so I define

\xiistring

```
\def\xiistring#1{%
  \if\@nx#1\xiispace
    \xiispace
  \else
    \string#1%
  \fi}
```

\@ifnextcat, \@ifnextac

As you guess, we \def \@ifnextcat à la \@ifnextchar, see LaTeX 2_ε 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

```
\long\def\@ifnextcat#1#2#3{%
  \def\reserved@d{#1}%
  \def\reserved@a{#2}%
  \def\reserved@b{#3}%
  \futurelet\@let@token\@ifncat}
```

\@ifncat

```
\def\@ifncat{%
  \ifx\@let@token\@sptoken
    \let\reserved@c\@xifncat
  \else
    \ifcat\reserved@d\@nx\@let@token
```

```
      \let\reserved@c\reserved@a
    \else
      \let\reserved@c\reserved@b
    \fi
  \fi
  \reserved@c}
```

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

```
\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

```
\long\def\@ifnextif#1#2#3{%
  \def\reserved@d{#1}%
  \def\reserved@a{#2}%
  \def\reserved@b{#3}%
  \futurelet\@let@token\@ifnif}
```

\@ifnif

```
\def\@ifnif{%
  \ifx\@let@token\@sptoken
    \let\reserved@c\@xifnif
  \else
    \if\reserved@d\@nx\@let@token
      \let\reserved@c\reserved@a
    \else
      \let\reserved@c\reserved@b
    \fi
  \fi
  \reserved@c}
```

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

```
\long\def\@ifnextac#1#2{%
  \@ifnextcat\bgroup{#2}{\gm@ifnac{#1}{#2}}}
```

\gm@ifnac

```
\long\def\gm@ifnac#1#2#3{%
  \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 $\neq 1$. (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 TEXbook* appendix D) since `\let`'s syntax includes one optional space after =.

```
\let\gmu@reserveda\*%
```

```
\*              \def\*{%
                  \let\*\gmu@reserveda
                  \let\gm@letspace=␣}%
                \*␣%
\@ifnextspace   \def\@ifnextspace#1#2{%
                  \let\gmu@reserveda\*%
\*                \def\*{%
                    \let\*\gmu@reserveda
                    \ifx\@let@token\gm@letspace\afterfi{#1}%
                    \else\afterfi{#2}%
                    \fi}%
                  \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.

\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 TeX Guru.

I think the situations when the Knuthian (the former) trick is not available are rather seldom, but they are imaginable at least: the \next trick involves an assignment so it won't work e.g. in \edef. But in general it's only a matter of taste which one to use.

One warning: those macros peel the braces off, i.e.,

```
\if..\afterfi{\@makeother\^^M}\fi
```

causes a leakage of ^^M$_{12}$. To avoid pollution write

```
\if..\afterfi{\bgroup\@makeother\^^M\egroup}\fi.
```

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

And two more of that family:

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

Notice the refined elegance of those macros, that cover both 'then' and 'else' cases thanks to #2 that is discarded.

```
\afterififfififi  \long\def\afterififfififi#1#2\fi#3\fi#4\fi{\fi#1}
\afteriffififi    \long\def\afteriffififi#1#2\fi#3\fi#4\fi{\fi\fi#1}
\afterfififi      \long\def\afterfififi#1#2\fi#3\fi#4\fi{\fi\fi\fi#1}
```

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 LaTeX'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

```
537 \long\def\@begnamedgroup#1{%
538   \@ignorefalse% not to ignore blanks after group
539   \begingroup\@endpefalse
540   \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 590.
544   \edef\@currenvline{\on@line}%
545   \csname␣#1\endcsname}% if the argument is a command's name (an environ-
          ment's e.g.), this command will now be executed. (If the corresponding
          control sequence hasn't been known to TeX, this line will act as \relax.)
```

For back compatibility with my earlier works

\bnamegroup

```
553 \let\bnamegroup\@begnamedgroup
```

And for the ending

\enamegroup

```
555 \def\enamegroup#1{\end{#1}}
```

And we make it the starred version of \begin.

\begin*
\begin

```
561 \def\begin{\@ifstar{\@begnamedgroup}{%
562      \@begnamedgroup@ifcs}}
```

\@begnamedgroup@ifcs

```
565 \def\@begnamedgroup@ifcs#1{%
566   \ifcsname#1\endcsname\afterfi{\@begnamedgroup{#1}}%
567   \else\afterfi{\@latex@error{Environment␣#1␣undefined}\@eha}%
568   \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 LaTeX 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 char catcodes are equivalent. 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.

\@ifenvir

```
590 \long\def\@ifenvir#1#2#3{%
592   \edef\gmu@reserveda{\@xa\string\csname\@currenvir\endcsname}%
593   \edef\gmu@reservedb{\@xa\string\csname#1\endcsname}%
```

```
  \ifx\gmu@reserveda\gmu@reservedb\afterfi{#2}%
  \else\afterfi{#3}%
  \fi}
```

\@checkend

```
\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*}.
```

Of course, you might 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, LaTeX 2$_\varepsilon$ version of these macros was written by Donald Arseneau `asnd@triumf.ca` and Matt Swift `swift@bu.edu` after the LaTeX 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
\smaller
\larger
\smallerr
\largerr

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.

(Since `\DeclareRobustCommand` doesn't issue an error if its argument has been defined and it only informs about redefining, loading relsize remains allowed.)

\relsize

```
\DeclareRobustCommand*\relsize[1]{%
  \ifmmode␣\@nomath\relsize\else
    \begingroup
     \@tempcnta␣% assign number representing current font size
       \ifx\@currsize\normalsize␣4\else␣␣␣% funny order is to have most
            ...
        \ifx\@currsize\small␣3\else␣␣␣␣␣␣␣% ...likely sizes checked first
         \ifx\@currsize\footnotesize␣2\else
          \ifx\@currsize\large␣5\else
           \ifx\@currsize\Large␣6\else
            \ifx\@currsize\LARGE␣7\else
             \ifx\@currsize\scriptsize␣1\else
              \ifx\@currsize\tiny␣0\else
               \ifx\@currsize\huge␣8\else
                \ifx\@currsize\Huge␣9\else
                4\rs@unknown@warning␣% unknown state: \normalsize as
                     starting point
     \fi\fi\fi\fi\fi\fi\fi\fi\fi\fi
```

Change the number by the given increment:

```
     \advance\@tempcnta#1\relax
```

watch out for size underflow:

```
      \ifnum\@tempcnta<\z@␣\rs@size@warning{small}{\string\tiny}%
           \@tempcnta\z@␣\fi
      \@xa\endgroup
      \ifcase\@tempcnta␣␣% set new size based on altered number
         \tiny␣\or␣\scriptsize␣\or␣\footnotesize␣\or␣\small␣\or␣%
              \normalsize␣\or
         \large␣\or␣\Large␣\or␣\LARGE␣\or␣\huge␣\or␣\Huge␣\else
          \rs@size@warning{large}{\string\Huge}\Huge
\fi\fi}% end of \relsize.
```

\rs@size@warning

```
\providecommand*\rs@size@warning[2]{\PackageWarning{gmutils␣(relsize)}{%
 Size␣requested␣is␣too␣#1.\MessageBreak␣Using␣#2␣instead}}
```

\rs@unknown@warning

```
\providecommand*\rs@unknown@warning{\PackageWarning{gmutils␣(relsize)}{Current␣font␣size
 is␣unknown!␣(Why?!?)\MessageBreak␣Assuming␣\string\normalsize}}
```

And a handful of shorthands:

\larger
\smaller
\textlarger
\textsmaller
\largerr
\smallerr

```
\DeclareRobustCommand*\larger[1][\@ne]{\relsize{+#1}}
\DeclareRobustCommand*\smaller[1][\@ne]{\relsize{-#1}}
\DeclareRobustCommand*\textlarger[2][\@ne]{{\relsize{+#1}#2}}
\DeclareRobustCommand*\textsmaller[2][\@ne]{{\relsize{-#1}#2}}
\DeclareRobustCommand*\largerr{\relsize{+2}}
\DeclareRobustCommand*\smallerr{\relsize{-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 TEX's eyes', as my TEX Guru taught me.

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

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

\foone

```
\long\def\foone#1{\bgroup#1\egroupfirstofone}
\long\def\egroupfirstofone#1{\egroup#1}
```

\fooatletter

```
\long\def\fooatletter{\foone\makeatletter}
```

And this one is defined, I know, but it's not \long with the standard definition.

\gobble
\gobbletwo

```
\long\def\gobble#1{}
\let\@gobble\gobble
\let\gobbletwo\@gobbletwo
```

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 \chardefed.

\subs

```
\foone{\catcode`\_=8␣}%
{\let\subs=_}
```

\xiiunder

```
\foone{\@makeother\_}%
{\def\xiiunder{_}}
```

```
\ifdefined\XeTeXversion
```

\xiiunder

```
  \def\xiiunder{\char"005F␣}%
  \let\_\xiiunder
\fi
```

```
\foone{\catcode`\[=1␣\@makeother\{
  \catcode`\]=2␣\@makeother\}}%
[%
```

\xiilbrace
\xiirbrace

```
  \def\xiilbrace[{]%
  \def\xiirbrace[}]%
]% of \firstofone
```

Note that LaTeX's \@charlb and \@charrb 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 $\_{}_{12}$ ('other') outside math.

```
\foone{\catcode`\_=\active}
{%
```

\smartunder

```
  \newcommand*\smartunder{%
    \catcode`\_=\active
    \def_{\ifmmode\subs\else\_\fi}}}% We define it as \_ not just as \xiiunder
```

because some font encodings don't have _ at the \char`_ position.

```
\foone{\catcode`\!=0
  \@makeother\\}
```

\xiibackslash

```
{!newcommand*!xiibackslash{\}}
```

\bslash

```
\let\bslash=\xiibackslash
```

```
\foone{\@makeother\%}
```

\xiipercent

```
{\def\xiipercent{%}}
```

```
\foone{\@makeother\&}%
```

\xiiand

```
{\def\xiiand{&}}
```

```
\foone{\@makeother\ }%
```

\xiispace

```
{\def\xiispace{␣}}
```

We introduce \visiblespace from Will Robertson's xltxtra if available. It's not sufficient \@ifpackageloaded{xltxtra} since \xxt@visiblespace 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.

```
\AtBeginDocument{%
  \ifdefined\xxt@visiblespace
    \let\visiblespace\xxt@visiblespace
  \else
    \let\visiblespace\xiispace
  \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 hypenation by defining a \language which has no patterns associated with it and use this to typeset the words within the angle brackets."

\meta

```
\DeclareRobustCommand*\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."

```
  \ensuremath\langle
  \ifmmode␣\@xa␣\nfss@text␣\fi
  {%
    \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.

```
    #1\/%
  }\ensuremath\rangle
}
```

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

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

The below \meta's drag[2] is a version of *The TEXbook*'s one.

\<...>

```
\def\<#1>{\meta{#1}}
```

Macros for Printing Macros and Filenames

First let's define three auxiliary macros analogous to \dywiz from polski.sty: a shorthands 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
\discret

```
\def\discre#1#2#3{\leavevmode\kernosp%
  \discretionary{#1}{#2}{#3}\penalty10000\hskiposp\relax}
\def\discret#1{\leavevmode\kernosp%
  \discretionary{#1}{#1}{#1}\penalty10000\hskiposp\relax}
```

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

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

\vs

```
\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\catcodeing has no effect).

\printspaces
\gm@pswords

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

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

```
                          \ifx\relax#1\relax\else#1\fi
                          \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                  \DeclareRobustCommand*\sfname[1]{\textsf{\printspaces{#1}}}
\gmu@discretionaryslash  \def\gmu@discretionaryslash{\discre{/}{\hbox{}}{/}}% the second pseudo-
                              argument nonempty to get \hyphenpenalty not \exhyphenpenalty.
\file                    \DeclareRobustCommand*\file[1]{\gmu@printslashes#1/%
                               \gmu@printslashes}
\gmu@printslashes        \def\gmu@printslashes#1/#2\gmu@printslashes{%
                          \sfname{#1}%
                          \ifx\gmu@printslashes#2\gmu@printslashes
                          \else
                          \textsf{\gmu@discretionaryslash}%
                          \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                      \DeclareRobustCommand*{\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                      \DeclareRobustCommand*{\cs}[2][\bslash]{{%
\-                             \def\-{\discretionary{{\rmfamily-}}{}{}}%
                               \def\{{\char`\{}\def\}{\char`\}}\ttfamily␣#1#2}}
\env                     \DeclareRobustCommand*{\env}[1]{\cs[]{#1}}
```

And for the special sequences like ^^A:

```
                         \foone{\@makeother\^}
\hathat                   {\DeclareRobustCommand*\hathat[1]{\cs[^^]{#1}}}
```

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

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

\marg{text} prints {⟨*text*⟩}, 'mandatory argument'.

```
\marg                    \def\marg#1{{\ttfamily\char`\{}\meta{#1}{\ttfamily\char`\}}}
```

\oarg{text} prints [⟨*text*⟩], 'optional argument'. Also \oarg[text] does that.

```
\oarg                    \def\oarg{\@ifnextchar[\@oargsq\@oarg}
\@oarg                   \def\@oarg#1{{\ttfamily[}\meta{#1}{\ttfamily]}}
\@oargsq                 \def\@oargsq[#1]{\@oarg{#1}}
```

\parg{te,xt} prints (⟨*te,xt*⟩), 'picture mode argument'.

```
\parg                    \def\parg{\@ifnextchar(\@pargp\@parg}
        \@parg  959 \def\@parg#1{{\ttfamily(}\meta{#1}{\ttfamily)}}
       \@pargp  960 \def\@pargp(#1){\@parg{#1}}
```

But we can have all three in one command.

```
                964 \AtBeginDocument{%
          \arg  965   \let\math@arg\arg
          \arg  966   \def\arg{\ifmmode\math@arg\else\afterfi{%
                967       \@ifnextchar[%
                968       \@oargsq{\@ifnextchar(%
                969         \@pargp\marg}}\fi}%
                970 }
```

Storing and Restoring the Meanings of CSs

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

```
\ifgmu@SMglobal  976 \newif\ifgmu@SMglobal
      \SMglobal  978 \def\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 meanig back (when grouping is not recommended):

```
    \StoreMacro  989 \def\StoreMacro{%
                 990   \bgroup\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 1113.

```
  \egStore@Macro  995 \long\def\egStore@Macro#1{\egroup\Store@Macro{#1}}
\egStore@MacroSt  996 \long\def\egStore@MacroSt#1{\egroup\Store@MacroSt{#1}}
    \Store@Macro  998 \long\def\Store@Macro#1{%
                  999   \escapechar92
                 1000   \ifgmu@SMglobal\afterfi\global\fi
                 1001   \@xa\let\csname␣/gmu/store\string#1\endcsname#1%
                 1002   \global\gmu@SMglobalfalse}
  \Store@MacroSt 1005 \long\def\Store@MacroSt#1{%
                 1006   \edef\gmu@smtempa{%
                 1007     \ifgmu@SMglobal\global\fi
                 1008     \@nx\let\@xa\@nx\csname/gmu/store\bslash#1\endcsname% we add back-
                              slash because to ensure compatibility between \(Re)StoreMacro and
                              \(Re)StoreMacro*, that is. to allow writing e.g. \StoreMacro\kitten
                              and then \RestoreMacro*{kitten} to restore the meaning of \kitten.
                 1013     \@xa\@nx\csname#1\endcsname}
                 1014   \gmu@smtempa
                 1015   \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 CSs and stores each of them. The CS may be separated with commas but they don't have to.

```
\StoreMacros         1031 \long\def\StoreMacros{\bgroup\makeatletter\Store@Macros}
\Store@Macros        1032 \long\def\Store@Macros#1{\egroup
                     1033   \gmu@setsetSMglobal
                     1034   \let\gml@StoreCS\Store@Macro
                     1035   \gml@storemacros#1.}

\gmu@setsetSMglobal  1038 \def\gmu@setsetSMglobal{%
                     1039   \ifgmu@SMglobal
                     1040     \let\gmu@setSMglobal\gmu@SMglobaltrue
                     1041   \else
                     1042     \let\gmu@setSMglobal\gmu@SMglobalfalse
                     1043   \fi}
```

And the inner iterating macro:

```
\gml@storemacros     1046 \long\def\gml@storemacros#1{%
\gmu@reserveda       1047   \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.
                     1050   \if\gmu@reserveda.% a dot finishes storing.
                     1051     \global\gmu@SMglobalfalse
                     1052   \else
                     1053     \if\gmu@reserveda,% The list this macro is put before may contain commas
                                   and that's O.K., we just continue the work.
                     1055       \afterfifi\gml@storemacros
                     1056     \else% what is else this shall be stored.
                     1057       \gml@StoreCS{#1}% we use a particular CS to may \let it both to the storing
                                       macro as above and to the restoring one as below.
                     1060       \afterfifi{\gmu@setSMglobal\gml@storemacros}%
                     1061     \fi
                     1062   \fi}
```

And for the restoring

```
\RestoreMacro        1069 \def\RestoreMacro{%
                     1070   \bgroup\makeatletter\@ifstar\egRestore@MacroSt\egRestore@Macro}

\egRestore@Macro     1072 \long\def\egRestore@Macro#1{\egroup\Restore@Macro{#1}}
\egRestore@MacroSt   1073 \long\def\egRestore@MacroSt#1{\egroup\Restore@MacroSt{#1}}

\Restore@Macro       1075 \long\def\Restore@Macro#1{%
                     1076   \escapechar92
                     1077   \ifgmu@SMglobal\afterfi\global\fi
                     1078   \@xa\let\@xa#1\csname␣/gmu/store\string#1\endcsname
                     1079   \global\gmu@SMglobalfalse}

\Restore@MacroSt     1081 \long\def\Restore@MacroSt#1{%
                     1082   \edef\gmu@smtempa{%
                     1083     \ifgmu@SMglobal\global\fi
                     1084     \@nx\let\@xa\@nx\csname#1\endcsname
                     1085     \@xa\@nx\csname/gmu/store\bslash#1\endcsname}% cf. the commentary
                                   in line 1008.
                     1087   \gmu@smtempa
                     1088   \global\gmu@SMglobalfalse}

\RestoreMacros       1091 \long\def\RestoreMacros{\bgroup\makeatletter\Restore@Macros}
```

\Restore@Macros

```
\long\def\Restore@Macros#1{\egroup
  \gmu@setsetSMglobal
  \let\gml@StoreCS\Restore@Macro% we direct the core CS towards restoring
          and call the same iterating macro as in line 1035.
  \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
\Stored@Macro

```
\def\StoredMacro{\bgroup\makeatletter\Stored@Macro}
\long\def\Stored@Macro#1{\egroup\Restore@Macro#1#1}
```

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

\storedcsname

```
\def\storedcsname#1{%
  \csname␣/gmu/store\bslash#1\endcsname}
```

2008/08/03 we need to store also an environment.

\StoreEnvironment

```
\def\StoreEnvironment#1{%
  \StoreMacro*{#1}\StoreMacro*{end#1}}
```

\RestoreEnvironment

```
\def\RestoreEnvironment#1{%
  \RestoreMacro*{#1}\RestoreMacro*{end#1}}
```

It happended (see the definition of \@docinclude in gmdoc.sty) that I needed to \relax a bunch of macros and restore them after some time. Because the macros were rather numerous and I wanted the code more readable, I wanted to \do them. After a proper defining of \do of course. So here is this proper definition of \do, provided as a macro (a declaration).

\StoringAndRelaxingDo

```
\long\def\StoringAndRelaxingDo{%
  \gmu@SMdo@setscope
  \long\def\do##1{%
    \gmu@SMdo@scope
    \@xa\let\csname␣/gmu/store\string##1\endcsname##1%
    \gmu@SMdo@scope\let##1\relax}}
```

\gmu@SMdo@setscope

```
\def\gmu@SMdo@setscope{%
  \ifgmu@SMglobal\let\gmu@SMdo@scope\global
  \else\let\gmu@SMdo@scope\relax
  \fi
  \global\gmu@SMglobalfalse}
```

And here is the counter-definition for restore.

\RestoringDo

```
\long\def\RestoringDo{%
  \gmu@SMdo@setscope
  \long\def\do##1{%
    \gmu@SMdo@scope
    \@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 becasuse the latter is defined in Till Tantau's beamer class another way) (both arguments should be text):

\n@melet

```
\def\n@melet#1#2{%
  \edef\gmu@nl@reserveda{%
```

```
    \let\@xa\@nx\csname#1\endcsname
    \@xa\@nx\csname#2\endcsname}%
  \gmu@nl@reserveda}
```

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

\gn@melet

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

```
\newcommand\not@onlypreamble[1]{{%
  \def\do##1{\ifx#1##1\else\@nx\do\@nx##1\fi}%
  \xdef\@preamblecmds{\@preamblecmds}}}
```

```
\not@onlypreamble\@preamblecmds
\not@onlypreamble\@ifpackageloaded
\not@onlypreamble\@ifclassloaded
\not@onlypreamble\@ifl@aded
\not@onlypreamble\@pkgextension
```

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

\gm@notprerr

```
\def\gm@notprerr{␣can␣be␣used␣only␣in␣preamble␣(\on@line)}
```

\@nx
\@nx

```
\AtBeginDocument{%
  \def\do#1{\@nx\do\@nx#1}%
  \edef\@preamblecmds{%
    \def\@nx\do##1{%
      \def##1{\@nx\PackageError{gmutils/LaTeX}%
        {\@nx\string##1␣\@nx\gm@notprerr}\@nx\@eha}}%
    \@preamblecmds}}
```

A subtle error raises: the LaTeX 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 is \nocite that checks \ifx\@onlypreamble\document. So hoping this is a rare problem, we circumvent in with

\nocite

```
\def\nocite#1{%
  \@bsphack{\setbox0=\hbox{\cite{#1}}}\@esphack}
```

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

```
\newcounter{gm@PronounGender}
```

\gm@atppron

```
\newcommand*\gm@atppron[2]{%
  \stepcounter{gm@PronounGender}% remember \stepcounter is global.
  \ifodd\value{gm@PronounGender}#1\else#2\fi}
```

\heshe \hisher \himher \hishers

```
\newcommand*\heshe{\gm@atppron{he}{she}}
\newcommand*\hisher{\gm@atppron{his}{her}}
\newcommand*\himher{\gm@atppron{him}{her}}
\newcommand*\hishers{\gm@atppron{his}{hers}}
```

\HeShe \HisHer \HimHer \HisHers

```
\newcommand*\HeShe{\gm@atppron{He}{She}}
\newcommand*\HisHer{\gm@atppron{His}{Her}}
\newcommand*\HimHer{\gm@atppron{Him}{Her}}
\newcommand*\HisHers{\gm@atppron{His}{Hers}}
```

To Save Precious Count Registers

It's a contribution to TEX's ecology ;-). You can use as many CSs as you wish and you may use only 256 count registers (although in ε-TEX there are 2^{16} count registers, which makes the following a bit obsolete).

\nummacro

```
\newcommand*\nummacro[1]{\gdef#1{0}}
```

\stepnummacro

```
\newcommand*\stepnummacro[1]{%
  \@tempcnta=#1\relax
  \advance\@tempcnta␣by1\relax
  \xdef#1{\the\@tempcnta}}% Because of some mysterious reasons explicit \count0
          interferred with page numbering when used in \gmd@evpaddonce in gm-
          doc.
```

\addtonummacro

```
\newcommand*\addtonummacro[2]{%
  \count0=#1\relax
  \advance\count0by#2\relax
  \xdef#1{\the\count\z@}}
```

Need an explanation? The \nummacro declaration defines its argument (that should be a CS) as {0} which is analogous to \newcount declaration but doesn't use up any count register.

Then you may use this numeric macro as something between TEX's count CS and LATEX's counter. The macros \stepnummacro and \addtonummacro are analogous to LATEX's \stepcounter and \addtocounter respectively: \stepnummacro advances the number stored in its argument by 1 and \addtonummacro advances it by the second argument. As the LATEX's analogoi, they have the global effect (the effect of global warming ;-)).

So far I've used only \nummacro and \stepnummacro. Notify me if you use them and whether you need sth. more, \multiplynummacro e.g.

Improvements to mwcls Sectioning Commands

That is, 'Expe-ri-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.

```
\AtBeginDocument{% because we don't know when exactly hyperref is loaded and maybe after this package.
  \@ifpackageloaded{hyperref}{\newcounter{NoNumSecs}%
    \setcounter{NoNumSecs}{617}% to make \refing to an unnumbered section visible (and funny?).
    \def\gm@hyperrefstepcounter{\refstepcounter{NoNumSecs}}%
    \DeclareRobustCommand*\gm@targetheading[1]{%
      \hypertarget{#1}{#1}}}% end of then
  {\def\gm@hyperrefstepcounter{}%
    \def\gm@targetheading#1{#1}}% end of else
}% of \AtBeginDocument
```

Auxiliary macros for the kernel sectioning macro:

```
\def\gm@dontnumbersectionsoutofmainmatter{%
  \if@mainmatter\else␣\HeadingNumberedfalse␣\fi}
\def\gm@clearpagesduetoopenright{%
  \if@openright\cleardoublepage\else␣\clearpage\fi}
```

To avoid \defing of \mw@sectionxx if it's undefined, we redefine \def to gobble the definition and restore the original meaning of itself.

Why shouldn't we change the ontological status of \mw@sectionxx (not define if undefined)? Because some macros (in gmdocc e.g.) check it to learn whether they are in an mwcls or not.

But let's make a shorthand for this test since we'll use it three times in this package and maybe also somewhere else.

```
\long\def\@ifnotmw#1#2{\@ifundefined{mw@sectionxx}{#1}{#2}}
```

```
\let\gmu@def\def
\@ifnotmw{%
  \StoreMacro\gmu@def␣\def\gmu@def#14#2{\RestoreMacro\gmu@def}}{}
```

I know it may be of bad taste (to write such a way *here*) but I feel so lonely and am in an alien state of mind after 3 hour sleep last night and, worst of all, listening to sir Edward Elgar's flamboyant Symphonies d'Art Nouveau.

A *decent* person would just wrap the following definition in \@ifundefined's Else. But look, the definition is so long and I feel so lonely etc. So, I define \def (for some people there's nothing sacred) to be a macro with two parameters, first of which is delimited by digit 4 (the last token of \mw@sectionxx's parameter string) and the latter is undelimited which means it'll be the body of the definition. Such defined \def does nothing else but restores its primitive meaning by the way sending its arguments to the Gobbled Tokens' Paradise. Luckily, \RestoreMacro contains \let not \def.

The kernel of MW's sectioning commands:

[3] A. Berg, *Wozzeck*.

```
\gmu@def\mw@sectionxx#1#2[#3]#4{%
  \edef\mw@HeadingLevel{\csname␣#1@level\endcsname
        \space}% space delimits level number!
  \ifHeadingNumbered
      \ifnum␣\mw@HeadingLevel>\c@secnumdepth␣%
           \HeadingNumberedfalse␣\fi
```

line below is in ifundefined to make it work in classes other than mwbk

```
      \@ifundefined{if@mainmatter}{}{%
           \gm@dontnumbersectionsoutofmainmatter}
  \fi
```

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

\HeadingRHeadText
\HeadingTOCText
\HeadingText
\mw@HeadingType

```
  \def\HeadingRHeadText{#2}%
  \def\HeadingTOCText{#3}%
  \def\HeadingText{#4}%
  \def\mw@HeadingType{#1}%
  \if\mw@HeadingBreakBefore
    \if@specialpage\else\thispagestyle{closing}\fi
    \@ifundefined{if@openright}{}{\gm@clearpagesduetoopenright}%
    \if\mw@HeadingBreakAfter
      \thispagestyle{blank}\else
      \thispagestyle{opening}\fi
       \global\@topnum\z@
  \fi% of \if\mw@HeadingBreakBefore
```

placement of \refstep suggested by me (GM)

```
  \ifHeadingNumbered
    \refstepcounter{#1}%
    \protected@edef\HeadingNumber{\csname␣the#1\endcsname\relax}%
  \else
    \let\HeadingNumber\@empty
    \gm@hyperrefstepcounter
  \fi% of \ifHeadingNumbered

  \if\mw@HeadingRunIn
    \mw@runinheading
  \else
    \if\mw@HeadingWholeWidth
      \if@twocolumn
        \if\mw@HeadingBreakAfter
        \onecolumn
        \mw@normalheading
        \pagebreak\relax
              \if@twoside
                \null
                \thispagestyle{blank}%
                \newpage
```

```
            \fi% of \if@twoside
        \twocolumn
        \else
          \@topnewpage[\mw@normalheading]%
        \fi% of \if\mw@HeadingBreakAfter
      \else
        \mw@normalheading
        \if\mw@HeadingBreakAfter\pagebreak\relax\fi
      \fi% of \if@twocolumn
    \else
      \mw@normalheading
      \if\mw@HeadingBreakAfter\pagebreak\relax\fi
    \fi% of \if\mw@HeadingWholeWidth
  \fi% of \if\mw@HeadingRunIn
  }
```

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 (optional) the flags, e.g. breakbefore, breakafter;
#2 the sectioning name, e.g. chapter, part;
#3 preskip;
#4 heading type;
#5 postskip

\SetSectionFormatting
\mw@HeadingRunIn
\mw@HeadingBreakBefore
\mw@HeadingBreakAfter
\mw@HeadingWholeWidth

```
\relaxen\SetSectionFormatting
\newcommand*\SetSectionFormatting[5][\empty]{%
  \ifx\empty#1\relax\else% empty (not \empty!) #1 also launches \else.
    \def\mw@HeadingRunIn{10}\def\mw@HeadingBreakBefore{10}%
    \def\mw@HeadingBreakAfter{10}\def\mw@HeadingWholeWidth{10}%
    \@ifempty{#1}{}{\mw@processflags#1,\relax}% If #1 is omitted, the flags
        are left unchanged. If #1 is given, even as [], the flags are first cleared and
        then processed again.
  \fi
  \@ifundefined{#2}{\@namedef{#2}{\mw@section{#2}}}{}%
  \mw@secdef{#2}{@preskip}␣{#3}{2␣oblig.}%
  \mw@secdef{#2}{@head}␣␣␣␣{#4}{3␣oblig.}%
  \mw@secdef{#2}{@postskip}{#5}{4␣oblig.}%
  \ifx\empty#1\relax
    \mw@secundef{#2@flags}{1␣(optional)}%
  \else\mw@setflags{#2}%
  \fi}
```

\mw@secdef

```
\def\mw@secdef#1#2#3#4{% #1 the heading name,
         % #2 the command distinctor,
         % #3 the meaning,
         % #4 the number of argument to error message.
  \@ifempty{#3}
    {\mw@secundef{#1#2}{#4}}
    {\@namedef{#1#2}{#3}}}
```

\mw@secundef

```
\def\mw@secundef#1#2{%
1497  \@ifundefined{#1}{%
1498    \ClassError{mwcls/gm}{%
1499      command␣\bslash#1␣␣undefined␣\MessageBreak
1500      after␣\bslash␣SetSectionFormatting!!!\MessageBreak}{%
1501      Provide␣the␣#2␣argument␣of␣\bslash␣
              SetSectionFormatting.}}{}}
```

First argument is a sectioning command (wo. \) and second the stuff to be added at the beginning of the heading declarations.

\addtoheading

```
1506 \def\addtoheading#1#2{%
1507     \n@melet{gmu@reserveda}{#1@head}%
1508     \toks\z@=\@xa{\gmu@reserveda}%
1509     \toks\tw@={#2}%
1510     \edef\gmu@reserveda{\the\toks\tw@\the\toks\z@}%
1511     \n@melet{#1@head}{gmu@reserveda}%
1513 }
```

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.

```
1525 \@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

```
1530 \def\@afterheading{%
1531   \@nobreaktrue
1532   \xdef\gmu@prevsec{\mw@HeadingType}% added now
1533   \everypar{%
1534     \grelaxen\gmu@prevsec% added now. All the rest is original LaTeX.
1535     \if@nobreak
1536     \@nobreakfalse
1537     \clubpenalty␣\@M
1538     \if@afterindent␣\else
1539     {\setbox\z@\lastbox}%
1540     \fi
1541     \else
1542     \clubpenalty␣\@clubpenalty
1543     \everypar{}%
1544     \fi}}
```

If we are (with the current heading) just after another heading (one level lower I suppose), then we add the less of the higher header's post-skip and the lower header pre-skip or, if defined, the two-header-skip. (We put the macro defined below just before \addvspace in MWCLS inner macros.)

\gmu@checkaftersec

```
1551 \def\gmu@checkaftersec{%
1552   \@ifundefined{gmu@prevsec}{}{%
1553     \ifgmu@postsec% an additional switch that is true by default but may be
              turned into an \ifdim in special cases, see line 1589.
```

```
                      {\@xa\mw@getflags\@xa{\gmu@prevsec}%
                        \glet\gmu@reserveda\mw@HeadingBreakAfter}%
\gmu@reserveda        \if\mw@HeadingBreakBefore\def\gmu@reserveda{11}\fi% if the current
                            heading inserts page break before itself, all the play with vskips is irrele-
                            vant.
                      \if\gmu@reserveda\else
                      \penalty10000\relax
                      \skip\z@=\csname\gmu@prevsec␣@postskip\endcsname\relax
                      \skip\tw@=\csname\mw@HeadingType␣@preskip\endcsname\relax
                      \@ifundefined{\mw@HeadingType␣@twoheadskip}{
                        \ifdim\skip\z@>\skip\tw@
                        \vskip-\skip\z@% we strip off the post-skip of previous header if it's bigger
                              than current pre-skip
                        \else
                        \vskip-\skip\tw@% we strip off the current pre-skip otherwise
                        \fi}{% But if the two-header-skip is defined, we put it
                        \penalty10000
                        \vskip-\skip\z@
                        \penalty10000
                        \vskip-\skip\tw@
                        \penalty10000
                        \vskip\csname\mw@HeadingType␣@twoheadskip\endcsname
                        \relax}%
                      \penalty10000
                      \hrule␣height\z@\relax% to hide the last (un)skip before subsequent \addvspaces.
                      \penalty10000
                      \fi
                      \fi
                    }% of \@ifundefined{gmu@prevsec} 'else'
                  }% of \def\gmu@checkaftersec

\ParanoidPostsec  \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      \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).
                    }}

                  \let\ifgmu@postsec\iftrue

\gmu@getaddvs     \def\gmu@getaddvs#1\addvspace#2\gmu@getaddvs{%
                    \toks\z@={#1}
                    \toks\tw@={#2}}
```

And the modification of the inner macros at last:

```
\gmu@setheading   \def\gmu@setheading#1{%
                    \@xa\gmu@getaddvs#1\gmu@getaddvs
                    \edef#1{%
                      \the\toks\z@\@nx\gmu@checkaftersec
                      \@nx\addvspace\the\toks\tw@}}

                  \gmu@setheading\mw@normalheading
                  \gmu@setheading\mw@runinheading
```

\SetTwoheadSkip

```
\def\SetTwoheadSkip#1#2{\@namedef{#1@twoheadskip}{#2}}
}% of \@ifnotmw
```

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.

\WPheadings

```
\@ifnotmw{}{% We define this declaration only when in mwcls.
\def\WPheadings{%
  \SetSectionFormatting[breakbefore,wholewidth]
      {part}{\z@\@plus1fill}{}{\z@\@plus3fill}%
  \@ifundefined{chapter}{}{%
    \SetSectionFormatting[breakbefore,wholewidth]
      {chapter}
      {66\p@}% {67\p@} for Adventor/Schola 0,95.
      {\FormatHangHeading{\LARGE}}
      {27\p@\@plus0,2\p@\@minus1\p@}%
  }%
  \SetTwoheadSkip{section}{27\p@\@plus0,5\p@}%
  \SetSectionFormatting{section}
      {24\p@\@plus0,5\p@\@minus5\p@}%
      {\FormatHangHeading␣{\Large}}
      {10\p@\@plus0,5\p@}% ed. Krajewska of "Wiedza Powszechna", as we understand her, wants the skip between a heading and text to be rigid.
  \SetTwoheadSkip{subsection}{11\p@\@plus0,5\p@\@minus1\p@}%
  \SetSectionFormatting{subsection}
      {19\p@\@plus0,4\p@\@minus6\p@}
      {\FormatHangHeading␣{\large}}% 12/14 pt
      {6\p@\@plus0,3\p@}% after-skip 6 pt due to p.12, not to squeeze the before-skip too much.
  \SetTwoheadSkip{subsubsection}{10\p@\@plus1,75\p@\@minus1\p@}%
  \SetSectionFormatting{subsubsection}
      {10\p@\@plus0,2\p@\@minus1\p@}
      {\FormatHangHeading␣{\normalsize}}
      {3\p@\@plus0,1\p@}% those little skips should be smaller than you calculate out of a geometric progression, because the interline skip enlarges them.
  \SetSectionFormatting[runin]{paragraph}
      {7\p@\@plus0,15\p@\@minus1\p@}
      {\FormatRunInHeading{\normalsize}}
      {2\p@}%
  \SetSectionFormatting[runin]{subparagraph}
      {4\p@\@plus1\p@\@minus0,5\p@}
      {\FormatRunInHeading{\normalsize}}
      {\z@}%
}% of \WPheadings
}% 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.)

```
              \@ifnotmw{% we are not in mwcls and want to handle mwcls-like sectionings i.e.,
                    those written with two optionals.
\gm@secini      \def\gm@secini{gm@la}%
\gm@secxx       \def\gm@secxx#1#2[#3]#4{%
                  \ifx\gm@secstar\@empty
                    \n@melet{gm@true@#1mark}{#1mark}% a little trick to allow a special version of the heading just to the running head.
                    \@namedef{#1mark}##1{% we redefine \⟨sec⟩mark to gobble its argument and to launch the stored true marking command on the appropriate argument.
                      \csname␣gm@true@#1mark\endcsname{#2}%
                      \n@melet{#1mark}{gm@true@#1mark}% after we've done what we wanted we restore original \#1mark.
                    }%
\gm@secstar         \def\gm@secstar{[#3]}% if \gm@secstar is empty, which means the sectioning command was written starless, we pass the 'true' sectioning command #3 as the optional argument. Otherwise the sectioning command was written with star so the 'true' s.c. takes no optional.
                  \fi
                  \@xa\@xa\csname\gm@secini#1\endcsname
                  \gm@secstar{#4}}%

              }{% we are in mwcls and want to reverse MW's optionals order i.e., if there's just one optional, it should go both to toc and to running head.
\gm@secini      \def\gm@secini{gm@mw}%
                \let\gm@secmarkh\@gobble% in mwcls there's no need to make tricks for special version to running headings.
\gm@secxx       \def\gm@secxx#1#2[#3]#4{%
                  \@xa\@xa\csname\gm@secini#1\endcsname
```

```
                        \gm@secstar[#2][#3]{#4}}%
                    }
\gm@sec             \def\gm@sec#1{\@dblarg{\gm@secx{#1}}}
\gm@secx            \def\gm@secx#1[#2]{%
                      \@ifnextchar[{\gm@secxx{#1}{#2}}{\gm@secxx{#1}{#2}[#2]}}% if there's
                            only one optional, we double it not the mandatory argument.
\gm@straightensec   \def\gm@straightensec#1{% the parameter is for the command's name.
                      \@ifundefined{#1}{}{% we don't change the ontological status of the command
                            because someone may test it.
                        \n@melet{\gm@secini#1}{#1}%
                        \@namedef{#1}{%
\gm@secstar               \@ifstar{\def\gm@secstar{*}\gm@sec{#1}}{%
\gm@secstar                 \def\gm@secstar{}\gm@sec{#1}}}}%
                    }%
                    \let\do\gm@straightensec
                    \do{part}\do{chapter}\do{section}\do{subsection}\do{%
                        subsubsection}
                    \@ifnotmw{}{\do{paragraph}}% this 'straightening' of \paragraph with the stan-
                        dard article caused the 'TEX capacity exceeded' error. Anyway, who on Earth
                        wants paragraph titles in toc or running head?
```

enumerate* and itemize*

We wish the starred version of `enumerate` to be just numbered paragraphs. But hyperref redefines `\item` so we should do it a smart way, to set the LaTeX'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*   \@namedef{enumerate*}{%
               \ifnum\@enumdepth>\thr@@
                 \@toodeep
               \else
                 \advance\@enumdepth\@ne
                 \edef\@enumctr{enum\romannumeral\the\@enumdepth}%
                 \@xa\list\csname␣label\@enumctr\endcsname{%
                   \partopsep\topsep␣\topsep\z@␣\leftmargin\z@
                   \itemindent\@parindent␣% %\advance\itemindent\labelsep
                   \labelwidth\@parindent
                   \advance\labelwidth-\labelsep
                   \listparindent\@parindent
                   \usecounter␣\@enumctr
                   \def\makelabel##1{##1\hfil}}%
               \fi}
             \@namedef{endenumerate*}{\endlist}
itemize*     \@namedef{itemize*}{%
               \ifnum\@itemdepth>\thr@@
                 \@toodeep
               \else
                 \advance\@itemdepth\@ne
```

```
    \edef\@itemitem{labelitem\romannumeral\the\@itemdepth}%
    \@xa\list\csname\@itemitem\endcsname{%
      \partopsep\topsep␣\topsep\z@␣\leftmargin\z@
      \itemindent\@parindent
      \labelwidth\@parindent
      \advance\labelwidth-\labelsep
      \listparindent\@parindent
      \def\makelabel##1{##1\hfil␣}}%
  \fi}
\@namedef{enditemize*}{\endlist}
```

The Logos

We'll modify The LaTeX logo now to make it fit better to various fonts.

```
\let\oldLaTeX\LaTeX
\let\oldLaTeXe\LaTeXe
```

```
\def\TeX{T\kern-.1667em\lower.5ex\hbox{E}\kern-.125emX\@}
```

```
\DeclareLogo           \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         \ifx\relax#1\def\gmu@reserveda{\@xa\@gobble\string#2}%
                       \else
\gmu@reserveda           \def\gmu@reserveda{#1}%
                       \fi
                       \edef\gmu@reserveda{%
\@nx                     \@nx\DeclareTextCommand\@nx#2{PD1}{\gmu@reserveda}}
                       \gmu@reserveda
                       \DeclareTextCommandDefault#2{#3}%
\DeclareRobustCommand* \DeclareRobustCommand*#2{#3}}% added for XeTeX
\DeclareLogo         \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

```
\DeclareLogo\LaTeXe{\mbox{\m@th␣\if
    b\expandafter\@car\f@series\@nil\boldmath\fi
    \LaTeX\kern.15em2$_{\textstyle\varepsilon}$}}
```

```
\StoreMacro\LaTeX
\StoreMacro*{LaTeX␣}
```

'(LA)TEX' in my opinion better describes what I work with/in than just 'LATEX'.

\LaTeXpar

```
\DeclareLogo[(La)TeX]{\LaTeXpar}{%
  {%
    \setbox\z@\hbox{(}%)
    \copy\z@
    \kern-.2\wd\z@␣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@{\box\z@%
      \vss}%
  }%
  \kern-.07em% originally −, 15 em for T.
  {%(
    \sbox\z@)%
    \kern-.2\wd\z@\copy\z@
    \kern-.2\wd\z@}\TeX
}
```

"Here are a few definitions which can usefully be employed when documenting package files: now we can readily refer to AMS-TEX, BIBTEX and SLITEX, as well as the usual TEX and LATEX. There's even a PLAIN TEX and a WEB."

\AmSTeX

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

\BibTeX

```
\DeclareLogo\BibTeX{{\rmfamily␣B\kern-.05em%
    \textsc{i{\kern-.025em}b}\kern-.08em% the kern is wrapped in braces
          for my \fakescaps' sake.
    \TeX}}
```

\SliTeX

```
\DeclareLogo\SliTeX{{\rmfamily␣S\kern-.06emL\kern-.18em%
      \raise.32ex\hbox
         {\scshape␣i}\kern␣-.03em\TeX}}
```

\PlainTeX

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

\Web

```
\DeclareLogo\Web{\textsc{Web}}
```

There's also the (LA)TEX logo got with the \LaTeXpar macro provided by gmutils. And here *The TEXbook*'s logo:

```
\TeXbook      \DeclareLogo[The␣TeX␣book]\TeXbook{\textsl{The␣\TeX␣book}}
              \let\TB\TeXbook% TUG Boat uses this.

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

\pdfeTeX      \DeclareLogo[pdfe-TeX]\pdfeTeX{pdf\eTeX}

\pdfTeX       \DeclareLogo\pdfTeX{pdf\TeX}

              \@ifundefined{XeTeX}{%
\XeTeX          \DeclareLogo\XeTeX{X\kern-.125em\relax
                  \@ifundefined{reflectbox}{%
                    \lower.5ex\hbox{E}\kern-.1667em\relax}{%
                    \lower.5ex\hbox{\reflectbox{E}}\kern-.1667em\relax}%
                  \TeX}}{}

              \@ifundefined{XeLaTeX}{%
\XeLaTeX        \DeclareLogo\XeLaTeX{X\kern-.125em\relax
                  \@ifundefined{reflectbox}{%
                    \lower.5ex\hbox{E}\kern-.1667em\relax}{%
                    \lower.5ex\hbox{\reflectbox{E}}\kern-.1667em\relax}%
                  \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-XƎTeX usage. With XƎTeX, you can load the xltxtra package (e.g. with the gmutils `\XeTeXthree` declaration) and then the reversed E you get as the Unicode Latin Letter Reversed E.

Expanding 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 CSs: they could'nt be used within `\csname...\endcsname`. My TeXGuru advised to use `\meanig` to make all the name 'other'. So—here we are.

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

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

```
\all@other    \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:

```
              \edef\gmu@reserveda{%
\@nx            \def\@nx\gm@gobmacro##1\@xa\@gobble\string\macro:->{}}
\gm@gobmacro  \gmu@reserveda
```

In the next two macros' names, 'unex' stands both for not expanding the argument(s) and for disastrously partial unexpandability of the macros themselves.

```
\unex@namedef \long\def\unex@namedef#1#2{%
                \edef@other\gmu@reserveda{#1}%
                \@xa\long\@xa\def\csname\gmu@reserveda\endcsname{#2}}

\unex@nameuse \long\def\unex@nameuse#1{%
                \edef@other\gmu@reserveda{#1}%
                \csname\gmu@reserveda\endcsname}
```

Brave New World of XƎTEX

```
\@ifXeTeX        2011 \newcommand\@ifXeTeX[2]{%
                 2012   \ifdefined\XeTeXversion
                 2013   \unless\ifx\XeTeXversion\relax\afterfifi{#1}\else\afterfifi{%
                              #2}\fi
                 2014   \else\afterfi{#2}\fi}
\XeTeXthree      2017 \def\XeTeXthree{%
                 2019   \@ifXeTeX{%
                 2025     \@ifpackageloaded{gmverb}{\StoreMacro\verb}{}%
                 2026     \RequirePackage{xltxtra}% 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.
                 2033     \@ifpackageloaded{gmverb}{\RestoreMacro\verb}{}%
                 2034     \AtBeginDocument{%
                 2035       \RestoreMacro\LaTeX\RestoreMacro*{LaTeX␣}}% my version of the LaTeX logo has been stored just after defining, in line 1882.
                 2039 }{}}
```

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

```
                 2045 \AtBeginDocument{%
                 2046   \@ifpackageloaded{fontspec}{%
\udigits         2047     \DeclareRobustCommand*\udigits{%
                 2048       \addfontfeature{Numbers=Uppercase}}%
                 2049   }{%
                 2050     \emptify\udigits}}
```

Fractions

```
\Xedekfracc      2055 \def\Xedekfracc{\@ifstar\gmu@xedekfraccstar\gmu@xedekfraccplain}
```

(plain) The starless version turns the font feature frac on. (*) But nor Minion GM 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 2069 \def\gmu@xedekfraccstar{%
\gmu@xefraccdef     2070   \def\gmu@xefraccdef##1##2{%
                    2071       \iffontchar\font␣##2
                    2072         \@namedef{gmu@xefracc##1}{\char##2␣}%
                    2073       \else
                    2074         \n@melet{gmu@xefracc##1}{relax}%
                    2075       \fi}%
\gmu@dekfracc       2077     \def\gmu@dekfracc##1/##2{%
                    2078       {\addfontfeature{VerticalPosition=Numerator}##1}%
                                 \gmu@numeratorkern
                    2079       \char"2044␣\gmu@denominatorkern
                    2080       {\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.

```
                          \gmu@xefraccdef{1/4}{"BC}%
                          \gmu@xefraccdef{1/2}{"BD}%
                          \gmu@xefraccdef{3/4}{"BE}%
                          \gmu@xefraccdef{1/3}{"2153}%
                          \gmu@xefraccdef{2/3}{"2154}%
                          \gmu@xefraccdef{1/8}{"215B}%
                          \gmu@xefraccdef{3/8}{"215C}%
                          \gmu@xefraccdef{5/8}{"215D}%
                          \gmu@xefraccdef{7/8}{"215E}%
\dekfracc                 \def\dekfracc##1/##2{%
\gm@duppa                   \def\gm@duppa{##1/##2}%
                            \@ifundefined{gmu@xefracc\all@other\gm@duppa}{%
                              \gmu@dekfracc{##1}/{##2}}{%
                              \csname␣gmu@xefracc\all@other\gm@duppa\endcsname}}%
                          \@ifstar{\let\gmu@dekfracc\gmu@dekfraccsimple}{}%
                        }

\gmu@xedekfraccplain    \def\gmu@xedekfraccplain{% 'else' of the main \@ifstar
\dekfracc                 \def\dekfracc##1/##2{{%
                              \addfontfeature{Fractions=On}%
                              ##1/##2}}%
                          }

\gmu@numeratorkern      \def\gmu@numeratorkern{\kern-.05em\relax}
                        \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 ½ or ⅞. 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 LaTeX 2$_\varepsilon$ 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 2491

\resizegraphics

```
                        \@ifXeTeX{%
\resizegraphics           \def\resizegraphics#1#2#3{{%
                              \setbox0=\hbox{\XeTeXpicfile␣#3␣}%
                              \ifx!#1\else
                                \dimen0=#1\relax
                                \count2=\wd0
                                \divide\count2␣by1000\relax
                                \count0=\dimen0\relax
                                \divide\count0\count2
                              \fi
                              \ifx!#2\else
                                \dimen0=#1\relax
                                \count6=\ht0
```

```
              \divide\count6␣by1000\relax
              \count4=\dimen0\relax
              \divide\count4\count6
            \fi
            \ifx!#1\count0=\count4\fi
            \ifx!#2\count4=\count0\fi
            \XeTeXpicfile␣#3␣xscaled␣\count0␣yscaled␣\count4
          }}}{%
\resizegraphics       \def\resizegraphics#1#2#3{%
            \resizebox{#1}{#2}{%
              \includegraphics{#3}}}}%
```

The [options] in the `\XeTeXpicfile` command use the following keywords:
`width` ⟨*dimen*⟩
`height` ⟨*dimen*⟩
`scaled` ⟨*scalefactor*⟩
`xscaled` ⟨*scalefactor*⟩
`yscaled` ⟨*scalefactor*⟩
`rotated` ⟨*degrees*⟩

```
\GMtextsuperscript    \def\GMtextsuperscript{%
                        \@ifXeTeX{%
\textsuperscript          \def\textsuperscript##1{{%
                              \addfontfeature{VerticalPosition=Numerator}##1}}%
                        }{\truetextsuperscript}}
\truetextsuperscript  \def\truetextsuperscript{%
\textsuperscript        \DeclareRobustCommand*\textsuperscript[1]{%
                          \@textsuperscript{\selectfont##1}}%
\@textsuperscript       \def\@textsuperscript##1{%
                          {\m@th\ensuremath{^{\mbox{\fontsize\sf@size\z@##1}}}}}}
```

Varia

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

```
\gmu@tilde            \def\gmu@tilde{%
                        \leavevmode\lower.8ex\hbox{$\,\widetilde{\mbox{␣}}\,$}}
```

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

```
\*                    \DeclareRobustCommand*\*{\gmu@tilde}

                      \AtBeginDocument{% to bypass redefinition of \~ as a text command with various
                              encodings
\texttilde              \DeclareRobustCommand*\texttilde{%
                          \@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 `\active`ate the space but also will (re)define it as the `\␣` primitive. So define `\gmobeyspaces` that obeys this requirement.

(This definition is repeated in gmverb.)

```
\foone{\catcode`\ \active}%
```

```
{\def\gmobeyspaces{\let␣\ \catcode`\ \active}}
```

While typesetting poetry, I was surprised that sth. didn't work. The reason was that original `\obeylines` does `\let` not `\def`, so I give the latter possibility.

```
\foone{\catcode`\^^M\active}% the comment signs here are crucial.
{\def\defobeylines{\catcode`\^^M=13␣\def^^M{\par}}}
```

Another thing I dislike in LaTeX yet is doing special things for `\...skip`'s, 'cause I like the Knuthian simplicity. So I sort of restore Knuthian meanings:

```
\def\deksmallskip{\vskip\smallskipamount}
\def\undeksmallskip{\vskip-\smallskipamount}
\def\dekmedskip{\vskip\medskipamount}
\def\dekbigskip{\vskip\bigskipamount}
```

```
\def\hfillneg{\hskip␣0pt␣plus␣-1fill\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.

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

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

```
\newcommand*{\TODO}[1][]{{%
    \sffamily\bfseries\huge␣TODO!\if\relax#1\relax\else\space%
        \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 LaTeX macro instead.

```
\newcommand*\twocoltoc{%
  \RequirePackage{multicol}%
  \def\@starttoc##1{%
    \begin{multicols}{2}\makeatletter\@input␣{\jobname␣.##1}%
      \if@filesw␣\@xa␣\newwrite␣\csname␣tf@##1\endcsname
        \immediate␣\openout␣\csname␣tf@##1\endcsname␣\jobname␣
            .##1\relax
      \fi
      \@nobreakfalse\end{multicols}}}
```

```
\@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.

```
\newcommand\enoughpage[1]{%
  \par
  \dimen0=\pagegoal
  \advance\dimen0␣by-\pagetotal
  \ifdim\dimen0<#1\relax\newpage\fi}
```

Two shorthands for debugging:

```
\newcommand*\tOnLine{\typeout{\on@line}}
```

```
\let\OnAtLine\on@line
```

An equality sign properly spaced:

```
\newcommand*\equals{${}={}$}
```

And for the LaTeX's pseudo-code statements:

```
\eequals        \newcommand*\eequals{${}=={}$}
```

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 \writen 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 \def\freeze@actives{%
                  \count\z@\z@
                  \@whilenum\count\z@<\@cclvi\do{%
                    \ifnum\catcode\count\z@=\active
             \~       \uccode`\~=\count\z@
                      \uppercase{\let~\relax}%
                    \fi
                    \advance\count\z@\@ne}}
```

A macro that typesets all 256 chars of given font. It makes use of \@whilenum.

```
\ShowFont       \newcommand*\ShowFont[1][6]{%
                  \begin{multicols}{#1}[The␣current␣font␣(the␣\f@encoding%
                         \ encoding):]
                    \parindent\z@
                    \count\z@\m@ne
                    \@whilenum\count\z@<\@cclv\do{
                      \advance\count\z@\@ne
                      \ \the\count\z@:~\char\count\z@\par}
                  \end{multicols}}
```

A couple of macros for typesetting liturgic texts such as psalmody of Liturgia Horarum. I wrap them into a declaration since they'll be needed not every time.

```
\liturgiques    \newcommand*\liturgiques[1][red]{% Requires the color package.
                  \gmu@RPif{color}{color}%
\czerwo           \newcommand*\czerwo{\small\color{#1}}% environment
\czer             \newcommand{\czer}[1]{\leavevmode{\czerwo##1}}% we leave vmode be-
                         cause if we don't, then verse's \everypar would be executed in a group
                         and thus its effect lost.
\*                \def\*{\czer{$*$}}
\+                \def\+{\czer{$\dag$}}
\nieczer          \newcommand*\nieczer[1]{\textcolor{black}{##1}}}
```

After the next definition you can write \gmu@RP[⟨*options*⟩]{⟨*package*⟩}{⟨*csname*⟩} to get the package #2 loaded with options #1 if the csname #3 is undefined.

```
\gmu@RPif       \newcommand*\gmu@RPif[3][]{%
                  \ifx\relax#1\relax
\gmu@resa         \else␣\def\gmu@resa{[#1]}%
                  \fi
                  \@xa\RequirePackage\gmu@resa{#2}}
```

Since inside document we cannot load a package, we'll redefine \gmu@RPif to issue a request before the error issued by undefined CS.

```
\AtBeginDocument{%
  \renewcommand*\gmu@RPif[3][]{%
    \@ifundefined{#3}{%
      \@ifpackageloaded{#2}{}{%
        \typeout{^^J!␣Package␣`#2'␣not␣loaded!!!␣(%
             \on@line)^^J}}}{}}}
```

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.

```
\providecommand*\continuum{\gmu@RPif{eufrak}{mathfrak}\mathfrak{%
      c}}
```

And this macro I saw in the ltugproc document class nad I liked it.

```
\def\iteracro{%
  \DeclareRobustCommand*\acro[1]{\gmu@acrospaces##1␣%
       \gmu@acrospaces}%
}

\iteracro

\def\gmu@acrospaces#1␣#2\gmu@acrospaces{%
  \gmu@acroinner#1\gmu@acroinner
  \ifx\relax#2\relax\else
    \space
    \afterfi{\gmu@acrospaces#2\gmu@acrospaces}% when #2 is nonempty, it
           is ended with a space. Adding one more space in this line resulted in an
           infinite loop.
  \fi}

\def\gmu@acroinner#1{%
  \ifx\gmu@acroinner#1\relax\else
    \ifcat␣a\@nx#1\relax%
      \ifnum`#1=\uccode`#1%
        {\acrocore{#1}}%
      \else{#1}% tu było \smallerr
      \fi
    \else#1%
    \fi
    \afterfi\gmu@acroinner
  \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.

```
\def\acrocore{\scshape\lowercase}
```

Since the fonts I am currently using do not support required font feature, I skip the following definition.

```
\newcommand*\IMO{\acro{IMO}}
\newcommand*\AKA{\acro{AKA}}

\DeclareRobustCommand*\usc[1]{{\addfontfeature{%
      Letters=UppercaseSmallCaps}#1}}

\def\uscacro{\let\acro\usc}

\newcommand*\qxenc{\fontencoding{QX}\selectfont}
```

The \copyright command is unavailable in T1 and U (unknown) encodings so provide

\qxcopyright \qxcopyrights

```
\newcommand*\qxcopyright{{\qxenc\copyright}}
\newcommand*\qxcopyrights{%
  \let\gmu@copyright\copyright
  \def\copyright{{\qxenc\gmu@copyright}}}
```

\fixcopyright

```
\newcommand*\fixcopyright{%
  \@ifXeTeX{\def\copyright{\char"00A9␣}}{\qxcopyrights}}
```

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 \ifSecondClass

```
\def\secondclass{%
  \newif\ifSecondClass
  \SecondClasstrue
  \@fileswithoptions\@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 LATEX:

```
% \check@mathfonts\fontsize\sf@size\z@\math@fontsfalse\selectfont
```

didn't work as I would wish: in a `\footnotesize`'s scope it still was `\scriptsize`, so too large.

\gmu@dekfraccsimple

```
\def\gmu@dekfraccsimple#1/#2{\leavevmode\kern.1em
  \raise.5ex\hbox{\udigits\smaller[3]#1}\gmu@numeratorkern
  \dekfraccslash\gmu@denominatorkern
  {\udigits\smaller[3]#2}}%
```

\dekfraccsimple \dekfraccslash \dekfraccslash

```
\def\dekfraccsimple{%
  \let\dekfracc\gmu@dekfraccsimple
}
\@ifXeTeX{\def\dekfraccslash{\char"2044␣}}{%
  \def\dekfraccslash{/}}␣% You can define it as the fraction slash, \char"2044

\dekfraccsimple
```

A macro that acts like `\,` (thin and unbreakable space) except it allows hyphenation afterwards:

\ikern

```
\newcommand*\ikern{\,\penalty10000\hskiposp\relax}
```

And a macro to forbid hyphenation of the next word:

\nohy \yeshy

```
\newcommand*\nohy{\leavevmode\kernosp\relax}
\newcommand*\yeshy{\leavevmode\penalty10000\hskiposp\relax}
```

In both of the above definitionc 'osp' not `\z@` to allow their writing to and reading from files where @ is 'other'.

`\@ifempty`

\@ifempty \gmu@reserveda

```
\long\def\@ifempty#1#2#3{%
  \def\gmu@reserveda{#1}%
  \ifx\gmu@reserveda\@empty\afterfi{#2}%
  \else\afterfi{#3}\fi
}
```

\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@getext
\gmu@filename
\gmu@fileext

```
\def\gmu@getext#1.#2\@@nil{%
  \def\gmu@filename{#1}%
  \def\gmu@fileext{#2}}
```

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

\@include
\gmu@fileext

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

```
    \deadcycles\z@
    \@nameuse{cp@\gmu@filename}%
  \fi
  \let\@auxout\@mainaux}
```

\whenonly
\gmu@whonly

```
\newcommand\whenonly[3]{%
  \def\gmu@whonly{#1,}%
  \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.

```
\def\inclasthook{\thispagestyle{closing}}
```

Faked small caps

```
\def\gmu@scapLetters#1{%
  \ifx#1\relax\relax\else% two \relaxes to cover the case of empty #1.
    \ifcat␣a#1\relax
      \ifnum\the\lccode`#1=`#1\relax
        {\fakescapscore\MakeUppercase{#1}}% not Plain \uppercase because that works bad with inputenc.
      \else#1%
      \fi
    \else#1%
    \fi%
    \@xa\gmu@scapLetters
  \fi}%
\def\gmu@scapSpaces#1␣#2\@@nil{%
  \ifx#1\relax\relax
  \else\gmu@scapLetters#1\relax
  \fi
  \ifx#2\relax\relax
  \else\afterfi{\ \gmu@scapSpaces#2\@@nil}%
  \fi}
\def\gmu@scapss#1\@@nil{{\def~{{\nobreakspace}}%
    \gmu@scapSpaces#1␣\@@nil}}% % \def\\{{\newline}}\relax adding redefinition of \\ caused stack overflow Note it disallows hyphenation except at \-.
\DeclareRobustCommand\fakescaps[1]{{%
    \gmu@scapss#1\@@nil}}
```

```
\let\fakescapscore\gmu@scalematchX
```

Experimente z akcentami patrz no3.tex.

```
\def\tinycae{{\tiny\AE}}% to use in \fakescaps[\tiny]{...}
```

```
\RequirePackage{calc}
```

wg \zf@calc@scale pakietu fontspec.

```
\@ifXeTeX{%
  \def\gmu@scalar{1.0}%
  \def\zf@scale{}%
  \def\gmu@scalematchX{%
    \begingroup
      \ifx\zf@scale\empty\def\gmu@scalar{1.0}%
      \else\let\gmu@scalar\zf@scale\fi
      \setlength\@tempdima{\fontdimen5\font}% 5—ex height
      \setlength\@tempdimb{\fontdimen8\font}% 8—XeTeX synthesized uppercase height.
      \divide\@tempdimb␣by1000\relax
      \divide\@tempdima␣by\@tempdimb
      \setlength{\@tempdima}{\@tempdima*\real{\gmu@scalar}}%
      \@ifundefined{fakesc@extrascale}{}{%
        \setlength{\@tempdima}{\@tempdima*\real{%
            \fakesc@extrascale}}}%
      \@tempcnta=\@tempdima
      \divide\@tempcnta␣by␣1000\relax
      \@tempcntb=-1000\relax
      \multiply\@tempcntb␣by\@tempcnta
      \advance\@tempcntb␣by\@tempdima
      \xdef\gmu@scscale{\the\@tempcnta.%
        \ifnum\@tempcntb<100␣0\fi
        \ifnum\@tempcntb<10␣0\fi
        \the\@tempcntb}%
    \endgroup
    \addfontfeature{Scale=\gmu@scscale}%
  }}{\let\gmu@scalematchX\smallerr}
```

\fakescextrascale
\fakesc@extrascale

```
\def\fakescextrascale#1{\def\fakesc@extrascale{#1}}
```

See above/see below

To generate a phrase as in the header depending of whether the respective label is before of after.

\wyzejnizej

```
\newcommand*\wyzejnizej[1]{%
  \edef\gmu@tempa{\@ifundefined{r@#1}{\arabic{page}}{%
      \@xa\@xa\@xa\@secondoftwo\csname␣r@#1\endcsname}}%
  \ifnum\gmu@tempa<\arabic{page}\relax␣wy\.zej\fi
  \ifnum\gmu@tempa>\arabic{page}\relax␣ni\.zej\fi
  \ifnum\gmu@tempa=\arabic{page}\relax␣\@xa\ignorespaces\fi
}
```

lugniej 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

```
\def\napapierkistretch{0,3pt}% It's quite much for 11/13pt typesetting
```

\napapierkicore

```
\def\napapierkicore{\advance\baselineskip%
  by␣0ptplus\napapierkistretch\relax}
```

napapierki

```
\newenvironment*{napapierki}{%
  \par\global\napapierkicore}{%
  \par\dimen\z@=\baselineskip
  \global\baselineskip=\dimen\z@}% so that you can use \endnapapierki in
          interlacing environments
```

\gmu@luzniej

```
\newcount\gmu@luzniej
```

\luzniejcore

```
\newcommand*\luzniejcore[1][1]{%
  \advance\gmu@luzniej\@ne% We use this count to check whether we open the
          environment or just set \looseness inside it again.
  \ifnum\gmu@luzniej=\@ne␣␣\multiply\tolerance␣by␣2␣\fi
  \looseness=#1\relax}
```

After \begin{luzniej} we may put the optional argument of \luzniejcore

luzniej

```
\newenvironment*{luzniej}{\par\luzniejcore}{\par}
```

The starred version does that \everypar, which has its advantages and disadvantages.

luzniej*

```
\newenvironment*{luzniej*}[1][1]{%
  \multiply\tolerance␣by␣2\relax
  \everypar{\looseness=#1\relax}}{\par}
\newcommand*\nawj{\kern0,1em\relax}% to put between parentheses and letters with lower ... 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.

```
\ifdefined\XeTeXversion
\AtBeginDocument{% to be independent of moment of loading of polski.
  \DeclareRobustCommand*\–{%
    \ifhmode
      \unskip\penalty10000
      \afterfi{%
        \@ifnextspace{\hskip0.2em␣plus0.1em\relax
          \pauzacore\hskip.2em␣plus0.1em\relax\ignorespaces}%
        {\pauzacore\penalty\hyphenpenalty\hskip\z@}}%
      \else
```

According to *Instrukcja technologiczna. Skład ręczny i maszynowy* the dialogue dash should be followed by a rigid hskip of ½ em.

```
      \leavevmode\pauzacore\penalty10000\hskip0,5em\ignorespaces
    \fi}%
```

The next command's name consists of letters and therefore it eats any spaces following it, so `\@ifnextspace` would always be false.

```
  \DeclareRobustCommand*\pauza{%
    \ifhmode
      \unskip\penalty10000
      \hskip0.2em␣plus0.1em\relax
      \pauzacore\hskip.2em␣plus0.1em\relax\ignorespaces%
    \else
```

According to *Instrukcja technologiczna. Skład ręczny i maszynowy* the dialogue dash should be followed by a rigid hskip of ½ em.

```
      \leavevmode\pauzacore\penalty10000\hskip0,5em\ignorespaces
    \fi}%
```

And a version with no space at the left, to begin a `\noindent` paragraph or a dialogue in quotation marks:

```
  \DeclareRobustCommand*\lpauza{%
    \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 space$_1$0. Of course you'll never get such a space after a literal CS so an explicit \ppauza will always result with a bare discretionary en dash, but if we `\let-`\ppauza...

```
  \DeclareRobustCommand*\-{%
    \ifvmode␣␣␣␣\PackageError{gmutils}{%
      command␣\bslash␣ppauza␣(en␣dash)␣not␣intended␣for␣vmode.}{%
      Use␣\bslash␣ppauza␣(en␣dash)␣only␣in␣number␣and␣numeral␣
          ranges.}%
    \else
      \afterfi{%
        \@ifnextspace{\unskip\penalty10000\hskip0.2em␣plus0.1em%
            \relax
```

```
                     -\hskip.2em␣plus0.1em\ignorespaces}{\unskip%\discretionary{-}{-}{-}}}%
                 \fi}%
\ppauza         \DeclareRobustCommand*\ppauza{%
                 \ifvmode␣␣␣␣\PackageError{gmutils}{%
                   command␣\bslash␣ppauza␣(en␣dash)␣not␣intended␣for␣vmode.}{%
                   Use␣\bslash␣ppauza␣(en␣dash)␣only␣in␣number␣and␣numeral␣ranges.}%
                 \else
                   \unskip\discretionary{-}{-}{-}%
                 \fi}%
\emdash        \def\emdash{\char`\—}
              }% of at begin document
\longpauza    \def\longpauza{\def\pauzacore{—}}
\pauzacore    \longpauza
\shortpauza   \def\shortpauza{%
\pauzacore      \def\pauzacore{-\kern,23em\relax\llap{-}}}
              \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 −.

```
              \ifdefined\XeTeXversion
              \foone{\catcode`—\active␣\catcode`–\active␣\catcode`-\active}{%
\adashes        \def\adashes{\AtBeginDocument\adashes}% because \pauza is defined at begin document.
\adashes        \AtBeginDocument{\def\adashes{%
                    \catcode`—\active␣\let—\—%
                    \catcode`–\active␣\let–\–%
              }}}
              \else
              \relaxen\adashes
              \fi
```

The hyphen shouldn't be active IMO because it's used in TeX control such as \hskip-2pt. Therefore we provide the \ahyphen declaration reluctanly, 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    \def\gmu@dywiz{\ifmmode-\else
                \ifvmode-\else\afterfifi\dywiz\fi\fi}%

              \foone{\catcode`-\active}{%
\ahyphen        \def\ahyphen{\let-\gmu@dywiz\catcode`\-\active}}
```

To get current time. Works in ε-TeXs, icluding XeTeX.

```
\czas         \newcommand*\czas[1][.]{%
                \the\numexpr(\time-30)/60\relax#1%
                \@tempcnta=\numexpr\time-(\time-30)/60*60\relax
                \ifnum\@tempcnta<10␣0\fi\the\@tempcnta}
```

To push the stuff up to the header and have the after heading skip after the stuff

```
\przeniesvskip \long\def\przeniesvskip#1{%
                 \edef\gmu@LastSkip{\the\lastskip}%
                 \vskip-\gmu@LastSkip\relax
```

```
               \vspace*{osp}%
               #1\vskip\gmu@LastSkip\relax}
\textbullet  \@ifXeTeX{\chardef\textbullet="2022␣}{\def\textbullet{$\bullet$}}
tytulowa     \newenvironment*{tytulowa}{\newpage}{\par\thispagestyle{empty}%
                  \newpage}
```

Nazwisko na stronę redakcyjną

```
\nazwired    \def\nazwired{\quad\textsc}
```

Settings for mathematics in main font

I used this terrible macros while typesetting E. Szarzyński's *Letters* in 2008.

```
\gmath       \def\gmath{%
               \def\do##1{\edef##1{{\@nx\mathit{\@xa\@gobble\string##1}}}}%
               \do\A␣\do\a␣\do\B␣\do\b␣\do\c␣\do\C\do\d␣\do\D␣\do\e␣\do\E\do\f
               \do\F\do\g\do\G␣\do\i\do\I␣\do\j\do\J␣\do\k\do\K␣\do\l␣\do\L␣%
                    \do\m
               \do\M␣\do\n␣\do\N␣\do\P␣\do\p␣\do\q␣\do\Q␣\do\R␣\do\r
               \let\sectionsign\S␣\do\S␣\do\s␣\do\T␣\do\t␣\do\u␣\do\U␣\do\v%
                    \do\V
               \do\w␣\do\W␣\do\x␣\do\X␣\do\Y␣\do\y␣\do\z\do\Z
               \def\do##1{\edef##1{{\@nx\mathrm{\@xa\@gobble\string##1}}}}%
               \do\0\do\1\do\2\do\3\do\4\do\5\do\6\do\7\do\8\do\9%
               \relaxen\do
               \newcommand*\do[4][\mathit]{\def##2{##3{##1{\char"##4}}}}%
               \do\alpha{}{03B1}%
               \do[\mathrm]\Delta{}{0394}%
               \do\varepsilon{}{03B5}%
               \do\vartheta{}{03D1}%
               \do\nu{}{03BD}%
               \do\pi{}{03C0}%
               \do\phi{}{03D5}%
               \do[\mathrm]\Phi{}{0424}%
               \do\sigma{}{03C3}%
               \do\varsigma{}{03DA}%
               \do\psi{}{03C8}%
               \do\omega{}{03C9}%
               \do\infty{}{221E}%
               \do[\mathrm]\neg{\mathbin}{00AC}%
               \do[\mathrm]\neq{\mathrel}{2260}%
               \do\partial{}{2202}%
               \do[\mathrm]\pm{}{00B1}%
               \do[\mathrm]\pm{\mathbin}{00B1}%
               \do[\mathrm]\sim{\mathrel}{007E}%
               \def\do##1##2##3{\def##1{%
\mathop            \mathop{\mathchoice{\hbox{%
                         \rm
                         \edef\gma@tempa{\the\fontdimen8\font}%
                       \larger[3]%
                       \lower\dimexpr(\fontdimen8\font-\gma@tempa)/2␣%
                       \hbox{##2}}}{\hbox{%
```

```
          \rm
          \edef\gma@tempa{\the\fontdimen8\font}%
          \larger[2]%
          \lower\dimexpr(\fontdimen8\font-\gma@tempa)/2␣%
          \hbox{##2}}}%
        {\mathrm{##2}}{\mathrm{##2}}}##3}}%
  \do\sum{\char"2211}{}%
  \do\forall{\gma@quantifierhook␣\rotatebox[origin=c]{180}{A}%
    \setbox0=\hbox{A}\setbox2=\hbox{\scriptsize␣x}%
    \kern\dimexpr\ht2/3*2␣-\wd0/2\relax}{\nolimits}%
  \do\exists{\rotatebox[origin=c]{180}{\gma@quantifierhook␣E}}%
      \nolimits%
  \def\do##1##2##3{\def##1{##3{%
\mathchoice      \mathchoice{\hbox{\rm##2}}{\hbox{\rm##2}}%
        {\hbox{\rm\scriptsize##2}}{\hbox{\rm\tiny##2}}}}}%
    \do\vee{\rotatebox[origin=c]{90}{<}}\mathbin
    \do\wedge{\rotatebox[origin=c]{-90}{<}}\mathbin
    \do\leftarrow{\char"2190}\mathrel
    \do\rightarrow{\char"2192}\mathrel
    \do\leftrightarrow{\char"2190\kern-0,1em␣\char"2192}\mathrel
   \def\do##1##2##3{%
     \catcode`##1=12\relax
     \scantokens{\mathcode`##1="8000\relax
       \foone{\catcode`##1=\active}{\def##1}{##3{%
\mathchoice          \mathchoice{\hbox{\rm##2}}{\hbox{\rm##2}}%
          {\hbox{\rm\scriptsize##2}}{\hbox{\rm\tiny##2}}}}%
       \ignorespaces}}% to eat the lineend (scantokens acts as \read icluding
           line end.
   \do..\mathpunct␣␣\do,,\mathpunct␣␣\do……\mathpunct
   \do((\mathopen
   \@ifundefined{resetMathstrut@}{}{% an error occured 'bad mathchar etc.'
         because amsmath.sty doesn't take account of a possibility of ( ) being math-
         active.
\resetMathstrut@     \def\resetMathstrut@{%
       \setbox\z@\hbox{%
```

%% \mathchardef\@tempa\mathcode`\(\relax %% \def\@tempb##1"##2##3{% \the\textfont"##3\char"}% %% \expandafter\@tempb\meaning\@tempa \relax

```
         (}%
       \ht\Mathstrutbox@\ht\z@␣\dp\Mathstrutbox@\dp\z@
     }}%
   \do))\mathclose
   \do[[\mathopen\do]]\mathclose
   \do-{\char"2212}\mathbin␣␣\do++\mathbin␣␣\do==\mathrel␣␣\do××%
       \mathbin
   \do::\mathbin␣␣\do··\mathbin␣␣\do//\mathbin␣\do<<\mathrel
   \do>>\mathrel
   \def\do##1##2##3{\def##1####1{##2{\hbox{%
           \rm
           \setbox0=\hbox{####1}%
           \edef\gma@tempa{\the\ht0}%
           \edef\gma@tempb{\the\dp0}%
           ##3%
```

```
                                \setbox0=\hbox{####1}%
                                \lower\dimexpr(\ht0␣+␣\dp0)/2-\dp0␣-((\gma@tempa+%
                                     \gma@tempb)/2-\gma@tempb)␣%
                                \box0}}}%
                       \do\bigl\mathopen\larger
                       \do\bigr\mathclose\larger
                       \do\Bigl\mathopen\largerr
                       \do\Bigr\mathclose\largerr
                       \do\biggl\mathopen{\larger[3]}%
                       \do\biggr\mathclose{\larger[3]}%
                       \do\Biggl\mathopen{\larger[4]}%
                       \do\Biggr\mathclose{\larger[4]}%
                       \def\do##1##2{\def##1{\ifmmode##2{\mathchoice
                             {\hbox{\rm\char`##1}}{\hbox{\rm\char`##1}}%
                             {\hbox{\rm\scriptsize\char`##1}}{\hbox{\rm\tiny%
                                  \char`##1}}}%
                         \else\char`##1\fi}}%
                     \StoreMacros{\{\}}%
                       \do\{\mathopen
                       \do\}\mathclose
                       \def\={\mathbin{=}}%
              \neqb    \def\neqb{\mathbin{\neq}}%
                       \def\do##1{\edef\gma@tempa{%
               \@xa        \def\@xa\@nx\csname␣\@xa\gobble\string##1r\endcsname{%
                             \@nx\mathrel{\@nx##1}}}%
                         \gma@tempa}%
                       \do\vee␣\do\wedge␣\do\neg
            \fakern    \def\fakern{\mkern-3mu}%
                       \thickmuskip=8mu␣plus␣4mu\relax
                       \gma@gmathhook
                     }% of def gmath

                     \emptify\gma@quantifierhook
   \quantifierhook   \def\quantifierhook#1{%
\gma@quantifierhook    \def\gma@quantifierhook{#1}}

                     \emptify\gma@gmathhook
         \gmathhook  \def\gmathhook#1{\addtomacro\gma@gmathhook{#1}}

        \gma@dollar  \def\gma@dollar$#1${{\gmath$#1$}}%
          \gma@bare  \def\gma@bare#1{\gma@dollar$#1$}%
  \gma@checkbracket  \def\gma@checkbracket{\@ifnextchar\[%
                       \gma@bracket\gma@bare}
       \gma@bracket  \def\gma@bracket\[#1\]{{\gmath\[#1\]}\@ifnextchar\par{}{%
                          \noindent}}
               \gma  \def\gma{\@ifnextchar$%
                       \gma@dollar\gma@checkbracket}

          \garamath  \def\garamath{%
                       \quantifierhook{\addfontfeature{OpticalSize=800}}%
     \gma@arrowdash    \def\gma@arrowdash{{%
                           \setbox0=\hbox{\char"2192}\copy0\kern-0,6\wd0
                           \bgcolor\rule[-\dp0]{0,6\wd0}{\dimexpr\ht0+\dp0}\kern-0,6%
                                \wd0}}%
     \gma@gmathhook    \def\gma@gmathhook{%
```

```
                    \def\do####1####2####3{\def####1{####3{%
\mathchoice               \mathchoice{\hbox{\rm####2}}{\hbox{\rm####2}}%
                          {\hbox{\rm\scriptsize####2}}{\hbox{\rm\tiny####2}}}}}%
                    \do\mapsto{\rule[0,4ex]{0,1ex}{0,4ex}\kern-0,05em%
                      \gma@arrowdash\kern-0,05em\char"2192}\mathrel
                    \do\cup{\scshape␣u}\mathbin
                    \do\varnothing{\setbox0=\hbox{\gma@quantifierhook%
                         \addfontfeature{Scale=1.272727}0}%
                      \setbox2=\hbox{\char"2044}%
                      \copy0␣\kern-0,5\wd0␣\kern-0,5\wd2␣\lower0,125\wd0␣\copy2
                      \kern0,5\wd0\kern-0,5\wd2}{}%
                    \do\leftarrow{\char"2190\kern-0,05em\gma@arrowdash}\mathrel
                    \do\rightarrow{\gma@arrowdash\kern-0,05em\char"2192}\mathrel
                    \do\in{\gma@quantifierhook\char"0454}\mathbin
                  }}
```

Typesetting dates in my memoirs

A date in the YYYY-MM-DD format we'll transform into DD mmmm YYYY format or we'll just typeset next two tokens/{...} if the arguments' string begins with --. The latter option is provided to preserve compatibility with already used macros and to avoid a starred version of \thedate and the same time to be able to turn \datef off in some cases (for SevSev04.tex).

```
\polskadata     \newcommand*\polskadata{%
     \datef       \def\datef##1-##2-##3##4{%
                    \if\relax##2\relax##3##4%
                    \else
                    \ifnum##3##4=0\relax
                    \else
                      \ifnum##3=0\relax
                      \else##3%
                      \fi##4%
                    \fi
                    \ifcase##2\relax\or\ stycznia\or\ lutego%
                      \or\ marca\or\ kwietnia\or\ maja\or\ czerwca\or\ lipca\or%
                           \ sierpnia%
                      \or\ września\or\ października\or\ listopada\or\ grudnia%
                           \else
                      {}%
                    \fi
                    \if\relax##1\relax\else\ \fi␣##1%
                  \fi}%
  \datefsl      \def\datefsl##1/##2/##3##4{%
                  \if\relax##2\relax##3##4%
                  \else
                    \ifnum##3##4=0\relax
                    \else
                      \ifnum##3=0\relax
                      \else##3%
                      \fi##4%
                    \fi
                    \ifcase##2\relax\or\ stycznia\or\ lutego%
```

```
        \or\ marca\or\ kwietnia\or\ maja\or\ czerwca\or\ lipca\or%
             \ sierpnia%
        \or\ września\or\ października\or\ listopada\or\ grudnia%
             \else
        {}%
      \fi
      \if\relax##1\relax\else\ \fi␣##1%
    \fi}%
}% of \polskadata
```

```
\polskadata
```

For documentation in English:

```
\englishdate  \newcommand*\englishdate{%
\datef          \def\datef##1-##2-##3##4{%
                  \if\relax##2\relax##3##4%
                  \else
                    \ifcase##2\relax\or␣January\or␣February%
                      \or␣March\or␣April\or␣May\or␣June\or␣July\or␣August%
                      \or␣September\or␣October\or␣November\or␣December\else
                      {}%
                    \fi
                    \ifnum##3##4=0\relax
                    \else
                      \ %
                      \ifnum##3=0\relax
                      \else##3%
                      \fi##4%
                      \ifcase##3##4\relax\or␣st\or␣nd\or␣rd\else␣th\fi
                    \fi
                    \if\relax##1\relax\else,\ \fi␣##1%
                  \fi
                }%
\datefsl        \def\datefsl##1/##2/##3##4{%
                  \if\relax##2\relax##3##4%
                  \else
                    \ifcase##2\relax\or␣January\or␣February%
                      \or␣March\or␣April\or␣May\or␣June\or␣July\or␣August%
                      \or␣September\or␣October\or␣November\or␣December\else
                      {}%
                    \fi
                    \ifnum##3##4=0\relax
                    \else
                      \ %
                      \ifnum##3=0\relax
                      \else##3%
                      \fi##4%
                      \ifcase##3##4\relax\or␣st\or␣nd\or␣rd\else␣th\fi
                    \fi
                    \if\relax##1\relax\else,\ \fi␣##1%
                  \fi
                }%
              }
```

```
\ifgmu@dash   \newif\ifgmu@dash
```

\gmu@ifnodash

```
\def\gmu@ifnodash#1-#2\@@nil{%
  \def\@tempa{#2}%
  \ifx\@tempa\@empty}
```

\gmu@testdash

```
\def\gmu@testdash#1\ifgmu@dash{%
  \gmu@ifnodash#1-\@@nil
    \gmu@dashfalse
  \else
    \gmu@dashtrue
  \fi
  \ifgmu@dash}
```

A word of explanation to the above pair of macros. `\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`\fi`ed) `\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 `\expandaftered`.

Dates for memoirs to be able to typeset them also as diaries.

\ifdate

```
\newif\ifdate
```

```
%\newcounter{dateinsection}[section]
```

\data

```
\newcommand*{\data}[1]{%
  \ifdate\gmu@testdash#1\ifgmu@dash\datef#1\else\datefsl#1\fi\fi}
```

\linedate

```
\newcommand*{\linedate}[1]{\par\ifdate\addvspace{\dateskip}%
  \date@line{\footnotesize\itshape␣\date@biway{#1}}%
  \nopagebreak\else% %\ifnum\arabic{dateinsection}>0\dekbigskip\fi
  \addvspace{\bigskipamount}%
  \fi}% end of \linedate.
```

```
\let\dateskip\medskipamount
```

\date@biway

```
\def\date@biway#1{%
  \gmu@testdash#1\ifgmu@dash\datef#1\else\datefsl#1\fi}
```

\rdate \ldate \runindate

```
\newcommand*\rdate[1]{\let\date@line\rightline␣\linedate{#1}}
\newcommand*\ldate[1]{\let\date@line\leftline␣\linedate{#1}}
\newcommand*\runindate[1]{%
  \paragraph{\footnotesize\itshape␣\datef#1\@@nil}\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.

```
\let\thedate\ldate
```

\zwrobcy

```
\DeclareRobustCommand*\zwrobcy[1]{\emph{#1}}␣% ostinato, allegro con moto,
        garden party etc., także kompliment
```

\tytul

```
\DeclareRobustCommand*\tytul[1]{\emph{#1}}
```

Maszynopis w świecie justowanym zrobi delikatną chorągiewkę.

maszynopis

```
\newenvironment{maszynopis}[1][]{#1\ttfamily
  \hyphenchar\font=45\relax% to przypisanie jest globalne do fontu.
  \@tempskipa=\glueexpr\rightskip+\leftskip\relax
  \ifdim\gluestretch\@tempskipa=\z@
  \tolerance900
```

sprawdziło się przy tolerancji 900

```
  \advance\rightskip␣by\z@␣plus0,5em\relax\fi
  \fontdimen3\font=\z@% zabraniamy rozciągania odstępów, ale %  \fontdimen4%
        \font=\z@ dopuszczamy ich skurczenie
  \hyphenpenalty0␣% żeby nie stresować TEXa: w maszynopisie ten wspaniały al-
        gorytm dzielenia akapitu powinien być wyłączony, a każdy wiersz łamany
        na ostatnim dopuszczalnym miejscu przełamania.
  \StoreMacro\pauzacore
```

\pauzacore

```
  \def\pauzacore{-\rlap{\kern-0,3em-}-}%
}{\par}
```

\justified

```
\newcommand*\justified{%
  \leftskip=1\leftskip% to preserve the natural length and discard stretch and
        shrink.
  \rightskip=1\rightskip
  \parfillskip=1\parfillskip
  \advance\parfillskip␣by␣0sp␣plus␣1fil\relax
  \let\\\@normalcr}
```

For dati under poems.

\wherncore

```
\newcommand\wherncore[1]{%
    \rightline{%
    \parbox{0,7666\textwidth}{
      \leftskip0sp␣plus␣\textwidth
      \parfillskip0sp\relax
      \let\\\linebreak
      \footnotesize␣#1}}}
```

\whern

```
\newcommand\whern[1]{%
  \vskip\whernskip
  \wherncore{#1}}
```

\whernskip

```
\newskip\whernskip
\whernskip2\baselineskip␣minus␣2\baselineskip\relax
```

\whernup

```
\newcommand\whernup[1]{\par\wherncore{#1}}
```

Minion and Garamond Premier kerning and ligature fixes

„Ws" nie będzie robiło długiego „s", bo źle wygląda przy „W"

\Ws

```
\DeclareRobustCommand*\Ws{W\kern-0,08em\penalty10000\hskip0sp%
      \relax
  s\penalty10000\hskip0sp\relax}
```

\Wz

```
\DeclareRobustCommand*\Wz{W\kern-0,05em\penalty10000\hskip0sp%
    \relax␣z}
\endinput
```

d. The gmiflink Package[1]

Written by Grzegorz 'Natror' Murzynowski,
natror at o2 dot pl

```
44 \NeedsTeXFormat{LaTeX2e}
45 \ProvidesPackage{gmiflink}
46     [2006/08/16␣v0.97␣Conditionally␣hyperlinking␣package␣(GM)]
```

Introduction, usage

This package protects you against an error when a link is dangling and typesets some plain text instead of a hyperlink then. It is intended for use with the hyperref package. Needs *two* LaTeX runs.

I used it for typesetting the names of the objects in a documentation of a computer program. If the object had been defined a `\hyperlink` to its definition was made, otherwise a plain object's name was typeset. I also use this package in authomatic making of hyperlinking indexes.

The package provides the macros `\gmiflink`, `\gmifref` and `\gmhypertarget` for conditional making of hyperlinks in your document.

\gmhypertarget `\gmhypertarget` `[`⟨*name*⟩`]` `{`⟨*text*⟩`}` makes a `\hypertarget{`⟨*@name*⟩`}{`⟨*text*⟩`}` and a `\label{`⟨*@name*⟩`}`.

\gmiflink `\gmiflink` `[`⟨*name*⟩`]` `{`⟨*text*⟩`}` makes a `\hyperlink{`⟨*@name*⟩`}{`⟨*text*⟩`}` to a proper hypertarget if the corresponding *label* exists, otherwise it typesets ⟨*text*⟩.

\gmifref `\gmifref` `[`⟨*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 ommitted.

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 v0.97 dated 2006/08/16.

gmiflink.pdf
gmiflink.tds.zip

The Code

```
\@ifpackageloaded{hyperref}{}{\message␣{^^J^^J␣gmiflink␣package:
    There's␣no␣use␣of␣me␣without␣hyperref␣package,␣I␣end␣my␣
         input.^^J}\endinput}
```

```
\providecommand\empty{}
```

A new counter, just in case

GMhlabel

```
\newcounter{GMhlabel}
\setcounter{GMhlabel}{0}
```

The macro given below creates both hypertarget and hyperlabel, so that you may reference both ways: via \hyperlink and via \ref. It's pattern is the \label macro, see LaTeX Source2e, 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

```
\DeclareRobustCommand*\gmhypertarget{%
  \@ifnextchar{[}{\gm@hypertarget}{\@dblarg{\gm@hypertarget}}}
```

\gm@hypertarget

```
\def\gm@hypertarget[#1]#2{% If argument #1 = \empty, then we'll use #2, i.e.,
        the same as name of hypertarget.
  \refstepcounter{GMhlabel}% we \label{\gmht@firstpar}
  \hypertarget{#1}{#2}%
  \protected@write\@auxout{}{%
    \string\newlabel{#1}{{#2}{\thepage}{\relax}{GMhlabel.%
         \arabic{GMhlabel}}{}}}%
}% end of \gm@hypertartget.
```

We define a macro such that if the target exists, it makes \ref, else it typesets ordinary text.

\gmifref

```
\DeclareRobustCommand*\gmifref{\@ifnextchar{[}{\gm@ifref}{%]
    \@dblarg{\gm@ifref}}}
```

\gm@ifref

```
\def\gm@ifref[#1]#2{%
  \expandafter\ifx\csname␣r@#1\endcsname\relax\relax%
  #2\else\ref{#1}\fi%
}% end of \gm@ifref
```

\gmiflink

```
\DeclareRobustCommand*\gmiflink{\@ifnextchar{[}{\gm@iflink}{%
    \@dblarg{\gm@iflink}}}
```

\gm@iflink

```
\def\gm@iflink[#1]#2{%
  \expandafter\ifx\csname␣r@#1\endcsname\relax\relax%
  #2\else\hyperlink{#1}{#2}\fi%
}% end of \gm@iflink
```

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.

```
\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 Package[1]

August 13, 2008

This is (a documentation of) file gmverb.sty, intended to be used with LaTeX 2$_\varepsilon$ 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

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

```
\NeedsTeXFormat{LaTeX2e}
\ProvidesPackage{gmverb}
    [2008/08/11␣v0.88␣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\verbatim\example| instead of \verb|\a\verbatim\example|.

The gmverb package redefines the \verb command and the verbatim environment in such a way that , { and \ are breakable, the first with no 'hyphen' and the other two with the comment char as a hyphen. I.e. {⟨*subsequent text*⟩} breaks into {%
⟨*subsequent text*⟩} and ⟨*text*⟩\mymacro breaks into ⟨*text*⟩%
\mymacro.

\fixbslash (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 \fixlbrace left brace: \fixlbrace.)

The default 'hyphen' is % since it's the default comment char. If you wish another \VerbHyphen char to appear at the linebreak, use the \VerbHyphen declaration that takes \⟨*char*⟩ as the only argument. This declaration is always global.

\verbeolOK Another difference is the \verbeolOK declaration (OCSR). Within its scope, \verb allows an end of a line in its argument and typesets it just as a space.

[1] This file has version number v0.88 dated 2008/08/11.

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
\dekclubs format, gmverb provides a macro \dekclubs as a shorthand for \MakeShortVerb*\|.

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
\DeleteShortVerb 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 in the math mode behave as if they were 'other', so you may type e.g., $|$ to get | and + \activeated this way is in the math mode typeset properly etc.

\OldMakeShortVerb 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
\dekclubs* for \MakeShortVerb\| and \dekclubs* for \OldMakeShortVerb\|.

So that, after the latter declaration, you can write

\[|⟨*verbatim stuff*⟩|\]

instead of

\[\hbox{|⟨*the stuff*⟩|}\]

to get a displayed shortverb.

Both versions of \dekclubs OCSR.

The verbatim environment inserts \topsep before and after itself, just as in standard version (as if it was a list).

In August 2008 Will Robertson suggested grey visible spaces for gmdoc. I added a respective option to gmdoc but I find them so nice that I want to make them available for
\VisSpacesGrey 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 ommitted.

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

```
243 \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      255 \let\verbhyphen\xiipercent
```

Provide a declaration for easy changing it. Its argument should be of \⟨*char*⟩ form (of course, a ⟨*char*⟩$_{12}$is also allowed).

```
\VerbHyphen      261 \def\VerbHyphen#1{%
                 262   {\escapechar\m@ne
                 263      \@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'.

```
\breaklbrace     272 \def\breaklbrace{%
                 273   \discretionary{\xiilbrace\verbhyphen}{}{\xiilbrace}}

                 276 \foone{\catcode`\[=1␣\catcode`\{=\active␣\catcode`\]=2␣}%
                 277 [%
\dobreaklbrace   278    \def\dobreaklbrace[\catcode`\{=\active
                 279    \def{%
\breaklbrace     280       [\breaklbrace\gm@lbracehook]]%
                 281 ]
```

Now we only initialize the hook. Real use of it will be made in gmdoc.

```
                 285 \relaxen\gm@lbracehook
```

The \bslash macro defined below I use also in more 'normal' TEXing, e.g., to \typeout some \outer macro's name.

```
                 290 \foone{\catcode`\!=0␣\@makeother\\}%
                 291 {%
\bslash          292   !def!bslash{\}%
\breakbslash     293   !def!breakbslash{!discretionary{!verbhyphen}{\}{\}}%
                 294 }
```

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 608. For the others let's give the user a countercrank:

```
\fixbslash       301 \newcommand*\fixbslash{\let\breakbslash=\bslash}%
```
to use due to the common scoping rules. But for the special case of a backslash opening a verbatim scope, we deal specially in the line 608.

Analogously, let's provide a possibility of 'fixing' the left brace:

```
\fixlbrace       307 \newcommand*\fixlbrace{\let\breaklbrace=\xiilbrace}

                 310 \foone{\catcode`\!=0␣\catcode`\\=\active}%
```

\dobreakbslash
\breakbslash

```
{%
  !def!dobreakbslash{!catcode`!\=!active␣!def\{!breakbslash}}%
}
```

The macros defined below, `\visiblebreakspaces` and `\xiiclub` we'll use in the almost Knuthian macro making verbatim. This 'almost' makes a difference.

```
\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 xltxtra if available.
```

\breakablevisspace

```
\def\breakablevisspace{\discretionary{\visiblespace}{}{%
        \visiblespace}}
```

\activespace
\dobreakvisiblespace
\breakablevisspace
\dobreakblankspace

```
\foone\obeyspaces% it's just re\catcode'ing.
{%
\newcommand*\activespace{␣}%
\newcommand*\dobreakvisiblespace{\def␣{\breakablevisspace}\obeyspaces}%
        % \defing it caused a stack overflow disaster with gmdoc.
\newcommand*\dobreakblankspace{\let␣=\space\obeyspaces}%
}
```

\xiiclub

```
\bgroup\@makeother\|
\firstofone{\egroup\def\xiiclub{|}}
```

Almost-Knuthian \ttverbatim

`\ttverbatim` comes from *The TEXbook* too, but I add into it a LaTeX macro changing the `\catcode`s and make spaces visible and breakable and left braces too.

\ttverbatim

```
\newcommand*\ttverbatim{%
  \let\do=\do@noligs␣\verbatim@nolig@list
  \let\do=\@makeother␣\dospecials
  \dobreaklbrace\dobreakbslash
  \dobreakspace
  \tt
  \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.

```
\@emptify\ttverbatim@hook
```

\VerbT1
\VerbT
\ttverbatim@hook

```
\def\VerbT1{\def\ttverbatim@hook{\fontencoding{T1}\selectfont}}
```

We wish the visible spaces to be the default.

```
\let\dobreakspace=\dobreakvisiblespace
```

The Core: From shortvrb

The below is copied verbatim ;-) from doc.pdf and then is added my slight changes.

\MakeShortVerb
\@shortvrbdef
\@shortvrbdef

```
\def\MakeShortVerb{%
  \@ifstar
  {\def\@shortvrbdef{\verb*}\@MakeShortVerb}%
  {\def\@shortvrbdef{\verb}\@MakeShortVerb}}
```

\@MakeShortVerb

```
\def\@MakeShortVerb#1{%
  \@xa\ifx\csname␣cc\string#1\endcsname\relax
  \@shortvrbinfo{Made␣}{#1}\@shortvrbdef
  \add@special{#1}%
  \AddtoPrivateOthers#1% a macro to be really defined in gmdoc.
  \@xa
  \xdef\csname␣cc\string#1\endcsname{\the\catcode`#1}%
  \begingroup
  \catcode`\~\active␣\lccode`\~`#1%
  \lowercase{%
    \global\@xa\let
    \csname␣ac\string#1\endcsname~%
    \@xa\gdef\@xa~\@xa{%
      \@xa\ifmmode\@xa\string\@xa~%
      \@xa\else\@xa\afterfi{\@shortvrbdef~}\fi}}% This terrible number
             of \expandafters is to make the shortverb char just other in the math
             mode (my addition).
  \endgroup
  \global\catcode`#1\active
 \else
 \@shortvrbinfo\@empty{#1␣already}{\@empty\verb(*)}%
 \fi}
```

\DeleteShortVerb

```
\def\DeleteShortVerb#1{%
  \@xa\ifx\csname␣cc\string#1\endcsname\relax
  \@shortvrbinfo\@empty{#1␣not}{\@empty\verb(*)}%
  \else
  \@shortvrbinfo{Deleted␣}{#1␣as}{\@empty\verb(*)}%
  \rem@special{#1}%
  \global\catcode`#1\csname␣cc\string#1\endcsname
  \global␣\@xa\let␣\csname␣cc\string#1\endcsname␣\relax
  \ifnum\catcode`#1=\active
  \begingroup
  \catcode`\~\active␣\lccode`\~`#1%
  \lowercase{%
    \global\@xa\let\@xa~%
    \csname␣ac\string#1\endcsname}%
  \endgroup␣\fi␣\fi}
```

My little addition

\gmv@packname
\gmv@packname

```
\@ifpackageloaded{gmdoc}{%
  \def\gmv@packname{gmdoc}}{%
  \def\gmv@packname{gmverb}}
```

\@shortvrbinfo

```
\def\@shortvrbinfo#1#2#3{%
  \PackageInfo{\gmv@packname}{%
   ^^J\@empty␣#1\@xa\@gobble\string#2␣a␣short␣reference
    for␣\@xa\string#3}}
```

\add@special

```
\def\add@special#1{%
  \rem@special{#1}%
  \@xa\gdef\@xa\dospecials\@xa
  {\dospecials␣\do␣#1}%
  \@xa\gdef\@xa\@sanitize\@xa
  {\@sanitize␣\@makeother␣#1}}
```

For the commentary on the below macro see the doc package's documentation. Here let's only say it's just amazing: so tricky and wicked use of \do. The internal macro

\rem@special defines \do to expand to nothing if the \do's argument is the one to be removed and to unexpandable CSs \do and ⟨\do's argument⟩ otherwise. With \do defined this way the entire list is just globally expanded itself. Analogous hack is done to the \@sanitize list.

\rem@special

```
\def\rem@special#1{%
  \def\do##1{%
    \ifnum`#1=`##1␣\else␣\@nx\do\@nx##1\fi}%
  \xdef\dospecials{\dospecials}%
  \begingroup
  \def\@makeother##1{%
    \ifnum`#1=`##1␣\else␣\@nx\@makeother\@nx##1\fi}%
  \xdef\@sanitize{\@sanitize}%
  \endgroup}
```

And now the definition of verbatim itself. As you'll see (I hope), the internal macros of it look for the name of the current environment (i.e., \@currenvir's meaning) to set their expectation of the environment's \end properly. This is done to allow the user to define his/her own environments with \verbatim inside them. I.e., as with the verbatim package, you may write \verbatim in the begdef of your environment and then necessarily \endverbatim in its enddef. Of course (or *maybe surprisingly*), the commands written in the begdef after \verbatim will also be executed at \begin{⟨*environment*⟩}.

verbatim
\verbatim

```
\def\verbatim{%
  \edef\gmv@hyphenpe{\the\hyphenpenalty}%
  \edef\gmv@exhyphenpe{\the\exhyphenpenalty}%
  \@beginparpenalty␣\predisplaypenalty␣\@verbatim
  \frenchspacing␣\gmobeyspaces␣\@xverbatim
  \hyphenpenalty=\gmv@hyphenpe\relax
  \exhyphenpenalty=\gmv@exhyphenpe
  \hyphenchar\font=\m@ne}% in the LaTeX version there's %\@vobeyspaces in-
        stead of %\gmobeyspaces.
```

verbatim*

```
\@namedef{verbatim*}{\@beginparpenalty␣\predisplaypenalty␣%
     \@verbatim
  \@sxverbatim}
```

\endverbatim

```
\def\endverbatim{\@@par
  \ifdim\lastskip␣>\z@
    \@tempskipa\lastskip␣\vskip␣-\lastskip
    \advance\@tempskipa\parskip␣\advance\@tempskipa␣-%
          \@outerparskip
    \vskip\@tempskipa
  \fi
  \addvspace\@topsepadd
  \@endparenv}
```

```
\n@melet{endverbatim*}{endverbatim}
```

```
\begingroup␣\catcode␣`!=0␣%
\catcode␣`[=␣1␣\catcode`]=2␣%
\catcode`\{=\active
\@makeother\}%
\catcode`\\=\active%
```

\@xverbatim

```
!gdef!@xverbatim[%
  !edef!verbatim@edef[%
    !def!noexpand!verbatim@end%
      ####1!noexpand\end!noexpand{!@currenvir}[%
      ####1!noexpand!end[!@currenvir]]]%
  !verbatim@edef
  !verbatim@end]%
!endgroup
```

\@sxverbatim

```
\let\@sxverbatim=\@xverbatim
```

F. Mittelbach says the below is copied almost verbatim from LaTeX source, modulo \check@percent.

\@verbatim

```
\def\@verbatim{%
```

Originally here was just \trivlist \item[], but it worked badly in my document(s), so let's take just highlights of if.

```
  \parsep\parskip
```

From \@trivlist:

```
  \if@noskipsec␣\leavevmode␣\fi
  \@topsepadd␣\topsep
  \ifvmode
    \advance\@topsepadd␣\partopsep
  \else
    \unskip␣\par
  \fi
  \@topsep␣\@topsepadd
  \advance\@topsep␣\parskip
  \@outerparskip␣\parskip
```

(End of \trivlistlist and \@trivlist highlights.)

```
  \@@par\addvspace\@topsep
  \if@minipage\else\vskip\parskip\fi
  \leftmargin\parindent% please notify me if it's a bad idea.
  \advance\@totalleftmargin\leftmargin
  \raggedright
  \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(*).
  \@@par% most probably redundant.
  \@tempswafalse
  \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}%
  \everypar{\@tempswatrue\hangindent\verbatimhangindent\hangafter%
        \@ne}% since several chars are breakable, there's a possibility of breaking
        some lines. We wish them to be hanging indented.
  \obeylines
  \ttverbatim}
```

\stanzaskip

```
\@ifundefined{stanzaskip}{\newlength\stanzaskip}{}
\stanzaskip=\medskipamount
```

\verbatimhangindent

```
\newlength\verbatimhangindent
```

```
\verbatimhangindent=3em
```

\check@percent

```
\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

```
\providecommand*\AddtoPrivateOthers[1]{}
```

Both of the above are `\provide`d 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*
\verb

```
\def\verb{\relax\ifmmode\hbox\else\leavevmode\null\fi
  \bgroup
  \ttverbatim
  \gm@verb@eol
  \@ifstar{\@sverb@chbsl}{\gmobeyspaces\frenchspacing\@sverb@chbsl}}%
```

in the LaTeX version there's `\@vobeyspaces` instead of `\gmobeyspaces`.

\@sverb@chbsl

```
\def\@sverb@chbsl#1{\@sverb#1\check@bslash}
```

\@def@breakbslash

```
\def\@def@breakbslash{\breakbslash}% because \ is \defined as \breakbslash 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

```
\def\check@bslash{\@ifnextchar{\@def@breakbslash}{\bslash%
      \@gobble}{}}
```

```
\let\verb@balance@group\@empty
```

\verb@egroup

```
\def\verb@egroup{\global\let\verb@balance@group\@empty\egroup}
```

\gm@verb@eol

```
\let\gm@verb@eol\verb@eol@error
```

The latter is a LaTeX 2$_\varepsilon$ kernel macro that `\active`ates 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.

\check@percent

```
\begingroup
\obeylines\obeyspaces%
\gdef\verb@eolOK{\obeylines%
\def^^M{␣\check@percent}%
}%
\endgroup
```

The `\check@percent` macro here is `\provide`d to be `\@empty` but in gmdoc employed shall it be.

Let us leave (give?) a user freedom of choice:

\verbeolOK

```
\def\verbeolOK{\let\gm@verb@eol\verb@eolOK}
```

And back to the main matter,

```
\def\@sverb#1{%
  \catcode`#1\active␣\lccode`\~`#1%
  \gdef\verb@balance@group{\verb@egroup
    \@latex@error{Illegal␣use␣of␣\bslash␣verb␣command}\@ehc}%
  \aftergroup\verb@balance@group
  \lowercase{\let~\verb@egroup}}
```

\verbatim@nolig@list

```
\def\verbatim@nolig@list{\do\`\do\<\do\>\do\,\do\'\do\-}
```

\do@noligs

```
\def\do@noligs#1{%
  \catcode`#1\active
  \begingroup
  \lccode`\~=`#1\relax
  \lowercase{\endgroup\def~{\leavevmode\kern\z@\char`#1}}}
```

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

```
\def\dekclubs{\@ifstar{\OldMakeShortVerb\|}{\MakeShortVerb\|}}
```

But even if a shortverb is unconditional, the spaces in the math mode are not printed. So,

\edverbs

```
\newcommand*\edverbs{%
  \let\gmv@dismath\[%
  \let\gmv@edismath\]%
  \def\[{%
    \@ifnextac\gmv@disverb\gmv@dismath}%
  \relaxen\edverbs}%
```

\gmv@disverb

```
\def\gmv@disverb{%
  \gmv@dismath
  \hbox\bgroup\def\]{\egroup\gmv@edismath}}
```

doc- And shortvrb-Compatibility

One of minor errors while TEXing 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/shortvrb'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

```
\def\old@MakeShortVerb#1{%
  \@xa\ifx\csname␣cc\string#1\endcsname\relax
  \@shortvrbinfo{Made␣}{#1}\@shortvrbdef
  \add@special{#1}%
  \AddtoPrivateOthers#1% a macro to be really defined in gmdoc.
  \@xa
  \xdef\csname␣cc\string#1\endcsname{\the\catcode`#1}%
  \begingroup
  \catcode`\~\active␣\lccode`\~`#1%
  \lowercase{%
    \global\@xa\let\csname␣ac\string#1\endcsname~%
    \@xa\gdef\@xa~\@xa{%
      \@shortvrbdef~}}%
  \endgroup
  \global\catcode`#1\active
  \else
  \@shortvrbinfo\@empty{#1␣already}{\@empty\verb(*)}%
  \fi}
```

\OldMakeShortVerb

```
\def\OldMakeShortVerb{\begingroup
```

```
  \let\@MakeShortVerb=\old@MakeShortVerb
  \@ifstar{\eg@MakeShortVerbStar}{\eg@MakeShortVerb}}
```

\eg@MakeShortVerbStar

\eg@MakeShortVerb

```
\def\eg@MakeShortVerbStar#1{\MakeShortVerb*#1\endgroup}
\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{doument}` to recognize possible redefinition of it when xltxtra is loaded.

```
\let\gmd@preambleABD\AtBeginDocument
\AtBeginDocument{\let\gmd@preambleABD\firstofone}
```

```
\RequirePackage{xcolor}% for \providecolor
```

\VisSpacesGrey

```
\def\VisSpacesGrey{%
  \providecolor{visspacesgrey}{gray}{0.5}%
  \gmd@preambleABD{%
    \edef\visiblespace{%
      \hbox{\@nx\textcolor{visspacesgrey}%
        {\@xa\unexpanded\@xa{\visiblespace}}}}%
  }}
```

```
\endinput% for the Tradition.
```

f. The gmeometric Package[1]

Written by Grzegorz Murzynowski,
natror at o2 dot pl

```
\NeedsTeXFormat{LaTeX2e}
\ProvidesPackage{gmeometric}
     [2008/08/06␣v0.72␣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 claused `\@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 trigges the first while the starred the latter. If that doesn't work quite as expected, try to precede or succede 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 XeTeX and sets the proper driver if so. Probably it's redundant since decent XeTeX 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 ommitted.

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 v0.72 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

```
\RequirePackage{gmutils}[2007/04/23]% this package defines the storing and
        restoring commands.
```

redefine `\@onlypreamble`, add storing to BeginDocument.

\gme@tobestored

```
\newcommand*\gme@tobestored{{%
    \Gm@cnth␣\Gm@cntv␣\c@Gm@tempcnt␣\Gm@bindingoffset␣\Gm@wd@mp
    \Gm@odd@mp␣\Gm@even@mp␣\Gm@orgw␣\Gm@orgh␣\Gm@dimlist}}

\@xa\AtBeginDocument\@xa{\@xa\StoreMacros\gme@tobestored}

\StoreMacro\@onlypreamble
\let\@onlypreamble\@gobble
```

To make it work properly in XeTeX:

\pdfoutput

```
\@ifXeTeX{%
  \@ifundefined{pdfoutput}{\newcount\pdfoutput}{}%
  \PassOptionsToPackage{dvipdfm}{geometry}%
}{}

\RequirePackageWithOptions{geometry}
```

Restore `\@onlypreamble`:

```
\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

```
\def\Gm@showparams{%
  --------------------␣Geometry␣parameters^^J%
  \ifGm@pass
  'pass'␣is␣specified!!␣(disables␣the␣geometry␣layouter)^^J%
  \else
  paper:␣\ifx\Gm@paper\@undefined␣class␣default\else\Gm@paper%
       \fi^^J%
  \Gm@checkbool{landscape}%
  twocolumn:␣\if@twocolumn\Gm@true\else--\fi^^J%
  twoside:␣\if@twoside\Gm@true\else--\fi^^J%
  asymmetric:␣\if@mparswitch␣--\else\if@twoside\Gm@true\else␣--%
       \fi\fi^^J%
  h-parts:␣\Gm@lmargin,␣\Gm@width,␣\Gm@rmargin%
  \ifnum\Gm@cnth=\z@\space(default)\fi^^J%
  v-parts:␣\Gm@tmargin,␣\Gm@height,␣\Gm@bmargin%
  \ifnum\Gm@cntv=\z@\space(default)\fi^^J%
  hmarginratio:␣\ifnum\Gm@cnth<5␣\ifnum\Gm@cnth=3--\else%
    \Gm@hmarginratio\fi\else--\fi^^J%
  vmarginratio:␣\ifnum\Gm@cntv<5␣\ifnum\Gm@cntv=3--\else%
    \Gm@vmarginratio\fi\else--\fi^^J%
  lines:␣\ifx\Gm@lines\@undefined--\else\Gm@lines\fi^^J% here I (natror) fix the bug: it was \@ifundefined that of course was assigning % \relax to \Gm@lines and that resulted in an error when \geometry was used inside document.
  \Gm@checkbool{heightrounded}%
  bindingoffset:␣\the\Gm@bindingoffset^^J%
  truedimen:␣\ifx\Gm@truedimen\@empty␣--\else\Gm@true\fi^^J%
  \Gm@checkbool{includehead}%
  \Gm@checkbool{includefoot}%
  \Gm@checkbool{includemp}%
  driver:␣\Gm@driver^^J%
  \fi
  --------------------␣Page␣layout␣dimensions␣and␣switches^^J%
  \string\paperwidth\space\space\the\paperwidth^^J%
  \string\paperheight\space\the\paperheight^^J%
  \string\textwidth\space\space\the\textwidth^^J%
  \string\textheight\space\the\textheight^^J%
  \string\oddsidemargin\space\space\the\oddsidemargin^^J%
  \string\evensidemargin\space\the\evensidemargin^^J%
  \string\topmargin\space\space\the\topmargin^^J%
  \string\headheight\space\the\headheight^^J%
  \string\headsep\@spaces\the\headsep^^J%
  \string\footskip\space\space\space\the\footskip^^J%
  \string\marginparwidth\space\the\marginparwidth^^J%
  \string\marginparsep\space\space\space\the\marginparsep^^J%
  \string\columnsep\space\space\the\columnsep^^J%
  \string\skip\string\footins\space\space\the\skip\footins^^J%
  \string\hoffset\space\the\hoffset^^J%
  \string\voffset\space\the\voffset^^J%
  \string\mag\space\the\mag^^J%
  \if@twocolumn\string\@twocolumntrue\space\fi%
  \if@twoside\string\@twosidetrue\space\fi%
  \if@mparswitch\string\@mparswitchtrue\space\fi%
  \if@reversemargin\string\@reversemargintrue\space\fi^^J%
  (1in=72.27pt,␣1cm=28.45pt)^^J%
  -----------------------}
```

Add restore to BeginDocument:

```
\@xa\AtBeginDocument\@xa{\@xa\RestoreMacros\gme@tobestored}
```

```
\endinput
```

g. The gmoldcomm Package[1]

August 13, 2008

This is a package for handling the old comments in LaTeX 2ε 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.

```
                   \NeedsTeXFormat{LaTeX2e}
                   \ProvidesPackage{gmoldcomm}
                            [2007/11/10␣v0.99␣LaTeX␣old␣comments␣handling␣(GM)]
oldcomments        \newenvironment{oldcomments}{%
                     \catcode`\\=\active
                     \let\do\@makeother
                     \do\$% Not only css but also special chars occur in the old comments.
                     \do\|\do\#\do\{\do\}\do\^\do\_\do\&%
                     \gmoc@defbslash
                     \obeylines
                     \StoreMacro\finish@macroscan
\finish@macroscan    \def\finish@macroscan{%
                       \@xa\gmd@ifinmeaning\macro@pname\of\gmoc@notprinted%
                       {}{{\tt\ifvmode\%\fi\bslash\macro@pname}}%
                       \gmoc@checkenv
                     }%
                   }{}

                   {\escapechar\m@ne
                   \xdef\gmoc@notprinted{\string\begin,\string\end}}

\gmoc@maccname     \def\gmoc@maccname{macrocode}
\gmoc@ocname       \def\gmoc@ocname{oldcomments}

                   \foone{%
                     \catcode`\[=1␣\catcode`\]=2
                     \catcode`\{=12␣\catcode`\}=12␣}
\gmoc@checkenv     [\def\gmoc@checkenv[%
                     \@ifnextchar{%
                       [\gmoc@checkenvinn][]]%
\gmoc@checkenvinn  \def\gmoc@checkenvinn{#1}[%
\gmoc@resa           \def\gmoc@resa[#1]%
                     \ifx\gmoc@resa\gmoc@maccname
                       \def\next[%
                         \begingroup
```

[1] This file has version number v0.99 dated 2007/11/10.

\@currenvir

```
   \def\@currenvir[macrocode]%
   \RestoreMacro\finish@macroscan
   \catcode`\\=\z@
   \catcode`\{=1␣\catcode`\}=2
   \macrocode]%
 \else
  \ifx\gmoc@resa\gmoc@ocname
   \def\next[\end[oldcomments]]%
  \else
   \def\next[%
    \{#1\}%
   ]%
  \fi
 \fi
 \next]%
]

\foone{%
 \catcode`\/=\z@
 \catcode`\\=\active}
```

\gmoc@defbslash

```
{/def/gmoc@defbslash{%
  /let\/scan@macro}}
```

\task

```
\def\task#1#2{}

\endinput
```

Change History

gmdoc v0.96
- General:
  - CheckSum 2395, a-0

gmdoc v0.98d
- \ChangesStart:
  - An entry to show the change history works: watch and admire. Some sixty \changes entries irrelevant for the users-other-than-myself are hidden due to the trick described on p. 81. a-5807

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-7515

gmdoc v0.99c
- General:
  - A bug fixed in \DocInput and all \expandafters changed to \@xa and \noexpands to \@nx, a-7515
  - The TEX-related logos now are declared with \DeclareLogo provided in gmutils, a-7515
- \DocInput:
  - added ensuring the code delimiter to be the same at the end as at the beginning, a-2340
- \gmd@bslashEOL:
  - a bug fix: redefinition of it left solely to \QueerEOL, a-3267

gmdoc v0.99d
- General:
  - \@namelet renamed to \n@melet to solve a conflict with the beamer class (in gmutils at first), a-7515
  - \afterfi & pals made two-argument, a-7515
- \FileInfo:
  - added, a-6679

gmdoc v0.99e
- General:
  - a bug fixed in \DocInput and \IndexInput, a-7515
  - CheckSum 4574, a-0

gmdoc v0.99g
- General:
  - CheckSum 5229, a-0
  - The bundle goes XƎTEX. 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-7515
- \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-3491
- hyperref:
  - added bypass of encoding for loading url, a-2062
- \inverb:
  - added, a-6854
- \OldDocInput:
  - obsolete redefinition of the macro environment removed, a-7361

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-7515

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-7515
  - 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-6828

- gmdoc v0.99k
  - General:
    - CheckSum 5261, a-0
  - `hyperref`:
    - removed some lines testing if XƎTEX colliding with tikz and most probably obsolete, a-2080
- gmdoc v0.99l
  - General:
    - CheckSum 5225, a-0
  - `\CodeSpacesGrey`:
    - added due to Will Robertson's suggestion, a-2532
  - `codespacesgrey`:
    - added due to Will Robertson's suggestion, a-2041
  - `\gmd@FIrescan`:
    - `\scantokens` used instead of `\write` and `\@@input` which simplified the macro, a-6711
  - `macrocode`:
    - removed `\CodeSpacesBlank`, a-4903
  - `\SelfInclude`:
    - Made a shorthand for `\Docinclude\jobname` instead of repeating 99% of `\DocInclude`'s code, a-6419
- gmdoc v0.99m
  - `\@oldmacrocode`:
    - renamed from `\VerbMacrocodes`, a-4995
  - `^^M`:
    - there was `\let^^M` but `\QueerEOL` is better: it also redefines
- hathat M, a-2323
  - General:
    - CheckSum 5354, a-0
    - CheckSum 5356, a-0
    - Counting of all lines developed (the `countalllines` package option), now it uses `\inputlineno`, a-7515
  - `\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-4722
  - `\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-6355
  - `\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-6696
  - `\gmd@bslashEOL`:
    - also `\StraightEOL` with `countalllines` package option lets
- hathat M to it, a-3267
  - `\thefilediv`:
    - let to `\relax` by default, a-6569
  - `theglossary`:
    - added `\IndexLinksBlack`, a-5878
- gmdocc v0.74
  - `\edverbs`:
    - used to simplify displaying shortverbs, b-442
- gmdocc v0.75
  - General:
    - CheckSum 130, b-0
- gmdocc v0.76
  - General:
    - CheckSum 257, b-0
  - `\EOFMark`:
    - The `gmeometric` option made obsolete and the gmeometric package is loaded always, for XƎTEX-compatibility. And the class options go xkeyval., b-460
- gmdocc v0.77
  - General:
    - CheckSum 262, b-0
  - `\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-460
- gmdocc v0.78
  - General:
    - CheckSum 267, b-0
  - `\EOFMark`:
    - Added the `pagella` option not to use Adobe Minion Pro that is not freely licensed, b-460
- gmdocc v0.79
  - General:
    - CheckSum 271, b-0
- gmeometric v0.69
  - General:
    - CheckSum 40, f-0
- 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-0

gmeometric v0.71
General:
a TDS-compliant zip archive made, f-399
CheckSum 41, f-0

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-0
`\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-568
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-3234

gmutils v0.75
`\@ifnextac`:
added, c-424
`\@ifnextcat`:
`\let` for `#1` changed to `\def` to allow things like `\noexpand~` , c-363
`\@ifnextif`:
`\let` for `#1` changed to `\def` to allow things like `\noexpand~` , c-399

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-3234
`\freeze@actives`:
added, c-2318

gmutils v0.77
General:
`\afterfi` & pals made two-argument as the Marcin Woliński's analogoi are. At this occasion some redundant macros of that family are deleted, c-3234

gmutils v0.78
General:
`\@namelet` renamed to `\n@melet` to solve a conflict with the beamer class. The package contents regrouped, c-3234

gmutils v0.79
`\not@onlypreamble`:
All the actions are done in a group and therefore `\xdef` used instead of `\edef` because this command has to use `\do` (which is contained in the `\@preamblecmds` list) and `\not@onlypreamble` itself should be able to be let to `\do`, c-1179

gmutils v0.80
General:
CheckSum 1689, c-0
`\hfillneg`:
added, c-2239

gmutils v0.81
`\dekfraccslash`:
moved here from pmlectionis.cls, c-2497
`\ifSecondClass`:
moved here from pmlectionis.cls, c-2466

gmutils v0.82
`\ikern`:
added, c-2505

gmutils v0.83
`\~`:
postponed to `\begin{document}` to avoid overwriting by a text command and made sensible to a subsequent /, c-2192

gmutils v0.84
General:
CheckSum 2684, c-0

gmutils v0.85
General:
CheckSum 2795, c-0
fixed behaviour of too clever headings with gmdoc by adding an `\ifdim` test, c-3234

gmutils v0.86
`\texttilde`:
renamed from
texttilde since the latter is one of LaTeX accents, c-2200

gmutils v0.87
General:
CheckSum 4027, c-0
the package goes ε-TeX even more, making use of `\ifdefined` and the code usingUTF-8 chars is wrapped in a XeTeX-condition, c-3234

gmutils v0.88

- General:
  - CheckSum 4040, c-0
- `\RestoreEnvironment`:
  - added, c-1117
- `\storedcsname`:
  - added, c-1108
- `\StoreEnvironment`:
  - added, c-1113

gmutils v0.89
- General:
  - removed obsolete adjustment of pgf for XƎTEX, c-3234

gmutils v0.90
- General:
  - CheckSum 4035, c-0
- `\XeTeXthree`:
  - adjusted to the redefinition of `\verb` in xlxtra 2008/07/29, c-2017

gmutils v0.91
- General:
  - CheckSum 4055, c-0
  - removed `\jobnamewoe` since `\jobname` is always without extension. `\xiispace` forked to `\visiblespace` `\let` to `\xxt@visiblespace` of xltxtra if available. The documentation driver integrated with the .sty file, c-3234

gmutils v0.92
- `\@checkend`:
  - shortened thanks to `\@ifenvir`, c-598
- `\@gif`:
  - added redefinition so that now switches defined with it are `\protected` so they won't expand to an further expanding or unbalanced `\iftrue/false` in an edef, c-198
- `\@ifenvir`:
  - added, c-590
- General:
  - CheckSum 4133, c-0

gmverb v0.79
- `\edverbs`:
  - added, e-663

gmverb v0.80
- `\edverbs`:
  - debugged, i.e. `\hbox` added back and redefinition of `\[`, e-663
- `\ttverbatim`:
  - `\ttverbatim@hook` added, e-337

gmverb v0.81
- General:
  - `\afterfi` made two-argument (first undelimited, the stuff to be put after `\fi`, and the other, delimited with `\fi`, to be discarded, e-750

gmverb v0.82
- General:
  - CheckSum 663, e-0

gmverb v0.83
- General:
  - added a hook in the active left brace definition intended for gmdoc automatic detection of definitions (in line 280), e-750
  - CheckSum 666, e-0

gmverb v0.84
- General:
  - CheckSum 658, e-0

gmverb v0.85
- General:
  - added restoring of `\hyphenpenalty` and `\exhyphenpenalty` and setting `\hyphenchar=-1`, e-750
  - CheckSum 673, e-0

gmverb v0.87
- General:
  - CheckSum 661, e-0
  - visible space tidyied and taken from xltxtra if available. gmutils required. The `\xii...` cs'es moved to gmutils. The documentation driver moved into the .sty file, e-750

gmverb v0.88
- General:
  - CheckSum 682, e-0
- `\VisSpacesGrey`:
  - added, or rather moved here from gmdoc, e-737

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-3371, a-3655, a-5554, a-7284, c-452, c-453, c-454, c-456, c-460, c-461, c-462, c-466, c-2192, c-2373
`\+`, *21*, a-5554, a-6931, c-2374
`\-`, a-5554, c-915, c-2811, e-651
`\<...>`, c-843, *100*
`\@@codeline@wrindex`, a-4752
`\@@nil`, a-4566, a-4652, a-5074, a-5092, a-5711, a-5769, a-5772, a-5791, a-6708, c-871, c-873, c-875, c-2247, c-2532, c-2542, c-2603, c-2608, c-2611, c-2612, c-2617, c-3103, c-3108, c-3163
`\@@par`, a-2403, a-2941, a-2955, a-3032, a-5211
`\@@settexcodehangi`, a-2162, a-2162, a-2656, a-2712
`\@EOF`, a-7499, a-7502
`\@M`, a-5775, c-1537
`\@MakeShortVerb`, a-7364, e-377, e-378, e-381, e-715
`\@NoEOF`, a-7497, a-7501
`\@aalph`, a-6317, a-6318
`\@aftercodegfalse`, a-2685, a-2944, a-3092
`\@aftercodegtrue`, a-2266, a-2690, a-2719, a-2922, a-5566, a-5600
`\@afterheading`, c-1530
`\@afternarrgfalse`, a-2266, a-2690, a-2922, a-5566, a-5600
`\@afternarrgtrue`, a-2379
`\@badend`, c-598
`\@beginputhook`, *a-2328*, a-2434, a-2435
`\@begnamedgroup`, c-537, c-553, c-561, c-566
`\@begnamedgroup@ifcs`, c-562, c-565
`\@car`, c-1878
`\@cclv`, c-2356
`\@cclvi`, c-2341
`\@charlb`, a-6303
`\@charrb`, a-6305
`\@checkend`, c-598
`\@clsextension`, c-2466
`\@clubpenalty`, a-2310, c-1542
`@codeskipput`, *40*
`\@codeskipputgfalse`, a-2379, a-2667, a-2923, a-5566, a-5601, a-6822
`\@codeskipputgtrue`, a-2254, a-2260, a-2266, a-2669, a-2685, a-2944, a-3032, a-3082, a-4891, a-4902, a-5034, a-5041
`\@codetonarrskip`, a-2330, a-2550, a-2571, a-2901, a-2919, a-2954, a-2975, a-3074, a-3117
`\@countalllinestrue`, a-1972, a-1976
`\@ctrerr`, a-6324
`\@currenvir`, a-4960, a-4983, a-4984, a-6835, a-6839, a-6986, c-540, c-592, e-508, e-509, g-63
`\@currenvir*`, a-4953
`\@currenvline`, c-544
`\@currsize`, c-640, c-641, c-642, c-643, c-644, c-645, c-646, c-647, c-648, c-649
`\@debugtrue`, b-186
`\@def@breakbslash`, e-602, e-608
`\@defentryze`, a-3570, a-4066, a-4456, a-4463, a-4467, a-4772
`\@docinclude`, a-6204, a-6210
`\@dsdirgfalse`, a-2678, a-2693, a-2755, a-2813, a-2892, a-2906, a-2910, a-5019
`\@dsdirgtrue`, a-2388, a-2662
`\@emptify`, a-2477, a-2628, a-4361, a-4484, a-4581, a-4664, a-4670, a-4671, a-5420, a-5749, a-6313, a-6446, a-6596, a-6858, a-6864, a-6897, a-6899, a-6906, a-6908, a-6983, a-6989, a-7311, a-7312, a-7316, c-301, c-302, c-306, e-359
`\@endinputhook`, *a-2356*, a-2430, a-2431
`\@enumctr`, a-6995, a-6996, c-1793, c-1794, c-1800
`\@enumdepth`, c-1789, c-1792, c-1793
`\@fileswfalse`, a-6798
`\@fileswithoptions`, c-2466
`\@firstofmany`, a-4566, a-4652, a-5711, a-6708, c-2247

\@fshdafalse, a-6644
\@fshdatrue, a-6642
\@gif, c-184, c-185, c-192
\@glossaryfile, a-4718
\@gmccnochangestrue, b-198
\@if, c-207, c-208, c-211
\@ifEOLactive, a-2459, a-2464, a-3148, a-3182, a-3318
\@ifQueerEOL, a-2446, a-2463, a-5476, a-5946
\@ifXeTeX, b-241, b-304, c-2011, c-2019, c-2131, c-2166, c-2456, c-2494, c-2628, c-2829, f-191
\@ifempty, c-1475, c-1492, c-2516
\@ifenvir, c-590, c-598
\@ifl@aded, c-1203
\@ifncat, c-370, c-373, c-388
\@ifnextac, c-436, e-675
\@ifnextcat, a-3422, a-3450, c-363, c-437
\@ifnextif, c-399
\@ifnextspace, c-459, c-2723, c-2762
\@ifnif, c-406, c-409
\@ifnotmw, b-425, c-1358, c-1361, c-1525, c-1630, c-1717, c-1772
\@ifstarl, a-3656, a-3665, a-4448, a-4489, a-4506, a-4553, a-4588, a-4605, a-4684, a-4688, a-4800, a-4823, a-7072
\@include, c-2539, c-2541
\@indexallmacrostrue, a-1999
\@itemdepth, c-1807, c-1810, c-1811
\@itemitem, c-1811, c-1812
\@latexerr, a-6203
\@linesnotnumtrue, a-1956
\@ltxDocIncludetrue, a-6437
\@makefntext, a-6495
\@marginparsusedfalse, a-2026
\@marginparsusedtrue, a-2016, a-2019, a-2021, a-2024
\@minus, c-1640, c-1645, c-1651, c-1653, c-1658, c-1660, c-1667, c-1672
\@mparswitchtrue, f-390
\@newlinegfalse, a-2569, a-2694, a-2844, a-2862, a-2872
\@newlinegtrue, a-2386, a-2661
\@nobreakfalse, c-1536, c-2294
\@nobreaktrue, c-1531
\@noindextrue, a-1985
\@normalcr, c-3201
\@nx, a-2333, c-170, c-342, c-377, c-413, c-440, c-1008, c-1013, c-1047, c-1084, c-1085, c-1169, c-1170, c-1177, c-1178, c-1197, c-1212, c-1214, c-1215, c-1216, c-1610, c-1611, c-1845, c-1845, c-1992, c-2423, c-2840, c-2847, c-2957, c-2958, e-453, e-457, e-742
\@oarg, c-950, c-952, c-953
\@oargsq, c-950, c-953, c-968
\@oldmacrocode, a-4954, a-4979
\@oldmacrocode@launch, a-4931, a-4933, a-4936
\@onlypreamble, a-6440, a-7167, a-7180, a-7184, c-2296, f-187, f-188, f-199
\@pageinclindexfalse, a-4316
\@pageinclindextrue, a-4868
\@pageindexfalse, a-7179
\@pageindextrue, a-1990, a-4381, a-7182
\@parg, c-957, c-959, c-960
\@pargp, c-957, c-960, c-969
\@parindent, c-1796, c-1797, c-1799, c-1814, c-1815, c-1817
\@pkgextension, c-1204
\@preamblecmds, c-1198, c-1200, c-1213, c-1217
\@printalllinenosfalse, a-1973
\@printalllinenostrue, a-1977
\@relaxen, a-2241, a-2986, a-3013, a-5360, a-5748, a-5858, a-6268, a-6329, a-6567, a-6568, a-6569, a-7229, a-7498, c-310, c-311, c-315
\@reversemargintrue, f-391
\@shortvrbdef, e-377, e-378, e-383, e-396, e-695, e-706
\@shortvrbinfo, e-383, e-402, e-408, e-410, e-429, e-695, e-710
\@starttoc, c-2289
\@sverb@chbsl, a-6892, e-595, e-599
\@tempdima, c-2635, c-2639, c-2640, c-2642, c-2643, c-2647
\@tempdima*, c-2640, c-2642
\@tempdimb, c-2636, c-2638, c-2639
\@textsuperscript, c-2173, c-2174
\@toodeep, c-1790, c-1808
\@topnewpage, c-1436
\@topsep, e-536, e-537, e-540
\@topsepadd, e-493, e-530, e-532, e-536
\@trimandstore, a-2389, a-2549, a-3104, a-3104, a-3112, a-3115
\@trimandstore@hash, a-3105, a-3106
\@trimandstore@ne, a-3112, a-3115
\@twocolumntrue, f-388
\@twosidetrue, f-389
\@undefined, c-152, f-344, f-357
\@uresetlinecounttrue, a-1963
\@usgentryze, a-4480, a-4494, a-4500, a-4557, a-4561, a-4779, a-4813, a-7080, a-7087
\@whilenum, c-2341, c-2356
\@xa, a-2135, c-169, c-193, c-195, c-212, c-213, c-283, c-293, c-298, c-329, c-330, c-388, c-592, c-593, c-656, c-816, c-939, c-1001,

c-1008, c-1013, c-1078, c-1084, c-1085, c-1138, c-1158, c-1169, c-1170, c-1177, c-1178, c-1508, c-1556, c-1608, c-1739, c-1751, c-1794, c-1812, c-1840, c-1986, c-1992, c-2001, c-2291, c-2384, c-2600, c-2665, c-2668, c-2840, c-2847, c-2957, c-2957, e-263, e-382, e-387, e-392, e-394, e-395, e-396, e-407, e-413, e-418, e-431, e-432, e-437, e-439, e-694, e-699, e-704, e-705, e-743, f-185, f-397, g-38

\@xifncat, c-375, c-388

\@xifnif, c-411

\@zf@euenctrue, b-338

^^A, *7*, a-3173

^^B, *7*, a-3139

\^^M, *7*, a-3260

^^M, a-2323, a-2660

\aalph, a-6317, a-6364

\abovedisplayskip, a-2211

\acro, c-2405, c-2440, c-2441, c-2445

\acrocore, c-2425, c-2435

\activespace, e-329

\actualchar, *20*, a-3342, a-3489, a-4324, a-5673, a-5731, a-5736, a-6163, a-7255

\adashes, c-2789, c-2789, c-2791, c-2797

\add@special, e-384, e-435, e-696

\addfontfeature, c-2048, c-2078, c-2080, c-2103, c-2168, c-2443, c-2654, c-2988, c-3001

\addto@estoindex, a-4462, a-4499, a-4513, a-4771, a-4778, a-4788

\addto@estomarginpar, a-4627, a-4770, a-4777, a-4782

\addto@macro, c-282, c-289

\addtoheading, c-1506

\addtomacro, a-4785, a-4790, a-4931, a-4932, a-5128, c-289, c-2972

\addtonummacro, c-1291

\AddtoPrivateOthers, *19*, a-3300, e-385, e-582, e-697

\addtotoks, c-297

\AE, c-2622

\ae, b-354

\afterfi, *a-1877*, a-2460, a-2464, a-2756, a-2759, a-2846, a-2848, a-3112, a-3297, a-3422, a-3436, a-3461, a-3464, a-3496, a-3497, a-3979, a-3983, a-3987, a-4544, a-5023, a-5093, a-5096, a-6836, a-6837, a-6840, a-6841, a-6970, b-302, b-376, c-440, c-463, c-464, c-497, c-566, c-567, c-594, c-595, c-861, c-895, c-966, c-1000, c-1077, c-2014, c-2414, c-2430, c-2518, c-2519, c-2580, c-2608, c-2722, c-2761, e-396

\afterfifi, a-2780, a-5020, a-5021, a-5182, a-5192, c-499, c-1055, c-1060, c-2013, c-2808

\afterfififi, c-506

\afteriffifi, a-2775, c-500

\afteriffififi, c-505

\afterififfififi, c-504

\AfterMacrocode, *23*, a-6973

\agrave, b-326, b-344

\ahyphen, c-2811

\AKA, c-2441

\all@other, c-1986, c-2095, c-2097

\alpha, c-2852

\AlsoImplementation, *20*, a-7199, a-7213

\AltMacroFont, a-7316

\AmSTeX, 21, c-1915

\and, a-6547, a-6582

\arg, c-965, c-966

article, b-172

\AtBeginDocument, a-2038, a-2046, a-2095, a-2234, a-3488, a-4382, a-4730, a-5932, a-7166, b-240, b-307, b-353, c-781, c-964, c-1211, c-1329, c-2034, c-2045, c-2198, c-2390, c-2718, c-2789, c-2791, e-732, e-733, f-185, f-397

\AtBegInput, *9*, a-2434, a-2444, a-2478, a-3148, a-3182, a-3294, a-3315, a-5757, a-6472, a-6490, a-6795

\AtBegInputOnce, *9*, a-2480, a-7363

\AtDIPrologue, *19*, a-5421

\AtEndDocument, a-6053

\AtEndInput, *9*, a-2430, a-6020, a-7132, a-7157

\AtEndOfPackage, a-2037, a-2044

\author, *a-1834*, a-6541

\AVerySpecialMacro, a-7402

\begin, c-561

\begin*, c-561

\belowdisplayshortskip, a-2218, a-2220, a-2221

\belowdisplayskip, a-2217

\beth, b-336

\bgcolor, c-2992

\BibTeX, 21, c-1918

\Biggl, c-2943

\biggl, c-2941

\Biggr, c-2944

\biggr, c-2942

\Bigl, c-2939

\bigl, c-2937

\Bigr, c-2940

\bigr, c-2938

\bigskipamount, c-2236, c-3152

\bnamegroup, c-553
\boldmath, c-1878
\box, c-1899, c-2936
\breakablevisspace, a-2513, a-2750, a-6880, e-324, e-330
\breakbslash, a-6882, e-293, e-301, e-313, e-602
\breaklbrace, a-6884, e-272, e-280, e-307
\bslash, a-2642, a-3489, a-3608, a-3942, a-4142, a-4145, a-4146, a-4149, a-4151, a-4162, a-4184, a-4185, a-4186, a-4187, a-4194, a-4325, a-4368, a-4566, a-4580, a-4652, a-4663, a-5665, a-5700, a-5701, a-5711, a-5731, a-5733, a-6883, a-6930, a-6986, a-7031, a-7141, c-760, c-914, c-1008, c-1085, c-1109, c-1499, c-1500, c-1501, c-2758, c-2759, c-2768, c-2769, e-292, e-301, e-608, e-647, g-39
\bullet, c-2829

\c@ChangesStartDate, a-5765, a-5770, a-5791, a-5793, a-5794, a-5796
\c@CheckSum, a-5990, a-6029, a-6034, a-6046, a-6066, a-6071
\c@codelinenum, a-2989, a-2993, a-4706, a-4720, a-6987
\c@DocInputsCount, a-2992
\c@footnote, a-6545, a-6592
\c@GlossaryColumns, a-5872, a-5872, a-5880
\c@gm@PronounGender, c-1257
\c@Gm@tempcnt, f-181
\c@gmd@mc, a-6964, a-6969, a-6970, a-6986
\c@GMhlabel, d-149
\c@IndexColumns, a-5438, a-5438, a-5440, a-5470
\c@NoNumSecs, c-1331
\c@secnumdepth, c-1385
\c@StandardModuleDepth, a-7301
\cacute, b-327, b-345
\catactive, *21*, a-6811
\catletter, *21*, a-6813
\catother, *21*, a-6808
\CDAnd, *22*, a-6942
\CDPerc, *22*, a-6944
\changes, a-5658, a-5664, a-5669
\changes@, a-5662, a-5680
\ChangesGeneral, a-5751, a-5757
\ChangesStart, 17, a-5788
ChangesStartDate, *17*
\Character@Table, a-7023, a-7029
\CharacterTable, a-7021
\chardef, c-2829
\check@bslash, e-599, e-608
\check@checksum, a-6020, a-6023
\check@percent, a-3294, e-560, e-575, e-634
\check@sum, a-5986, a-5988, a-6024, a-6034, a-6045, a-6056
\CheckModules, a-7312
CheckSum, a-5990
\CheckSum, *17*, a-5988, a-6077
ChneOelze, a-4175
\chschange, a-6063, a-6067, a-6074
\chunkskip, *18*, *22*, a-2251
\cite, c-1228
class, b-155
\ClassError, c-1498
\cleardoublepage, c-1345
\clubpenalty, *a-2310*, a-2426, c-1537, c-1542
\cmd, c-939
\cmd@to@cs, c-939, c-941
\Code@CommonIndex, a-4506, a-4509
\Code@CommonIndexStar, a-4506, a-4512
\Code@DefEnvir, a-4684, a-4768
\Code@DefIndex, a-4448, a-4453, a-4692, a-5103
\Code@DefIndexStar, a-4448, a-4460, a-5107
\Code@DefMacro, a-4684, a-4691
\Code@Delim, a-2131, a-2133, a-2138
\code@delim, a-2135, *a-2318*, *a-2340*, a-2345, a-2410, a-2419, a-2777, a-2800, a-2881, a-3297, a-4940, a-6762, a-6766, a-6767
\Code@Delim@St, a-2131, a-2138
\code@escape@char, a-2812, a-3360
\Code@MarginizeEnvir, a-4624, a-4627
\Code@MarginizeMacro, a-3569, a-4614, a-4617, a-4693, a-4698
\Code@UsgEnvir, a-4688, a-4775
\Code@UsgIndex, a-4489, a-4492, a-4697, a-4806
\Code@UsgIndexStar, a-4489, a-4497
\Code@UsgMacro, a-4688, a-4696
\CodeCommonIndex, a-4504, a-7249
\CodeCommonIndex*, *15*
\CodeDelim, *18*, a-2131, a-2340, a-4941, a-6763, a-6942
\CodeDelim*, a-2147, a-6944
\CodeEscapeChar, *18*, a-3357, *a-3367*, a-4893, a-4904, a-7118
\CodeIndent, *18*, a-2172, a-2175, a-2682, a-2969, a-3290, a-7112, b-307, *99*
\codeline@glossary, a-4710, a-4737
\codeline@wrindex, a-4703, a-4736, a-4745, a-4750
\CodelineIndex, a-7179, a-7180

File Key: a=gmdoc.sty, b=gmdocc.cls, c=gmutils.sty, d=gmiflink.sty, e=gmverb.sty, f=gmeometric.sty, g=gmoldcomm.sty

codelinenum, *19*, a-2989, a-2993
\CodelineNumbered, a-7166, a-7167, *99*
\CodeMarginize, *14*, a-4603
\CodeSpacesBlank, *11*, a-2038, a-2520
codespacesblank, *11*, a-2036
\CodeSpacesGrey, *11*, a-2046, a-2532
codespacesgrey, *11*, a-2041
\CodeSpacesSmall, a-2525
\CodeSpacesVisible, a-2511, a-2535, a-2542
\CodeTopsep, *18*, a-2190, a-2209, a-2253, a-2259, a-3032, a-4891, a-4893, a-4902, a-4904, a-5033, a-7114
\CodeUsage, *14*, a-4686
\CodeUsgIndex, *15*, a-4487
\color, c-2369
\columnsep, a-5452, f-383
\CommonEntryCmd, *19*, a-4307, a-4430
\continue@macroscan, a-3444, a-3464
\continuum, c-2400
\copy, c-1861, c-1891, c-1905, c-2991, c-3003
copyrnote, *22*, a-6819
\count, a-5776, a-5780, a-5781, c-1292, c-1293, c-1294, c-1866, c-1867, c-1868, c-1869, c-1870, c-1871, c-1872, c-1873, c-2136, c-2137, c-2138, c-2139, c-2143, c-2144, c-2145, c-2146, c-2148, c-2149, c-2150, c-2339, c-2341, c-2342, c-2343, c-2346, c-2355, c-2356, c-2357, c-2358
countalllines, *10*, a-1971
countalllines*, *10*, a-1975
\cs, *20*, a-6907, a-6910, a-6996, c-914, c-920, c-924, c-939
\cup, c-3000
\currentfile, a-6184, a-6185, a-6186, a-6187, a-6189, a-6191, a-6193, a-6195, a-6201, a-6240, a-6247, a-6272, a-6384, a-6385, a-6387, a-6446
\czas, c-2815
\czer, c-2370, c-2373, c-2374
\czerwo, c-2369, c-2370

\dag, c-2374
\daleth, b-336
\data, c-3146
\date, *a-1835*, a-6542
\date@biway, c-3150, c-3157
\date@line, c-3150, c-3160, c-3161
\datef, c-3021, c-3061, c-3147, c-3158, c-3163
\datefsl, c-3039, c-3080, c-3147, c-3158
\dateskip, c-3149, c-3155
\day, a-6065, a-6069
\deadcycles, c-2572
debug, b-186, *106*
\debug@special, a-2827
\Declare@Dfng, a-3666, a-3667, a-3672
\Declare@Dfng@inner, a-3674, a-3677, a-3684
\DeclareBoolOption, a-4185, a-4196
\DeclareComplementaryOption, a-4186, a-4197
\DeclareDefining, *12*, a-3663, a-4085, a-4117, a-4118, a-4119, a-4120, a-4121, a-4122, a-4123, a-4124, a-4125, a-4127, a-4128, a-4129, a-4130, a-4131, a-4132, a-4145, a-4146, a-4149, a-4151, a-4184, a-4185, a-4186, a-4187
\DeclareDefining*, a-4134, a-4135, a-4136, a-4142
\DeclareDOXHead, *13*, a-4160
\DeclareKVOFam, *13*, a-4191
\DeclareLogo, c-1834, c-1851, c-1877, c-1888, c-1918, c-1924, c-1927, c-1929, c-1932, c-1935, c-1939, c-1941, c-1944, c-1951
\DeclareOption, a-1956, a-1963, a-1971, a-1975, a-1985, a-1990, a-1999, a-2024, a-2026, a-2036, a-2041, a-4136
\DeclareOptionX, a-4151, a-4163, a-4170, a-4175, b-142
\DeclareOptionX*, b-264
\DeclareRobustCommand, a-4130, c-2616
\DeclareRobustCommand*, a-7284, c-636, c-673, c-674, c-675, c-676, c-677, c-678, c-807, c-881, c-888, c-903, c-914, c-920, c-924, c-1335, c-1848, c-2047, c-2172, c-2192, c-2200, c-2405, c-2443, c-2719, c-2734, c-2747, c-2756, c-2766, c-3169, c-3172, c-3228, c-3231, d-160, d-177, d-186
\DeclareStringOption, a-4184, a-4195
\DeclareTextCommand, a-4131, c-1845
\DeclareTextCommandDefault, a-4132, c-1847
\DeclareVoidOption, a-4187, a-4198
\defaultfontfeatures, b-242
\DefaultIndexExclusions, *16*, a-5210, a-5346, a-5374
\DefEntry, *19*, a-4427, a-7291
\DefIndex, *15*, a-4446, a-7241
\Define, *14*, a-4678
\define@boolkey, a-3725, a-4146
\define@choicekey, a-3782, a-4149

File Key: a=gmdoc.sty, b=gmdocc.cls, c=gmutils.sty, d=gmiflink.sty, e=gmverb.sty, f=gmeometric.sty, g=gmoldcomm.sty

\define@key, a-3734, a-3749, a-3770, a-4145
\definecolor, a-2062
\defobeylines, c-2224
\dekbigskip, c-2236
\dekclubs, *11*, *b-442*, e-663, *163*
\dekclubs*, *163*
\dekfracc, c-2093, c-2102, c-2492
\dekfraccsimple, c-2491, c-2497
\dekfraccslash, c-2486, c-2494, c-2495
\dekmedskip, c-2235
\deksmallskip, c-2233
\DeleteShortVerb, *11*, e-406, *163*
\Delta, c-2853
\Describe, *14*, a-7069
\Describe@Env, a-7058, a-7064, a-7072, a-7084
\Describe@Macro, a-7058, a-7072, a-7077
\DescribeEnv, a-7063, *98*
\DescribeMacro, a-7056, *98*
\dimen, c-2135, c-2138, c-2142, c-2145, c-2304, c-2305, c-2306, c-2686, c-2687
\dimexpr, c-2877, c-2882, c-2888, c-2935, c-2992
\DisableCrossrefs, a-7188, a-7191
\discre, a-6931, c-854, c-864, c-883
\discret, c-856, c-861
\divide, c-1868, c-1871, c-1872, c-2137, c-2139, c-2144, c-2146, c-2638, c-2639, c-2644
\division, *21*, a-6398, a-6952
\Do@Index, a-5353, a-5360
\do@noligs, e-347, e-653
\do@properindex, a-4528, a-4583, a-4866
\dobreakblankspace, e-332
\dobreakbslash, e-313, e-349
\dobreaklbrace, e-278, e-349
\dobreakspace, e-350, e-366
\dobreakvisiblespace, e-330, e-366
\Doc@Include, a-6147, a-6150
\Doc@Input, a-2294, a-2298, a-7369
\DocInclude, *8*, 10, *23*, *a-1855*, *a-1862*, *a-1863*, *a-1864*, *a-1865*, *a-1866*, a-6147, a-6197, a-6203, a-6419
\DocInput, *8*, a-2294, a-6444, a-6471, a-6767
DocInputsCount, a-2992
\docstrips@percent, a-4946
\DocstyleParms, a-7306
\documentclass, *a-1825*
\DoIndex, *16*, *a-1850*, a-5353, a-5372
\DoNot@Index, a-5159, a-5167
\DoNotIndex, *15*, a-5159, a-5370, a-5372, a-5376
\dont@index, a-5171, a-5174, a-5182, a-5192, a-5360
\DontCheckModules, a-7311
\doprivateothers, a-3301, a-3302, a-3379, a-3383
\dp, c-2920, c-2932, c-2935, c-2992
\ds, 22, a-6935
\dywiz, c-2808

\eacute, b-328, b-347
\edef@other, c-2000, c-2005
\edverbs, b-446, e-671, e-676
\eequals, c-2318
\eg@MakeShortVerb, e-716, e-720
\eg@MakeShortVerbStar, e-716, e-719
\egCode@MarginizeEnvir, a-4606, a-4623
\egCode@MarginizeMacro, a-4607, a-4613
\egRestore@Macro, c-1070, c-1072
\egRestore@MacroSt, c-1070, c-1073
\egroupfirstofone, c-694
\egStore@Macro, c-990, c-995
\egStore@MacroSt, c-990, c-996
\egText@Marginize, a-4823, a-4826
\emdash, c-2774
\emptify, a-2439, a-2613, a-2632, a-3847, a-4241, a-4937, c-302, c-302, c-2050, c-2967, c-2971
\EnableCrossrefs, a-6308, a-7190
\enamegroup, c-555
\encapchar, *20*, a-3344, a-3489, a-4329, a-4719, a-5674
\endenumerate, a-6997
\endenvironment, a-5144
\endlinechar, a-6726
\endlist, c-1803, c-1820
\endmacro, a-5041, a-5056
\endmacro*, a-5056
\endmacrocode, a-4924
\endoldmc, a-4924
\endskiplines, *24*
\endtheglossary, a-6310
\endverbatim, e-487
\englishdate, c-3060
\enoughpage, c-2302
\enspace, b-428
\ensuremath, b-456, c-815, c-828, c-1936, c-2175
\EntryPrefix, *19*, a-4320, a-4322, a-4361, a-4713, a-4715, a-5466, a-6158
enumargs, a-6993
\enumerate, a-6994
enumerate*, c-1788
\env, *21*, c-920
\environment, a-5143
environment, *15*, a-5143
\envirs@toindex, a-4484, a-4640, a-4643, a-4644, a-4671, a-4790
\envirs@tomarginpar, a-4634, a-4637, a-4638, a-4670, a-4785
\EOF, a-7502
\EOFMark, *18*, *a-2338*, a-2498, a-6766, a-7498, b-456, *106*
\equals, c-2316
\errorcontextlines, b-373
\eTeX, 21, c-1935, c-1939
\evensidemargin, a-6115, f-376
\everyeof, *18*, a-2352

File Key: a=gmdoc.sty, b=gmdocc.cls, c=gmutils.sty, d=gmiflink.sty, e=gmverb.sty, f=gmeometric.sty, g=gmoldcomm.sty

\everypar, a-2330, *a-2330*, a-2389, a-2549, a-2550, a-2570, a-2656, a-2901, a-2919, a-2953, a-2974, a-3112, a-3120, a-5129, a-6820, c-1533, c-1543, c-2708, e-561
\ExecuteOptionsX, b-143
\exhyphenpenalty, b-450, e-474, e-478
\exists, c-2889

\f@encoding, c-2353
\f@series, c-1878
\fakern, c-2961
\fakesc@extrascale, c-2642, c-2657
\fakescaps, c-2616
\fakescapscore, c-2594, c-2619
\fakescextrascale, c-2657
\file, *21*, c-888
\filedate, *22*, a-6389, a-6655, a-6744
\filediv, a-6333, a-6350, a-6397, a-6415, a-6567, a-6626
\filedivname, a-6334, a-6344, a-6347, a-6351, a-6364, a-6366, a-6395, a-6414, a-6568
\FileInfo, *22*, a-6670
\fileinfo, *22*, a-6657
\filekey, a-6272, a-6369, a-6372
\filename, a-6388, a-6653
\filenote, *22*, a-6744, a-6746
\filesep, a-6157, a-6158, a-6313, a-6368
\fileversion, 22, a-6390, a-6656, a-6744
\Finale, *20*, a-7200, a-7229
\finish@macroscan, a-3422, a-3436, a-3450, a-3461, a-3562, g-36, g-37, g-64
\Finv, b-336
\fixbslash, e-301, *162*
\fixcopyright, c-2455
\fixlbrace, e-307, *162*
\fontencoding, c-2447, e-362
\fontseries, b-386
\fontspec, b-399
\fooatletter, c-698
\foone, a-2462, a-2647, a-3138, a-3172, a-3210, a-3259, a-3655, a-3865, a-3954, a-3997, a-4956, a-4970, a-5549, a-5593, a-6682, a-6710, a-6757, a-6806, a-7025, a-7496, c-694, c-698, c-717, c-720, c-728, c-744, c-764, c-768, c-771, c-923, c-2215, c-2223, c-2788, c-2810, c-2903, e-276, e-290, e-310, e-320, e-327, g-51, g-82
\footins, f-384
\footskip, f-380
\forall, c-2886
\FormatHangHeading, c-1639, c-1646, c-1654, c-1661
\FormatRunInHeading, c-1668, c-1673
\freeze@actives, c-2338
\fullcurrentfile, a-6185, a-6201, a-6249

\g@emptify, a-2493, a-3475, a-4638, a-4644, a-5949, a-6508, a-6509, a-6629, a-6789, c-306, c-307
\g@relaxen, a-3616, a-4348, a-4350, a-4359, a-5122, a-6628, c-315, c-316
\gaddtomacro, 20, a-2493, a-3290, a-3697, c-277
\gag@index, a-2095, a-4743, a-7166, a-7188
\Game, b-336
\garamath, c-2987
\gemptify, c-307, c-307
\GeneralName, a-5719, a-5720, a-5749, a-5803, a-6163
\generalname, a-5680, a-5686, a-5736, a-5845
\geometry, b-363
\GetFileInfo, *22*, a-6247, a-6387, a-6652
\gimel, b-336
\glet, a-2359, a-2712, a-3474, a-4468, a-4618, a-4940, a-5550, a-5594, a-6374, a-6379, a-6393, c-269, c-1557
\glossary@prologue, a-5801, a-5881, a-5919, a-5926, a-6376
\glossaryentry, a-4719
\GlossaryMin, *16*, a-5870, a-5870, a-5881
\GlossaryParms, *16*, a-5882, a-5933
\GlossaryPrologue, *16*, a-5918
\glueexpr, c-3178
\gluestretch, c-3179
\gm@atppron, c-1259, c-1263, c-1264, c-1265, c-1266, c-1268, c-1269, c-1270, c-1271
\Gm@bindingoffset, f-181, f-363
\Gm@bmargin, f-351
\Gm@checkbool, f-345, f-362, f-365, f-366, f-367
\gm@clearpagesduetoopenright, c-1344, c-1404
\Gm@cnth, f-181, f-350, f-353
\Gm@cntv, f-181, f-352, f-355
\Gm@dimlist, f-182
\gm@dontnumbersectionsoutofmainma
c-1342, c-1388
\gm@DOX, b-142, b-155, b-162, b-165, b-169, b-172, b-180, b-186, b-191, b-198, b-202, b-223, b-232, b-250, b-251
\Gm@driver, f-368
\gm@duppa, c-2094, c-2095, c-2097
\gm@EOX, b-143, b-257, b-260
\Gm@even@mp, f-182
\gm@gobmacro, c-1986, c-1992
\Gm@height, f-351
\Gm@hmarginratio, f-354
\gm@hyperrefstepcounter, c-1334, c-1337, c-1417

File Key: a=gmdoc.sty, b=gmdocc.cls, c=gmutils.sty, d=gmiflink.sty, e=gmverb.sty, f=gmeometric.sty, g=gmoldcomm.sty

\gm@hypertarget, d-161, d-164
\gm@iflink, d-186, d-187, d-189
\gm@ifnac, c-437, c-439
\gm@ifref, d-177, d-178, d-180
\gm@lbracehook, a-3976, e-280, e-285
\gm@letspace, c-455, c-463
\Gm@lines, f-357
\Gm@lmargin, f-349
\gm@notprerr, c-1209, c-1216
\Gm@odd@mp, f-182
\Gm@orgh, f-182
\Gm@orgw, f-182
\Gm@paper, f-344
gm@PronounGender, c-1257
\gm@pswords, c-871, c-873, c-875
\Gm@rmargin, f-349
\gm@sec, c-1755, c-1766, c-1767
\gm@secini, c-1720, c-1739, c-1745, c-1751, c-1764
\gm@secmarkh, c-1747
\gm@secstar, c-1723, c-1733, c-1740, c-1752, c-1766, c-1767
\gm@secx, c-1755, c-1756
\gm@secxx, c-1722, c-1750, c-1757
\Gm@showparams, f-336
\gm@straightensec, c-1761, c-1770
\gm@targetheading, c-1335, c-1338
\Gm@tmargin, f-351
\Gm@true, f-346, f-347, f-348, f-364
\Gm@truedimen, f-364
\gm@verb@eol, a-3315, a-6889, e-594, e-619, e-641
\Gm@vmarginratio, f-356
\Gm@wd@mp, f-181
\Gm@width, f-349
\gm@xistar, a-4957, a-4960
\gma, c-2980
\gma@arrowdash, c-2990, c-2999, c-3005, c-3006
\gma@bare, c-2976, c-2978
\gma@bracket, c-2978, c-2979
\gma@checkbracket, c-2977, c-2981
\gma@dollar, c-2975, c-2976, c-2981
\gma@gmathhook, c-2964, c-2971, c-2972, c-2994
\gma@quantifierhook, c-2886, c-2889, c-2967, c-2969, c-3001, c-3007
\gma@tempa, c-2875, c-2877, c-2880, c-2882, c-2931, c-2935, c-2956, c-2959
\gma@tempb, c-2932, c-2935
\gmath, c-2839, c-2975, c-2979
\gmathhook, c-2972
\gmboxedspace, a-6865, a-6868, a-6872, a-6881, a-6883, a-6887, a-6900, a-6909, a-6931
\gmcc@article, b-172
\gmcc@CLASS, b-157, b-159, b-281, b-289
\gmcc@class, b-155, b-162, b-165, b-169, b-172
\gmcc@debug, b-186
\gmcc@gmeometric, b-202
\gmcc@minion, b-250
\gmcc@mptt, b-232
\gmcc@mwart, b-162
\gmcc@mwbk, b-169
\gmcc@mwclsfalse, b-281
\gmcc@mwclstrue, b-159
\gmcc@mwrep, b-165
\gmcc@nochanges, b-198
\gmcc@noindex, b-191
\gmcc@oldfontsfalse, b-223, b-238
\gmcc@oldfontstrue, b-222, b-304
\gmcc@outeroff, b-180
\gmcc@PAGELLA, b-252
\gmcc@pagella, b-251
\gmcc@resa, b-158, b-159
\gmcc@setfont, b-237, b-250, b-251
\gmcc@sysfonts, b-223
\gmd@@toc, a-2444, a-2446, a-2447
\gmd@ABIOnce, a-2477, a-2478, a-2493
\gmd@adef@altindex, a-4042, a-4052, a-4053, a-4055, a-4056, a-4059, a-4061
\gmd@adef@checkDOXopts, a-3894, a-3909
\gmd@adef@checklbracket, a-3879, a-3900
\gmd@adef@cs, a-3855
\gmd@adef@cshookfalse, a-3572
\gmd@adef@cshooktrue, a-3855
\gmd@adef@currdef, a-3689, a-3694, a-3698, a-3702, a-3703, a-3705, a-3707, a-3710, a-3713, a-3753, a-3772, a-3774, a-3841, a-3921, a-3926, a-4025, a-4031, a-4043, a-4046, a-4054, a-4057
\gmd@adef@defaulttype, a-3666, a-3667, a-3686
\gmd@adef@deftext, a-4018, a-4038
\gmd@adef@dfKVpref, a-3876, a-3889, a-3916, a-3920
\gmd@adef@dk, a-3871
\gmd@adef@dofam, a-3892, a-3950, a-3958, a-4008, a-4024
\gmd@adef@dox, a-3882
\gmd@adef@fam, a-3891, a-3948, a-3951, a-3956, a-3959, a-4006, a-4009, a-4032, a-4033
\gmd@adef@indextext, a-4041, a-4062, a-4065
\gmd@adef@KVfam, a-3770
\gmd@adef@KVpref, a-3749
\gmd@adef@prefix, a-3734
\gmd@adef@scanDKfam, a-3985, a-4005
\gmd@adef@scanDOXfam, a-3884, a-3911, a-3934
\gmd@adef@scanfamact, a-3939, a-3955
\gmd@adef@scanfamoth, a-3936, a-3947

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@scanKVpref, a-3872, a-3883, a-3901, a-3910, a-3915
\gmd@adef@scanname, a-3981, a-3989, a-4013
\gmd@adef@setkeysdefault, a-3691, a-3718
\gmd@adef@setKV, a-3754, a-3778, a-3836, a-3839
\gmd@adef@settype, a-3806, a-3808, a-3810, a-3812, a-3814, a-3816, a-3818, a-3820, a-3822, a-3824, a-3832
\gmd@adef@text, a-3863
\gmd@adef@TYPE, a-3709, a-3833
\gmd@adef@type, a-3782
\gmd@adef@typenr, a-3783, a-3805
\gmd@adef@typevals, a-3783
\gmd@auxext, a-6175, a-6177, a-6217, a-6226
\gmd@bslashEOL, a-3218, a-3261, a-3264
\gmd@charbychar, a-2678, a-2722, a-2795, a-2850, a-3592, a-3855, a-3863, a-3901, a-3911, a-3917, a-3952, a-3960, a-4010, a-4020
\gmd@checkifEOL, a-2551, a-2899
\gmd@checkifEOLmixd, a-2806, a-2917
\gmd@chschangeline, a-6030, a-6038, a-6047, a-6062
\gmd@closingspacewd, a-2663, a-3246, a-3250, a-3252
\gmd@codecheckifds, a-5016
\gmd@codeskip, a-2685, a-2944, a-3032, a-3054, a-3082
\gmd@continuenarration, *a-2421*, a-2546, a-2779
\gmd@countnarrline@, a-2568, a-2608
\gmd@counttheline, a-2818, a-2850, a-2857
\gmd@cpnarrline, a-2548, a-2606, a-2613, a-2628, a-2900, a-2918, a-3295
\gmd@ctallsetup, a-2616, a-2632, a-4939, a-6672
\gmd@currentlabel@before, a-2303, a-2359
\gmd@currenvxistar, a-4953, a-4959
\gmd@DefineChanges, a-5657, a-5857
\gmd@detect@def, a-4094, a-4096
\gmd@detectname@def, a-4095
\gmd@detectors, a-3593, a-3696, a-3697, a-3847, a-4238, a-4241, a-4249, a-4349
\gmd@difilename, a-6171, a-6174
\gmd@dip@hook, a-5413, a-5420, a-5421
\gmd@docincludeaux, a-6183, a-6327, a-6329
\gmd@docstripdirective, a-2893, a-2911, a-5020, a-5552
\gmd@docstripinner, a-5560, a-5562
\gmd@docstripshook, a-5602
\gmd@docstripverb, a-5559, a-5597
\gmd@doindexingtext, a-4069, a-4642, a-4647
\gmd@doIndexRelated, a-6239, a-6256, a-6307
\gmd@dolspaces, *a-2422*, a-2678, a-2753
\gmd@DoTeXCodeSpace, a-2407, a-2512, a-2521, a-2526, a-4942
\gmd@eatlspace, a-2758, a-2770, a-2775
\gmd@endpe, a-2925, a-2929, a-2950, a-2959, a-2966
\gmd@EOLorcharbychar, a-2822, a-2837
\gmd@evpaddonce, a-5112, a-5118
\gmd@fileinfo, a-6679, a-6691
\gmd@finishifstar, a-3422, a-3450, a-3460
\gmd@FIrescan, a-6696, a-6711
\gmd@glossary, a-4733, a-4737, a-5718
\gmd@glossCStest, a-5713, a-5717, a-5733, a-5742
\gmd@gobbleuntilM, a-3176, a-3177
\gmd@grefstep, a-2569, a-2578, a-2624, a-2629, a-2694, a-2862, a-2872
\gmd@guardedinput, a-2332, a-2353
\gmd@iedir, a-5168, a-5187, a-5360
\gmd@ifinmeaning, a-3473, a-3491, a-3693, g-38
\gmd@ifonetoken, a-5039, a-5054, a-5089, a-7058
\gmd@ifsingle, a-5074, a-5092
\gmd@iihook, a-2337, a-2439, a-6764
\gmd@in@@, a-3495, a-3495, a-3499
\gmd@inputname, a-2301, a-3891, a-6027, a-6037, a-6044
\gmd@inverb, a-6863, a-6866, a-6875
\gmd@jobname, a-6170, a-6174
\gmd@justadot, a-4465, a-4468, a-4481, a-4618, a-5168
\gmd@KVprefdefault, a-3741, a-3749, a-3751, a-3876, a-3889, a-4160
\gmd@lbracecase, a-3863, a-3875, a-3888, a-3977, a-3980, a-3984, a-3988, a-3992
\gmd@ldspaceswd, a-2692, a-2702, a-2703, a-2716, a-2742, a-2757, a-2772, a-2778
\gmd@maybequote, a-3413, a-3434, a-3446, a-3474, a-3475, a-4543
gmd@mc, a-6964

File Key: a=gmdoc.sty, b=gmdocc.cls, c=gmutils.sty, d=gmiflink.sty, e=gmverb.sty, f=gmeometric.sty, g=gmoldcomm.sty

\gmd@mcdiag, a-6968, a-6983, a-6985, a-6989

\gmd@mchook, a-6967

\gmd@modulehashone, a-5564, a-5568, a-5599, a-5603

\gmd@narrcheckifds, a-2906, a-2909

\gmd@narrcheckifds@ne, a-2885, a-2891

\gmd@nlperc, a-6870, a-6876, a-6898, a-6901, a-6907, a-6910

\gmd@nocodeskip, a-2683, a-2687, a-2946, a-2948, a-3048, a-3056, a-3076, a-3089

\gmd@oldmcfinis, a-4984

\gmd@oncenum, a-5119, a-5121, a-5123, a-5128, a-5130, a-5133

\gmd@parfixclosingspace, a-2649, a-3245

\gmd@percenthack, a-2803, a-2880

\gmd@preambleABD, e-732, e-733, e-740

\gmd@preverypar, a-2155, a-2571, a-2904, a-2919, a-2954, a-2975, a-3110, a-3118, a-3120

\gmd@providefii, a-6728, a-6733

\gmd@quotationname, a-6827, a-6835, a-6839

\gmd@resa, a-3685, a-3687, a-3735, a-3738, a-3750, a-3751, a-3757, a-3758, a-3760, a-3763, a-3771, a-3773, a-3775, a-3777, a-3840, a-3843, a-4655, a-4658, a-4660

\gmd@resetlinecount, *a-2320*, a-2986, a-2999

\gmd@ResumeDfng, a-4282, a-4284

\gmd@revprefix, a-4394, a-4396

\gmd@setChDate, a-5769, a-5772, a-5791

\gmd@setclosingspacewd, a-3251

\gmd@setclubpenalty, a-2308, a-2398, a-2402, a-2426

\gmd@skipgmltext, a-6788, a-6789, a-6799

\gmd@skiplines, a-2639, a-2642

\gmd@spacewd, a-2739, a-2756, a-2772

\gmd@texcodeEOL, a-2681, a-2839

\gmd@texcodespace, a-2522, a-2528, a-2677, a-2750, a-2754, a-2771

\gmd@textEOL, a-2375, a-2464, a-2905, a-2924, a-3213, a-4937, a-5568, a-5603

\gmd@typesettexcode, a-2648, a-2765, a-2780

\gmd@writeckpt, a-6260, a-6300

\gmd@writeFI, a-6695, a-6704

\gmd@writemauxinpaux, a-6217, a-6278

\gmdindexpagecs, a-4387, a-4393

\gmdindexrefcs, a-4384, a-4387, a-4391

\gmdmarginpar, *15*, *22*, a-4839, a-4845, a-4849

\gmdnoindent, *22*, a-6847

\gmdoccMargins, b-362, b-367

\gmdocIncludes, *9*, a-6470

\gme@tobestored, f-180, f-185, f-397

gmeometric, b-202

gmglo.ist, 81

GMhlabel, d-149

\gmhypertarget, a-3020, d-160, *159*

\gmiflink, a-4391, d-186, *159*

\gmifref, d-177, *159*

\gml@StoreCS, c-1034, c-1057, c-1095

\gml@storemacros, c-1035, c-1046, c-1055, c-1060, c-1098

gmlonely, *21*, a-6784, a-6796

\gmobeyspaces, *a-2521*, c-2216, e-476, e-595

\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-326

\GMtextsuperscript, c-2165

\gmu@acroinner, c-2411, c-2421, c-2422, c-2430

\gmu@acrospaces, c-2405, c-2410, c-2410, c-2414

\gmu@checkaftersec, c-1551, c-1610

\gmu@copyright, c-2452, c-2453

\gmu@dashfalse, c-3109

\gmu@dashtrue, c-3111

\gmu@def, c-1360, c-1362, c-1362, c-1381

\gmu@dekfracc, c-2077, c-2096, c-2098

\gmu@dekfraccsimple, c-2098, c-2484, c-2492

\gmu@denominatorkern, c-2079, c-2108, c-2486

\gmu@discretionaryslash, c-883, c-894

\gmu@dywiz, c-2807, c-2811

\gmu@fileext, c-2534, c-2543, c-2543, c-2561

\gmu@filename, c-2533, c-2546, c-2558, c-2561, c-2564, c-2573

\gmu@getaddvs, c-1602, c-1602, c-1608

\gmu@getext, c-2532, c-2542

\gmu@ifnodash, c-3103, c-3108

\gmu@LastSkip, c-2824, c-2825, c-2827

\gmu@luzniej, c-2691, c-2694, c-2696

\gmu@nl@reserveda, c-1168, c-1171, c-1176, c-1179

\gmu@numeratorkern, c-2078, c-2107, c-2108, c-2485

\gmu@prevsec, c-1532, c-1534, c-1556, c-1563, c-1593

File Key: a=gmdoc.sty, b=gmdocc.cls, c=gmutils.sty, d=gmiflink.sty, e=gmverb.sty, f=gmeometric.sty, g=gmoldcomm.sty

\gmu@printslashes, c-888, c-890, c-890, c-892, c-895
\gmu@resa, a-3924, a-3930, a-4029, a-4035, c-2382, c-2384
\gmu@reserveda, c-452, c-454, c-460, c-462, c-592, c-594, c-1047, c-1050, c-1053, c-1508, c-1510, c-1557, c-1558, c-1561, c-1840, c-1842, c-1844, c-1845, c-1846, c-1991, c-1993, c-2000, c-2001, c-2005, c-2006, c-2517, c-2518
\gmu@reservedb, c-593, c-594
\gmu@RPif, c-2368, c-2380, c-2391, c-2400
\gmu@scalar, c-2629, c-2633, c-2634, c-2640
\gmu@scalematchX, c-2619, c-2631, c-2655
\gmu@scapLetters, c-2590, c-2600, c-2605
\gmu@scapSpaces, c-2603, c-2608, c-2612
\gmu@scapss, c-2611, c-2617
\gmu@scscale, c-2648, c-2654
\gmu@setheading, c-1607, c-1613, c-1614
\gmu@setsetSMglobal, c-1033, c-1038, c-1094
\gmu@setSMglobal, c-1040, c-1042, c-1060
\gmu@SMdo@scope, c-1137, c-1139, c-1142, c-1143, c-1157
\gmu@SMdo@setscope, c-1135, c-1141, c-1155
\gmu@SMglobalfalse, c-1002, c-1015, c-1042, c-1051, c-1079, c-1088, c-1145
\gmu@SMglobaltrue, c-978, c-1040
\gmu@smtempa, c-1006, c-1014, c-1082, c-1087
\gmu@tempa, a-2641, a-2643, a-3704, a-3712, a-4320, a-4322, a-4324, a-4329, a-4330, a-4397, a-4398, a-4569, a-4572, a-4575, a-4713, a-4715, a-4717, a-4719, a-4721, a-4783, a-4785, a-4789, a-4790, a-5075, a-5076, a-5095, a-5096, a-5175, a-5178, a-5180, a-5185, a-5187, a-5718, a-5741, a-5800, a-5806, a-6025, a-6035, a-6042, a-6052, a-6053, a-6222, a-6658, a-6659, a-6787, a-6794, a-6795, a-6857, a-6864, a-6872, a-6896, a-6899, a-6905, a-6908, c-2664, c-2666, c-2667, c-2668
\gmu@tempb, a-5090, a-5093, a-5095, a-5692, a-5701, a-5710, a-5730, a-5732, a-5733, a-5734, a-6221, a-6222, a-6654, a-6659, a-6858, a-6865, a-6872, a-6897, a-6900, a-6906, a-6909
\gmu@testdash, c-3107, c-3147, c-3158
\gmu@tilde, c-2187, c-2192, c-2203
\gmu@whonly, c-2579, c-2580
\gmu@xedekfraccplain, c-2055, c-2101
\gmu@xedekfraccstar, c-2055, c-2069
\gmu@xefraccdef, c-2070, c-2084, c-2085, c-2086, c-2087, c-2088, c-2089, c-2090, c-2091, c-2092
\gmv@dismath, e-672, e-675, e-679
\gmv@disverb, e-675, e-678
\gmv@edismath, e-673, e-681
\gmv@exhyphenpe, e-474, e-478
\gmv@hyphenpe, e-473, e-477
\gmv@packname, e-425, e-426, e-430
\gn@melet, a-4270, a-4271, c-1175
\gobble, c-705, c-706, c-2957
\gobbletwo, c-707
\grefstepcounter, a-2593, c-236, c-252
\grelaxen, a-5742, a-5751, c-316, c-316, c-1534

\hathat, c-924
\headheight, f-378
\HeadingNumber, c-1414, c-1416
\HeadingNumberedfalse, c-1343, c-1385
\HeadingRHeadText, c-1398
\HeadingText, c-1400
\HeadingTOCText, c-1399
\headsep, f-379
\HeShe, c-1268
\heshe, *7*, c-1263
\hfillneg, c-2239
\hgrefstepcounter, a-2624, a-2629, c-251
\Hide@Dfng, a-4266, a-4268
\HideAllDefining, *13*, a-4236
\HideDef, *13*, a-4108
\HideDefining, *13*, a-4108, a-4264
\HimHer, c-1270
\himher, c-1265
\HisHer, c-1269
\hisher, c-1264
\HisHers, c-1271
\hishers, c-1266
\HLPrefix, *19*, a-3021, a-4320, a-4322, a-4400, a-4706, a-4713, a-4715, a-4720, a-5404, *a-6157*
\hoffset, f-385
\hrule, c-1581
\hunskip, c-259
\Hybrid@DefEnvir, a-5039, a-5106
\Hybrid@DefMacro, a-5039, a-5102

File Key: a=gmdoc.sty, b=gmdocc.cls, c=gmutils.sty, d=gmiflink.sty, e=gmverb.sty, f=gmeometric.sty, g=gmoldcomm.sty

hyperindex, 59
\hyperlabel@line, a-2609, a-2698, a-2863, a-2873, a-3015
\hypersetup, a-2075, a-5445
\hyphenpenalty, b-450, c-875, c-2725, c-3185, e-473, e-477

\idiaeres, b-329, b-348
\if*, a-3461, a-4960
\if@aftercode, a-2683, a-2943, a-3064, a-3077
\if@afterindent, c-1538
\if@afternarr, a-2683, a-2942, a-3069, a-3076
\if@codeskipput, a-2253, a-2259, a-2668, a-2685, a-2944, a-3043, a-3075, a-4891, a-4902, a-5033
\if@countalllines, a-1968, a-2558
\if@debug, b-184, b-370, b-376, b-377
\if@dsdir, a-2281, a-5019
\if@filesw, a-4703, a-6217, a-6225, a-6261, c-2291, c-2545, c-2557, c-2565
\if@fshda, a-6585, a-6603, a-6638
\if@gmccnochanges, b-196, b-435
\if@indexallmacros, a-1997, a-5345
\if@linesnotnum, a-1954, a-3013, a-4381
\if@ltxDocInclude, a-6240, a-6246, a-6250, a-6432
\if@mainmatter, c-1343
\if@marginparsused, a-2009, a-4831
\if@mparswitch, f-348, f-390
\if@newline, a-2273, a-2607, a-2694, a-2840, a-2859, a-2870
\if@nobreak, c-1535
\if@noindex, a-1983, a-2094
\if@noskipsec, e-529
\if@openright, c-1345
\if@pageinclindex, a-4319, a-4366, a-4712
\if@pageindex, a-1988, a-3016, a-4316, a-4383, a-4731, a-5397, a-5400, a-5401, a-5403
\if@printalllinenos, a-1969, a-2604, a-4867
\if@RecentChange, a-5683, a-5768
\if@reversemargin, f-391
\if@specialpage, c-1403
\if@twoside, c-1429, f-347, f-348, f-389
\if@uresetlinecount, a-1961, a-2985
\ifcsname, a-6969, c-566
\ifdate, c-3144, c-3147, c-3149
\ifdefined, c-161, c-723, c-782, c-2012, c-2717, c-2787
\ifdtraceoff, b-377
\ifdtraceon, b-376
\iffontchar, c-2071
\ifGm@pass, f-341
\ifgmcc@mwcls, b-152, b-280, b-284, b-302
\ifgmcc@oldfonts, b-221, b-315, b-382
\ifgmd@adef@cshook, a-3565, a-3853
\ifgmd@adef@star, a-3675, a-3725
\ifgmd@glosscs, a-3559
\ifgmu@dash, c-3101, c-3107, c-3113, c-3147, c-3158
\ifgmu@postsec, c-1553, c-1592, c-1600
\ifgmu@SMglobal, c-976, c-1000, c-1007, c-1039, c-1077, c-1083, c-1142
\ifHeadingNumbered, c-1384, c-1412
\ifodd, c-1261
\ifprevhmode, a-2787, a-2881, a-2967
\ifSecondClass, c-2464
\ikern, c-2505
\im@firstpar, a-3586, a-3587, a-3588, a-4525, a-4526, a-4529
\IMO, c-2440
\in, c-3007
\incl@DocInput, a-6249, a-6444, a-6467, a-6471
\incl@filedivtitle, a-6597, a-6626
\incl@titletotoc, a-6585, a-6598
\inclasthook, c-2562, c-2584
\InclMaketitle, a-6243, a-6579
\includegraphics, c-2154
\incs, *21*, a-6903, a-6912
\index@macro, a-3588, a-4300, a-4529, a-4584, a-4665
\index@prologue, a-5388, a-5395, a-5440, a-6370
indexallmacros, *10*, a-1999
IndexColumns, *19*
\indexcontrols, a-3473, a-3488
\indexdiv, a-5391, a-5395, a-5926
\indexentry, a-4705
\IndexInput, *10*, a-6759
\IndexLinksBlack, *20*, a-5408, a-5441, a-5445, a-5882
\IndexMin, *19*, a-5435, a-5435, a-5440
\IndexParms, *20*, a-5442, a-5449, a-5933
\IndexPrefix, *19*, a-4324, a-4372
\IndexPrologue, *19*, a-5388, *101*
\inenv, *21*, a-6912
\infty, c-2864
\inputlineno, a-2585, a-2586, a-2623
\interlinepenalty, e-560
\inverb, *20*, a-6854
\itemindent, c-1796, c-1814
itemize*, c-1806
\iteracro, c-2404, c-2408

\justified, c-3195

\kernel@ifnextchar, a-6728
\kind@fentry, a-4307, a-4309, a-4313, a-4320, a-4322
KVfam, *13*, a-3770
KVpref, *13*, a-3749

File Key: a=gmdoc.sty, b=gmdocc.cls, c=gmutils.sty, d=gmiflink.sty, e=gmverb.sty, f=gmeometric.sty, g=gmoldcomm.sty

\labelsep, c-1798, c-1816
\labelwidth, c-1797, c-1798, c-1815, c-1816
\larger, c-673, c-2876, c-2881, c-2937, c-2938, c-2941, c-2942, c-2943, c-2944, *118*
\largerr, c-677, c-2939, c-2940, *118*
\last@defmark, a-4350, a-4471, a-4476, a-5688, a-5700, a-5701, a-5702, a-5748, a-5751
\LaTeXe, c-1830, c-1877
\LaTeXpar, 22, c-1888
\ldate, c-3161, c-3166
\leftarrow, c-2896, c-3005
\leftline, c-3161
\leftmargin, c-1795, c-1813, e-542, e-543
\leftrightarrow, c-2898
\levelchar, *20*, a-3345, a-3489, a-5674, a-5724, a-5738
\linebreak, c-3211
\linedate, c-3149, c-3160, c-3161
\LineNumFont, *19*, a-2610, a-3008, a-3011, a-7134, *99*
\lineskip, a-6529
linesnotnum, *10*, a-1956
\list, c-1794, c-1812
\listparindent, c-1799, c-1817
\liturgiques, c-2367
\LoadClass, b-288, b-293
\longpauza, c-2777, c-2778
\looseness, c-2697, c-2708
\lpauza, c-2747
\ltxLookSetup, *9*, a-6434, a-6440
\ltxPageLayout, *9*, a-6099, a-6436
luzniej, c-2701
luzniej*, c-2706
\luzniejcore, c-2693, c-2701

\macro, a-5031, a-5054, c-1992
macro, *15*, a-5031
macro*, a-5054
\macro@iname, a-3413, a-3431, a-3434, a-3446, a-3588, a-4529, a-4535, a-4543, a-4584, a-4665
\macro@pname, a-3415, a-3435, a-3447, a-3567, a-3569, a-3570, a-3579, a-3588, a-3590, a-3591, a-3708, a-4039, a-4040, a-4061, a-4066, a-4071, a-4095, a-4519, a-4520, a-4524, a-4529, a-4565, a-4566, a-4569, a-4572, a-4584, g-38, g-39
\macrocode, a-4923, g-67
macrocode, *8*, *23*, a-4901
macrocode*, a-4890
\MacrocodeTopsep, a-7114
\MacroFont, a-7104, *99*
\MacroIndent, a-7112, 99
\MacroTopsep, a-2191, a-2210, a-2252, a-5032, a-5041, *99*
\mag, f-387
\main, a-7291
\MakeGlossaryControls, *17*, a-5661, a-5672
\MakePercentComment, a-7323
\MakePercentIgnore, a-5659, a-7322
\MakePrivateLetters, *14*, *19*, a-2409, a-3371, a-3664, a-4265, a-4281, a-4447, a-4488, a-4505, a-4552, a-4587, a-4604, a-4679, a-4687, a-4799, a-4822, a-4944, a-5035, a-5159, a-5353, a-5660, a-7057, a-7071
\MakePrivateOthers, a-3379, a-4448, a-4489, a-4506, a-4553, a-4588, a-4606, a-4684, a-4688, a-4800, a-4823, a-5035, a-7064, a-7072
\MakeShortVerb, *11*, e-375, e-663, e-720, *163*
\MakeShortVerb*, e-719
\maketitle, *8*, *a-1845*, *a-1850*, a-5325, a-6243, a-6491, *89*
\MakeUppercase, c-2594
\mapsto, c-2998
\marg, c-945, c-969
\marginparpush, a-4833
\marginparsep, f-382
\marginpartt, *14*, a-4849, a-4856, b-386, b-399
\marginparwidth, a-4834, a-6113, f-381
\mark@envir, a-2700, a-2868, a-4633
maszynopis, c-3176
\math@arg, c-965, c-966
\mathbin, c-2865, c-2869, c-2894, c-2895, c-2924, c-2925, c-2954, c-2955, c-3000, c-3007
\mathchoice, c-2873, c-2892, c-2904, c-2946, c-2996
\mathclose, c-2922, c-2923, c-2938, c-2940, c-2942, c-2944, c-2952
\mathfrak, c-2400
\mathindent, b-307
\mathit, c-2840, c-2851
\mathop, c-2873
\mathopen, c-2909, c-2923, c-2937, c-2939, c-2941, c-2943, c-2951
\mathpunct, c-2908
\mathrel, c-2866, c-2870, c-2896, c-2897, c-2898, c-2924, c-2925, c-2926, c-2958, c-2999, c-3005, c-3006
\mathrm, c-2847, c-2853, c-2859, c-2865, c-2866, c-2868, c-2869, c-2870, c-2884
\Mathstrutbox@, c-2920
\maybe@marginpar, a-3590, a-3605
\mcdiagOff, a-6989
\mcdiagOn, a-6985
\medmuskip, c-861

File Key: a=gmdoc.sty, b=gmdocc.cls, c=gmutils.sty, d=gmiflink.sty, e=gmverb.sty, f=gmeometric.sty, g=gmoldcomm.sty

\meta, c-807, c-843, c-945, c-952, c-959, *100*
\meta@font@select, c-818, c-837
minion, b-250
\mkern, c-2961
\mod@math@codes, a-5607, a-5609, a-5611
\Module, a-5565, a-5607
\ModuleVerb, a-5600, a-5609
\month, a-6065, a-6069
mptt, b-232
\mpttversion, b-232
\mskip, c-861
\multiply, a-5775, a-5780, c-1867, c-1870, c-2646, c-2696, c-2707
\mw@getflags, c-1556
\mw@HeadingBreakAfter, c-1405, c-1425, c-1440, c-1444, c-1474, c-1557
\mw@HeadingBreakBefore, c-1402, c-1473, c-1558
\mw@HeadingLevel, c-1382, c-1385
\mw@HeadingRunIn, c-1420, c-1473
\mw@HeadingType, c-1401, c-1532, c-1564, c-1565, c-1578
\mw@HeadingWholeWidth, c-1423, c-1474
\mw@normalheading, c-1427, c-1436, c-1439, c-1443, c-1613
\mw@processflags, c-1475
\mw@runinheading, c-1421, c-1614
\mw@secdef, c-1480, c-1481, c-1482, c-1488
\mw@section, c-1479
\mw@sectionxx, c-1381
\mw@secundef, c-1484, c-1493, c-1496
\mw@setflags, c-1485
mwart, b-162, *106*
mwbk, b-169, *106*
mwrep, *a-1825*, b-165, *106*

\n@melet, a-3709, a-3710, a-4926, a-4927, a-5702, c-1167, c-1507, c-1511, c-1724, c-1730, c-1764, c-2074, e-497
\nacute, b-330, b-349
\nameshow, c-329
\nameshowthe, c-330
napapierki, c-2684
\napapierkicore, c-2681, c-2685
\napapierkistretch, c-2679, c-2682
\nawj, c-2710
\nazwired, c-2834
\neg, c-2865, c-2960
\neq, c-2866, c-2955
\neqb, c-2955
NeuroOncer, *a-5121*
\newbox, a-4122
\newcount, a-4117, a-5438, a-5765, a-5872, a-5986, c-2691, f-192
\newcounter, a-2989, a-2992, a-2993, a-4142, a-5990, a-6964, a-7301, c-1257, c-1331, d-149
\newdimen, a-4118, a-5435, a-5870
\newgif, a-2273, a-2281, a-2787, a-3043, a-3064, a-3069, c-181
\newlength, a-2168, a-2172, a-2178, a-2739, a-2742, a-4125, e-567, e-572
\newlinechar, a-6716
\newread, a-4123
\newskip, a-2190, a-2191, a-3250, a-4119, c-3218
\newtoks, a-2155, a-4121
\newwrite, a-4124, c-2291
\nfss@text, c-816
\nieczer, c-2375
\nlpercent, *21*, a-6895
\nobreakspace, c-2611
nochanges, b-198, *105*
\nocite, c-1227
\noeffect@info, a-7140, a-7158, a-7159, a-7160, a-7161, a-7306, a-7311, a-7312, a-7316
\NoEOF, a-7501
\nohy, c-2509
noindex, *10*, a-1985, b-191, *105*
\nolimits, c-2888, c-2889
nomarginpar, *11*, a-2026
NoNumSecs, c-1331
\NonUniformSkips, *18*, a-2241
\nostanza, *18*, a-2265
\not@onlypreamble, c-1196, c-1200, c-1201, c-1202, c-1203, c-1204
\nu, c-2856
\numexpr, a-2586, c-2816, c-2817
\nummacro, a-5133, c-1280

\oarg, c-950
\obeyspaces, a-2513, a-2527, a-4976, e-327, e-330, e-332, e-632
\ocircum, b-331, b-352
\oddsidemargin, a-6114, f-375
\oe, b-356
\old@MakeShortVerb, a-7364, e-693, e-715
oldcomments, g-28
\olddocIncludes, *9*, *23*, a-6466
\OldDocInput, *8*, *23*, a-6467, a-7361
\oldLaTeX, c-1829
\oldLaTeXe, c-1830
\OldMacrocodes, *23*, a-7366
\OldMakeShortVerb, e-663, e-714, *163*
\oldmc, a-4923, a-4931
oldmc, *23*, a-4923
oldmc*, a-4926
\oldmc@def, a-4981, a-4987
\oldmc@end, a-4982, a-4988
\omega, c-2863
\OnAtLine, c-2312
\OnlyDescription, *20*, a-7220
\oumlaut, b-332, b-350
outeroff, *a-1825*, b-180, *106*

\PackageError, a-3941, a-6197, a-6350, c-1215, c-2757, c-2767
\PackageInfo, a-7132, a-7140, e-430
\PackageWarning, c-664, c-668

File Key: a=gmdoc.sty, b=gmdocc.cls, c=gmutils.sty, d=gmiflink.sty, e=gmverb.sty, f=gmeometric.sty, g=gmoldcomm.sty

\PackageWarningNoLine, a-5664
\pagebreak, c-1428, c-1440, c-1444
\pagegoal, c-2304
\PageIndex, a-7182, a-7184
pageindex, *10*, a-1990
pagella, b-251
\pagestyle, b-429
\pagetotal, c-2305
\paperheight, f-372
\paperwidth, f-371
\par, a-2254, a-2260, a-2354, a-2403, a-2667, a-2764, a-2906, a-2925, a-2937, a-2955, a-3246, a-4891, a-4893, a-4902, a-4904, a-5034, a-5041, a-5253, a-5462, a-5465, a-6491, a-6527, a-6532, a-6536, a-6821, a-6836, a-6840
\paragraph, a-6400, c-3163
\ParanoidPostsec, b-302, c-1589
\parg, c-957
\parsep, e-527
\partial, c-2867
\partopsep, a-2226, c-1795, c-1813, e-532
\PassOptionsToPackage, b-192, b-264, f-193
\pauza, c-2734
\pauzacore, c-2724, c-2725, c-2730, c-2738, c-2743, c-2748, c-2777, c-2780, c-3189, c-3190
\pdfeTeX, 21, c-1939
\pdfoutput, f-192
\pdfTeX, 21, c-1941
\Phi, c-2859
\phi, c-2858
\pi, c-2857
\pk, *21*, *a-1832*, *a-1833*, *a-1861*, a-6163, c-903
\PlainTeX, 21, c-1927
\pm, c-2868, c-2869
\polskadata, c-3020, c-3057
\possfil, c-929
\ppauza, c-2766
\predisplaypenalty, e-475, e-484
prefix, a-3734
\prependtomacro, c-292
\prevhmodegfalse, a-2674, a-2717, a-2797, a-2928, a-2950
\prevhmodegtrue, a-2796
\PrintChanges, *16*, *a-1871*, a-5945, a-5949, a-6309
\PrintDescribeEnv, *99*
\PrintDescribeMacro, *99*
\PrintEnvName, *99*
\PrintFilesAuthors, *8*, a-6642
\PrintIndex, *a-1875*, a-5474, a-6309
\printindex, a-5476, a-5477, a-6309
\printlinenumber, a-2696, a-2866, a-3007, a-3013
\PrintMacroName, *99*
\printspaces, c-871, c-881
\ProcessOptionsX, b-266
\protected, a-2375, a-3140, a-3261, a-3264, c-181, c-193, c-198, c-212, c-236, c-251, c-259
\providecolor, e-739
\ProvideFileInfo, *22*, a-6724, a-6740
\ProvidesClass, b-40
\ProvideSelfInfo, a-6740
\przeniesvskip, c-2823
\ps@plain, a-6517
\ps@titlepage, a-6517
\psi, c-2862

\quad, b-427, b-428, c-2834
\quantifierhook, c-2968, c-2988
\QueerCharOne, a-3173, a-3180, a-3182
\QueerCharTwo, a-3139, a-3146, a-3148
\QueerEOL, *7*, a-2323, a-2446, a-3211, a-4892, a-4903, a-5476, a-5947, a-6472, a-7501, a-7502
quotation, *22*, a-6828
\quote@char, a-3412, a-3433, a-3445, a-3469, a-4542
\quote@charbychar, a-4536, a-4539, a-4544
\quote@mname, a-4520, a-4534, a-4575, a-4660
\quotechar, *20*, a-3343, a-3474, a-3489, a-4325, a-4368, a-4580, a-4663, a-5673, a-5733
\quoted@eschar, a-4325, a-4368, a-4580, a-4581, a-4663, a-4664
\qxcopyright, c-2450
\qxcopyrights, c-2451, c-2456
\qxenc, c-2447, c-2450, c-2453

\raggedbottom, b-417
\rdate, c-3160
\real, c-2640, c-2642
\RecordChanges, *16*, a-5666, a-5857, a-5858, a-6309, b-435, b-438
\reflectbox, c-1947, c-1954
\relaxen, a-4108, b-335, c-311, c-311, c-1470, c-2797, c-2850, e-285, e-676
\relsize, c-636, c-637, c-673, c-674, c-675, c-676, c-677, c-678, *118*
\rem@special, e-411, e-436, e-451
\renewcommand, a-4128, a-5669
\renewcommand*, a-3054, a-3056, b-422, c-2391
\RequirePackage, a-1947, a-1949, a-2057, a-2060, a-2077, a-2089, a-2092, a-2098, a-5426, b-137, b-300, b-318, b-321, b-361, b-372, b-380, b-412, b-455, c-2026, c-2288, c-2384, c-2624, e-243, e-735, f-176
\RequirePackageWithOptions, f-196

File Key: a=gmdoc.sty, b=gmdocc.cls, c=gmutils.sty, d=gmiflink.sty, e=gmverb.sty, f=gmeometric.sty, g=gmoldcomm.sty

\resetlinecountwith, a-2988
\resetMathstrut@, c-2914
\resizebox, c-2153
\resizegraphics, c-2132, c-2152
\Restore@Macro, c-1072, c-1075, c-1095, c-1105
\Restore@Macros, c-1091, c-1093
\Restore@MacroSt, c-1073, c-1081
\RestoreEnvironment, c-1117
\RestoreMacro, a-4096, a-4110, a-4249, a-4250, a-5376, a-6767, c-1069, c-1362, c-2033, c-2035, f-199, g-64
\RestoreMacro*, a-4286, a-4287, c-1119, c-2035
\RestoreMacros, a-4752, c-1091, f-397
\RestoringDo, a-6256, c-1154
\ResumeAllDefining, *13*, a-4247
\ResumeDef, *13*, a-4110
\ResumeDefining, *13*, a-4110, a-4280
\reversemarginpar, a-4832
\rightarrow, c-2897, c-3006
\rightline, b-456, c-3160, c-3207
\romannumeral, c-1793, c-1811
\rotatebox, c-2886, c-2889, c-2894, c-2895
\rs@size@warning, c-655, c-660, c-664
\rs@unknown@warning, c-650, c-668
\runindate, c-3162

\scan@macro, a-2818, a-3396, g-87
\scantokens, a-6716, c-2902
\scshape, c-1925, c-2435, c-3000
\secondclass, c-2463
\SecondClasstrue, c-2465
\sectionsign, c-2844
\SelfInclude, *9*, *a-1853*, a-6419
\SetFileDiv, *21*, a-6339, a-6342, a-6344, a-6352, a-6413, a-6435
\setkeys, a-3686, a-3692, a-3719
\setmainfont, b-243
\setmonofont, b-245
\setsansfont, b-244
\SetSectionFormatting, c-1470, c-1471, c-1632, c-1636, c-1644, c-1652, c-1659, c-1666, c-1671
\settexcodehangi, a-2157, a-2162, a-2704, a-2712, a-2968
\SetTOCIndents, b-427, b-428
\SetTwoheadSkip, c-1616, c-1643, c-1651, c-1658
\sfname, c-881, c-891
\sgtleftxii, a-5550, a-5594
\shortpauza, c-2779
\showboxbreadth, c-326
\showboxdepth, c-326
\ShowFont, c-2352
\showlists, c-326
\showthe, c-330
\sigma, c-2860
\sim, c-2870
\SkipFilesAuthors, *9*, a-6644
\skipgmlonely, *21*, *a-1859*, a-6786
\skiplines, *24*, a-2635
\sl, b-246
\SliTeX, 21, c-1924
\smaller, c-674, c-2485, c-2488, *118*
\smallerr, a-6601, c-678, c-2655, *118*
\smallskipamount, a-2220, a-2221, c-2233, c-2234
\smartunder, *b-447*, c-746
\SMglobal, a-3713, a-4096, a-4238, a-4249, a-4250, a-4286, a-4287, c-978
\SortIndex, a-7255
\special, a-2828, a-2829
\special@index, a-4330, a-4732, a-4736, a-4869
\SpecialEnvIndex, a-7253
\SpecialEscapechar, a-7118
\SpecialIndex, a-7249
\SpecialMainEnvIndex, a-7244
\SpecialMainIndex, a-7241
\SpecialUsageIndex, a-7251
\square, b-456
StandardModuleDepth, a-7301
\stanza, *18*, *22*, *a-1859*, *a-1861*, a-2257, a-6822
\stanzaskip, *18*, a-2178, a-2183, a-2184, a-2209, a-2210, a-2211, a-2217, a-2218, a-2225, a-2258, a-2669, e-558, e-567, e-568
star, *13*, a-3725
\step@checksum, a-3402, a-5993
\stepnummacro, a-5119, c-1282
\StopEventually, *20*, a-7200, a-7220
\Store@Macro, c-995, c-998, c-1034
\Store@Macros, c-1031, c-1032
\Store@MacroSt, c-996, c-1005
\stored@code@delim, a-4940
\Stored@Macro, c-1104, c-1105
\storedcsname, a-6837, a-6841, c-1108
\StoredMacro, c-1104
\StoreEnvironment, a-6826, c-1113
\StoreMacro, a-4106, a-4238, a-5370, a-6762, c-989, c-1362, c-1881, c-2025, c-3189, f-187, g-36
\StoreMacro*, a-3713, c-1115, c-1882
\StoreMacros, a-4750, c-1031, c-2950, f-185
\StoringAndRelaxingDo, a-6239, c-1134
\StraightEOL, *7*, a-2446, a-3198, a-5476, a-5660, a-5947, a-6784, a-6797, a-6820, a-7363

File Key: a=gmdoc.sty, b=gmdocc.cls, c=gmutils.sty, d=gmiflink.sty, e=gmverb.sty, f=gmeometric.sty, g=gmoldcomm.sty

\subdivision, *21*, a-6399, a-6955
\subitem, a-5463
\subs, c-718, c-748
\subsubdivision, *21*, a-6400, a-6958
\subsubitem, a-5464
\sum, c-2885
sysfonts, b-223, *106*

\tableofcontents, *a-1849*, a-2444, a-2445, a-6308
\task, g-90
\TB, 21, c-1933
\TeXbook, 21, c-1932, c-1933
\Text@CommonIndex, a-4588, a-4591
\Text@CommonIndexStar, a-4588, a-4595
\text@indexenvir, a-4562, a-4564, a-4596, a-4815, a-7089
\text@indexmacro, a-4518, a-4558, a-4592, a-4807, a-7081
\Text@Marginize, a-3608, a-4040, a-4636, a-4805, a-4812, a-4827, a-4848, a-5112, a-7079, a-7086
\Text@MarginizeNext, a-5104, a-5109, a-5111
\Text@UsgEnvir, a-4800, a-4810
\Text@UsgIndex, a-4553, a-4556
\Text@UsgIndexStar, a-4553, a-4561
\Text@UsgMacro, a-4800, a-4803
\textbullet, c-2829, c-2829
\textcolor, c-2375, e-742
\TextCommonIndex, *15*, a-4586
\textheight, f-374
\TextIndent, *18*, a-2168, a-2972, a-3084
\textlarger, c-675
\TextMarginize, *14*, a-4821
\textsl, b-246, c-1932
\textsmaller, c-676
\textstyle, c-1879
\textsuperscript, c-2167, c-2172
\texttilde, c-2200
\TextUsage, *14*, a-4797
\TextUsgIndex, *15*, a-4551, a-7251
\textwidth, a-6103, a-6371, c-3208, c-3209, f-373
\thanks, a-6526, a-6543, a-6583, a-6589, a-6746
\theCodelineNo, a-7133, *99*
\thecodelinenum, a-2610, a-3008, a-7135
\thedate, c-3166
\thefilediv, a-6272, a-6364, a-6366, a-6368, a-6385, a-6388, a-6569
\theglossary, a-6310
theglossary, a-5878
theindex, a-5439
\thesection, b-422
\thfileinfo, *22*, a-6746
\thickmuskip, c-2962
\thr@@, c-1789, c-1807
\time, c-2816, c-2817
\tinycae, c-2622
\title, *a-1832*, a-6540
\titlesetup, a-6525, a-6552, b-432
\TODO, c-2251
\toks, c-1508, c-1509, c-1510, c-1603, c-1604, c-1610, c-1611
\tolerance, *a-2315*, c-2696, c-2707, c-3180
\tOnLine, c-2310
\traceoff, b-377
\traceon, b-376
\trimmed@everypar, a-3116, a-3118
\truetextsuperscript, c-2169, c-2171
\ttverbatim, *a-2405*, a-4942, a-6879, e-346, e-565, e-593
\ttverbatim@hook, e-352, e-359, e-362
\twocoltoc, *a-1831*, c-2287, c-2296
type, *12*, a-3782
\tytul, c-3172
tytulowa, c-2831
\udigits, c-2047, c-2050, c-2485, c-2488
\un@defentryze, a-4314, a-4347
\un@usgentryze, a-4310, a-4358
\UnDef, *13*, a-4093, a-4106, a-4108, a-4110, a-4250
\undeksmallskip, c-2234
\UndoDefaultIndexExclusions, 16, a-5369
\unex@namedef, c-1999
\unex@nameuse, c-2004
\unexpanded, c-293, e-743
\ungag@index, a-4752, a-7190
\UniformSkips, *18*, a-2207, a-2229, a-2234, a-2241
\unless, c-2013
uresetlinecount, *10*, a-1963
\usage, a-7293
\usc, c-2443, c-2445
\uscacro, c-2445
\usecounter, c-1800
\UsgEntry, *19*, a-4428, a-7293
\uumlaut, b-333, b-351

\value, a-2585, c-1261
\varepsilon, c-1879, c-1936, c-2854
\varnothing, c-3001
\varsigma, c-2861
\vartheta, c-2855
\vee, c-2894, c-2960
\verb, *20*, a-3317, c-2025, c-2033, e-378, e-402, e-408, e-410, e-591, e-710
\verb*, e-377, e-591
\verb@balance@group, e-612, e-615, e-646, e-648
\verb@egroup, a-3316, e-615, e-646, e-649
\verb@eol@error, e-619
\verb@eolOK, e-633, e-641
\verbatim, e-472
verbatim, e-472
verbatim*, e-484
\verbatim@edef, e-506, e-510
\verbatim@end, e-507, e-511
\verbatim@nolig@list, e-347, e-651
\verbatimchar, *20*, a-3579, a-4300,

a-4524, a-5732, a-5734, a-7275, *101*
\verbatimhangindent, a-2158, e-561, e-572, e-573
\verbeolOK, *11*, e-641, *162*
\VerbHyphen, a-2138, e-261, *162*
\verbhyphen, a-6886, e-255, e-263, e-273, e-293
\VerbT, e-362
\VerbT1, e-362
\visiblespace, a-2756, a-6881, c-783, c-785, c-864, e-324, e-741, e-743
\VisSpacesGrey, *11*, a-2536, e-737, *163*
\voffset, f-386
\vs, c-864, c-871, c-875
\vspace*, c-2826

\wd, c-1859, c-1862, c-1892, c-1897, c-1905, c-1906, c-2136, c-2888, c-2991, c-2992, c-3003, c-3004
\Web, 21, c-1929
\Webern@Lieder@ChneOelze, a-4175
\wedge, c-2895, c-2960
\whenonly, c-2578
\whern, c-3214
\wherncore, c-3206, c-3216, c-3221
\whernskip, c-3215, c-3218, c-3219
\whernup, c-3221
\widowpenalty, *a-2310*
withmarginpar, *10*, a-2024
\WPheadings, c-1631
\Ws, c-3228
\wyzejnizej, c-2663
\Wz, c-3231

\xdef@filekey, a-6246, a-6250, a-6268
\Xedekfracc, c-2055
\XeLaTeX, c-1951
\XeTeX, 22, c-1944
\XeTeXdefaultencoding, b-286, b-291
\XeTeXinputencoding, c-162
\XeTeXpicfile, c-2133, c-2150
\XeTeXthree, b-339, c-2017
\XeTeXversion, c-151, c-152, c-161, c-723, c-2012, c-2013, c-2717, c-2787
\xiiand, c-769
\xiibackslash, c-756, c-760
\xiiclub, a-3344, a-5674, e-337
\xiilbrace, a-6886, a-6888, c-731, e-273, e-307
\xiipercent, a-5021, a-5023, a-6063, a-6067, a-6857, a-6872, a-6881, a-6883, a-6887, a-6896, a-6905, a-6931, c-765, e-255
\xiirbrace, c-732
\xiispace, c-342, c-343, c-772, c-785
\xiistring, a-3435, a-4519, a-4565, a-4783, a-4789, a-4804, a-4811, a-4849, c-341
\xiiunder, c-721, c-724, c-725
\XKV@ifundefined, b-285, b-290
\xxt@visiblespace, c-782, c-783

\year, a-6065, a-6069
\yeshy, c-2510

\zf@euencfalse, b-338
\zf@scale, c-2630, c-2633, c-2634
\zwrobcy, c-3169

-, c-2756, c-2793
\–, c-2719, c-2774, c-2792

File Key: a=gmdoc.sty, b=gmdocc.cls, c=gmutils.sty, d=gmiflink.sty, e=gmverb.sty, f=gmeometric.sty, g=gmoldcomm.sty