

Grzegorz Murzynowski

## The gmutils Packages Bundle \*

Copyright © 2005, 2006, 2007, 2008, 2009, 2010, 2011

by Grzegorz ‘Natrór’ Murzynowski

natror (at) gmail (dot) com

This program is subject to the L<sup>A</sup>T<sub>E</sub>X Project Public License.

See <http://www.ctan.org/tex-archive/help/Catalogue/licenses.lpl.html> for the details of that license.

LPPL status: "author-maintained".

Many thanks to my T<sub>E</sub>X Guru Marcin Woliński for his T<sub>E</sub>Xnical support.

For documentation please refer to the file(s)

**gmutils.{gmd,pdf}**.

```
47 <*master>
```

(A handful of meta-settings skipped)

```
101 </master>
```

```
102 <*ins>
```

```
\supposedJobname 103 \def\supposedJobname{%
```

```
104     gmutils%
```

```
105 }
```

```
107 \let\xA\expandafter
```

```
108 \let\nX\noexpand
```

```
109 \long\def\firstofone#1{#1}
```

```
111 \long\def\@firstoftwo #1#2{#1}
```

```
112 \long\def\@secondoftwo #1#2{#2} % in LATEX they are short. Which is bad.
```

```
114 \unless\ifnum\strcmp {\jobname} {\supposedJobname} =0
```

If we want to generate files from this file, we should call

```
xelatex --jobname=<sth. else> gmutils.gmd
```

Then the `\strcmp` primitive expands to some nonzero value and the conditional turns true.

```
121 \NeedsTeXFormat{LaTeX2e}[1996/12/01]
```

```
\gmBundleName 123 \def\gmBundleName{%
```

```
124     gmutils%
```

```
125 }
```

```
\currentBundle 127 \def\currentBundle{%
```

```
128     utilsbundle%
```

```
129 }
```

```
131 \edef\batchfile{\gmBundleName .gmd}
```

---

\* This file has version number v0.996 dated 2011/10/12.

```

134 \input docstrip.tex
\NOO 136 \def\NOO{\FromDir\gmBundleFile .gmd}
      Note it's \def so the BundleName expands to its current value.
139 \let\skiplines\relax
140 \let\endskiplines\relax
141 \askforoverwritefalse
\MetaPrefixS 143 \def\MetaPrefixS{\MetaPrefix\space}
\perCentS 144 \def\perCentS{\perCent\space}
146 \begingroup
147 \endlinechar=\newlinechar
148 \catcode\newlinechar=12\relax%
150 \catcode`\^=12\relax%
151 \catcode`\ =0\relax % Tifinagh Letter Yay
152 \catcode`\|=12 relax %
153 \catcode`<=12 relax %
154 \firstofone{ endgroup %
155     def preamBeginningLeaf{%
156         RCSInfo
157         MetaPrefixS This is file "outFileName" generated with the
158             DocStrip utility.
159         MetaPrefixS
160         ReferenceLines %
161         MetaPrefix %
162     }% of \preamBeginningLeaf
163
164     def copyRightLeaf{Copyright © }%
165     def licenseNoteLeaf{%
166         This program is subject to the LaTeX Project Public License.
167         MetaPrefixS See
168             http://www.ctan.org/tex-archive/help/Catalogue/licenses.lppl.html
169         MetaPrefixS for the details of that license.
170         MetaPrefix
171         MetaPrefixS LPPL status: "author-maintained".
172         MetaPrefix %
173     }% of \licenseNoteLeaf
174
175     def preamEndingLeaf{%
176         gmBundleFile.{gmd,pdf} gobble{ or \file{NatrOr-OperaOmnia.{%
177             gmd,pdf}}}.
178         MetaPrefixS %
179     }% of \preamEndingLeaf
180
181     def providesStatement{%
182         \NeedsTeXFormat{LaTeX2e}
183         \Provides gmFileKind{ gmOutName}
184         space space space space[ gmFileDate space gmFileVersion space
185             gmFileInfo space (GM) ]
186     }%
187 }% of \firstofone of changed catcodes.
\beforeDot 194 \def\beforeDot#1.#2\empty{#1}

```

```

\firstoftwo 196 \def\firstoftwo#1#2{#1}
\secondoftwo 197 \def\secondoftwo#1#2{#2}

    To gobble the default heading lines put by DocStrip:

200 \Name\def{ds@heading}#1{}

\csnameIf 202 \def\csnameIf#1{%
203   \ifcsname#1\endcsname
204   \csname#1\xA\endcsname
205   \fi
206 }

\writeto 208 \def\writeto#1{\edef\destdir{#1}}
\FromDir 209 \def\FromDir{}
\writefrom 210 \def\writefrom#1{\def\FromDir{#1/}}
\FromDir 212 \def\WritePreamble#1{%
\WritePreamble 213   \xA\ifx\csname pre@\@stripstring#1\endcsname\empty
214   \else
216     \edef\outFileName{\@stripstring#1}%
218     \edef\gmOutName{%
219       \xA\beforeDot\outFileName\empty
220     }% of \gmOutName

222     \edef\gmOutTitle{%
223       \xA\xA\xA\detokenize\xA\xA\xA{%
224         \csname \gmOutName Title\endcsname}%
225     }% of \gmOutTitle

227     \edef\gmOutYears{%
228       \csnameIf {\gmOutName Years}%
229     }%

231     \edef\gmOutThanks{%
232       \ifcsname \gmOutName Thanks\endcsname
233       \xA\xA\xA\detokenize\xA\xA\xA{%
234         \csname \gmOutName Thanks\endcsname
235       }%
236       \fi
237     }%

239     \edefInfo{Date}% \gmFileDate
240     \edefInfo{Version}% \gmFileVersion
241     \edefInfo{Info}% \gmFileInfo

243     \StreamPut#1{\csname pre@\@stripstring#1\endcsname}%
244     \fi}

```

First we look for the info at the leaf-level, then at standalone level, then at the bundle level. If we don't find it, it'll be empty.

```

\edefInfo 248 \def\edefInfo#1{%
249   \Name\edef{gmFile#1}{%
250     \ifcsname \gmOutName Leaf#1\endcsname % e.g. gmbaseLeafVersion
251     \xA\xA\xA\detokenize\xA\xA\xA{%
252       \csname \gmOutName Leaf#1\endcsname
253     }%
254   \else
255     \ifcsname \gmOutName #1\endcsname % e.g. gmbaseVersion

```

```

256     \xA\xA\xA\detokenize\xA\xA\xA{%
257     \csname \gmOutName #1\endcsname
258     }%
259     \else
260     \ifcsname \gmBundleFile #1\endcsname % e.g. gmutilsVersion
261     \xA\xA\xA\detokenize\xA\xA\xA{%
262     \csname \gmBundleFile #1\endcsname
263     }%
264     \fi
265     \fi
266     \fi
267     }% of edefined macro
268 }% of \edefInfo

270 \let\gmOutName\relax
271 \let\gmOutTitle\relax
272 \let\gmOutYears\relax
273 \let\gmFileDate\relax
274 \let\gmFileVersion\relax
275 \let\gmFileInfo\relax
276 \let\gmOutThanks\relax
277 \let\gmBundleFile\relax
278 \let\gmFileKind\relax

281 \declarepreamble\gmdLeaf
282 \preamBeginningLeaf

284 \copyrightLeaf \gmOutYears
285 by Grzegorz 'Natror' Murzynowski
286 natror (at) gmail (dot) com

288 \licenseNoteLeaf

290 For documentation please refer to the file(s)
291 \preamEndingLeaf
292 \providesStatement
293 \endpreamble

295 \keepsilent

    We declare all the preambles later and use the \empty Docstrip preamble.

299 \errorcontextlines=1000

301 \@makeother\^^A
302 \@makeother\^^B
303 \@makeother\^^C
304 \@makeother\^^V

\gmfile 308 \def\gmfile
309 #1% file name
310 #2% DocStrip directive(s)
311 #3% file extension
312 {%
313   \file{gm#1.#3}{\from{\gmBundleFile/\NOO}{#2}}%
314 }

\pack 317 \def\pack#1{\gmfile{#1}{#1}{sty}}

```

```

319 \begingroup\catcode`\ =9
320 \catcode`\^^I=9\relax
321 \catcode`\^^M=9\relax
322 \firstofone{\endgroup

\gmBundleFile 325 \def\gmBundleFile{gmutils}

327 \generate{
329 \usepreamble\gmdLeaf

\gmFileKind 331 \def\gmFileKind{ Package }

334 \writeto{ gmutils }

336 \pack{ base }% gmbase
337 \pack{ utils }% gmutils
338 \pack{ command }% gmcommand
339 \pack{ ampulex }% gmampulex
340 \pack{ envvir }% gmenvir
341 \pack{ relsize }% gmrelsize
342 \pack{ meta }% gmmeta
343 \pack{ logos }% gmlogos
344 \pack{ notonlypream}% gmnotonlypream
345 \pack{ mw }% gmmw
346 \pack{ typos }% gmtypos
347 \pack{ parts }% gmparts
348 \pack{ url }% gmurl
349 \pack{ RCS }% gmRCS
350 }
351 }% of changed catcodes' \firstofone

353 \Msg{*****}
354 \Msg{ }
355 \Msg{ To finish the installation you have to move}
356 \Msg{ the generated files into a directory searched by TeX.}
357 \Msg{ }
358 \Msg{ To type-set the documentation, run the file '\NOO'}
359 \Msg{ twice through LaTeX and maybe MakeIndex it. }
360 \Msg{ }
361 \Msg{*****}

364 \csname fi\endcsname % probably for the directive's clause
365 \csname endinput\expandafter\endcsname %
366 \fi % of unless job name other than name of this file, which indicates the DocStrip
pass.

369 </ins>

```

## Contents

<b>Intro</b> . . . . .	7	<b>Ampulex Compressa-like</b>	
Brave New <b>gmutils</b> and its inner		<b>modifications of macros—the</b>	
dependencies . . . . .	7	<b>gmampulex package</b> . . . . .	131
Installation . . . . .	8	A definer for expandable loops . . . . .	134
Contents of the <b>gmutils.zip</b> archive . . . . .	8	<b>The gmenvir Package</b> . . . . .	134
Compiling of the documentation . . . . .	8	<b>Environments redefined</b> . . . . .	135
<b>Options</b> . . . . .	9	Almost an environment or	
<b>The gmbase package</b> . . . . .	10	redefinition of <code>\begin</code> . . . . .	135
A couple of abbreviations . . . . .	10	<code>\ifenvir</code> and improvement of <code>\end</code> . . . . .	136
<code>\ifs</code> as a four-argument $\LaTeX$		<b>From relsize</b> . . . . .	138
command robust to unbalanced		<b>The gmmeta package for</b>	
<code>\ifs</code> and <code>\fis</code> . . . . .	12	<b>meta-symbols</b> . . . . .	139
<code>\firstofone</code> and the queer <code>\catcodes</code> . . . . .	16	<b>Macros for printing macros and</b>	
<code>\afterfi</code> and <code>pals</code> . . . . .	17	<b>filenames</b> . . . . .	140
<code>\foone</code> . . . . .	17	Typesetting arguments and commands . . . . .	142
<code>\@ifempty</code> , <code>\IfAmong</code> , <code>\IfIntersect</code>		<b>The gmlogos package—a couple of</b>	
<code>\@ifinmeaning</code> . . . . .	18	<b><math>\TeX</math>-related logos</b> . . . . .	145
Global Boolean switches . . . . .	32	<b>The gmnoton-</b>	
<code>\gm@ifundefined</code> —a test that		<b>lypream—modification of</b>	
doesn't create any hash entry		the 'only preamble' clause . . . . .	148
unlike <code>\@ifundefined</code> . . . . .	35	<b>Not only preamble!</b> . . . . .	148
Some 'other' and active stuff . . . . .	36	<b>Improvements to mwcls sectioning</b>	
<code>\@ifnextcat</code> , <code>\@ifnextac</code> ,		<b>commands</b> . . . . .	149
catcode-independent <code>\gm@ifstar</code> ,		An improvement of MW's	
<code>\@ifnextnotgroup</code> , <code>\@ifnextgroup</code> . . . . .	38	<code>\SetSectionFormatting</code> . . . . .	152
Storing and restoring the catcodes of		Negative <code>\addvspace</code> . . . . .	153
specials . . . . .	45	My setup of headings for <b>mwcls</b> . . . . .	154
Storing and restoring the meanings		Compatibilising standard and <b>mwcls</b>	
of CSes . . . . .	46	sectionings . . . . .	155
Setting for $X\TeX$ . . . . .	52	<b>The gmtypos package—some</b>	
Expandable turning stuff all into		<b>bits of typography (in a</b>	
'other' . . . . .	52	<b>non-systematic way)</b> . . . . .	157
Show must go on . . . . .	52	<b>Brave New World of <math>X\TeX</math></b> . . . . .	157
Second class document class . . . . .	54	Fractions . . . . .	158
Storing the catcode of line end . . . . .	54	Settings for mathematics in main font . . . . .	159
<code>\resizegraphics</code> . . . . .	54	Minion and Garamond Premier	
Comparison of detokenised strings . . . . .	56	kerning and ligature fixes . . . . .	170
Hashes for meta-defining macros . . . . .	57	A left-slanted font . . . . .	171
Deducing whether hash was braced . . . . .	63	Fake Old-style Numbers . . . . .	172
Some typesetting macros . . . . .	68	<b>Varia</b> . . . . .	176
Absolute values of <code>dimens</code> & <code>nums</code> . . . . .	70	Faked small caps . . . . .	180
<b>Combining accents (in <math>X\TeX</math> only)</b> . . . . .	73	See above/see below . . . . .	181
<b>The (gmutils package) options</b> . . . . .	75	<b>luzniej and napa!</b>	
<code>\DeclareCommand</code> and <code>\DeclareEn!</code>		<b>pierki</b> —environments for	
<b>vironment—the gmcommand</b>		fine-tuning of paragraphs vs. page	
<b>package</b> . . . . .	76	breaks . . . . .	181
The <code>\SameAs</code> parsing . . . . .	116	Typesetting dates in my memoirs . . . . .	185
Immediate uses . . . . .	118	For <code>dati</code> under poems . . . . .	189
Setting for $X\TeX$ . . . . .	120	Thousand separator . . . . .	191
Getting height ;-) . . . . .	128		
Enlarging and scaling of the font size . . . . .	128		

Footnotes suggested by Andrzej Tomaszewski . . . . .	193	Switching on and off parts of one file . . . . .	199
Only this paragraph . . . . .	194	Fix of including when <code>fontspec</code> is used . . . . .	200
Conditional tilde . . . . .	194	The <code>gmurl</code> package . . . . .	200
A really empty page . . . . .	194	<code>hyperref</code> 's <code>\nolinkurl</code> into <code>\url*</code> . . . . .	200
<code>enumerate*</code> and <code>itemize*</code> . . . . .	197	A fix to the <code>url</code> package . . . . .	201
The <code>gmparts</code> pack- age—in/exclusion of parts of one file analogous to <code>\include</code> . . . . .	197	The <code>gmRCS</code> package . . . . .	205
<code>\include</code> not only <code>.tex</code> 's . . . . .	198	Back to <code>gmutils</code> . . . . .	206
		Third person pronouns . . . . .	206
		Change History . . . . .	207
		Index . . . . .	212

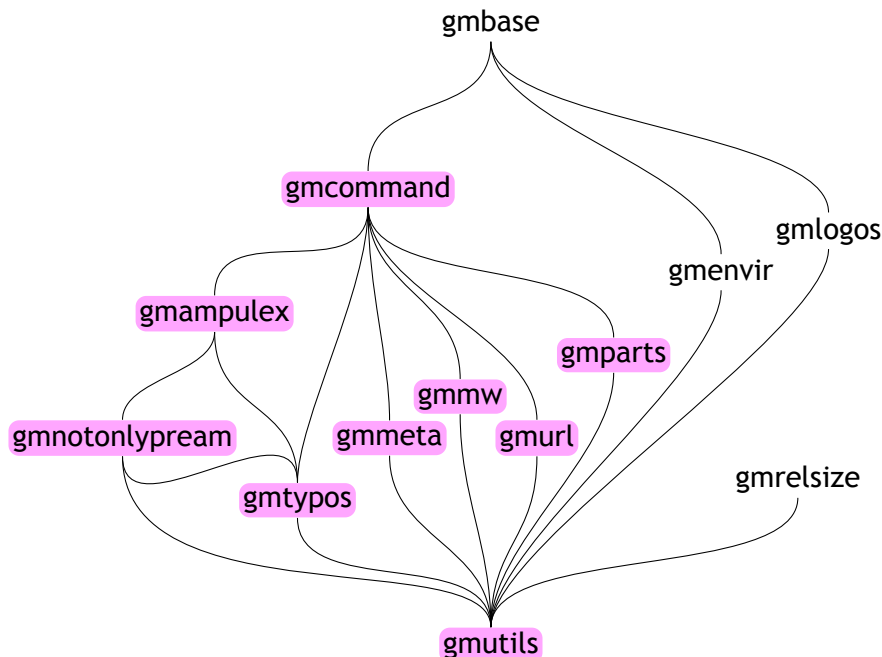
## Intro

The `gmutils` package bundle 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  $(\LaTeX)$ -awareness of the user.

## Brave New `gmutils` and its inner dependencies

For details just read the code part (where you'll find some comments) or intros to the particular packages. Here let's give a short description of what you could expect of them.



- **gmbase**: basic, low-level macros such as `\@ifnextcat` and other tests for peeping next token, including such that respect blank space; conditionals in an argument form robust to open `\if`'s and `\fi`'s. Also some expandable tests comparing strings of detokenised or not detokenised tokens (use of `\strcmp`); test whether a token is of a given kind (`\dimen`, `\skip`, `\count` &c.).

- **gmenvir**: a modification of `\begin` and `\end` to fully expand and detokenise environment’s name, so that the comparisons are fully compatible with `\csname...\endcsname`.
- **gmcommand**: probably the most important package of mine, providing a brand new implementation of the ancient (pre-`expl3`) idea of a command to declare L<sup>A</sup>T<sub>E</sub>X commands with many different optional arguments. This package implements `\DeclareCommand`, `\DeclareEnvironment` (you can use `#1...#9` also in the end-defs!). For the details see the package’s intro, where all the arg. specifiers are described.
- **gmlogos**: a couple of T<sub>E</sub>X-related logos, with a trial of improving the position of »A« in L<sup>A</sup>T<sub>E</sub>X. (Presented at BachoT<sub>E</sub>X 2007).
- **gmresize**: some macros taken from the **resize** package not to load all of them, and some new added, namely `\largerr` and `\smallerr`.
- **gmampulex**: modification of macros including those having parameters without total redefinition of them: you give the start tokens, the end tokens and the replacement. Use with care.
- **gmnotonlypream**: a modification of the `\@onlypreamble` declaration to provide a bit more informative error message and removal of many comands from the “only preamble” list that are useful also in the document.
- **gmurl**: some fixes to the **url** package not to use math mode but `\scantokens` which allows to get proper kerning.
- **gmparts**: in/exclude parts of one and the same file just as if you’d do with `\include` and `\includeonly` (in fact, you use *the* `\includeonly` to get some parts excluded).
- **gmtypos**: macros written while typesetting real books for money. Most of them do conform Polish typesetting standards of the 1960’s, which I like best, or refined advice of leading (contemporary) Polish typographers (e.g. `\ATfootnotes`).
- **gmmw**: compatibilising the sectioning commands of my favourite MWCLS classes with the standard ones.
- **gmmeta**: a couple of macros for description of macros.

## Installation

Unpack the `\jobname-tds.zip` archive (this is an archive that conforms the TDS standard, see [CTAN/tds/tds.pdf](#)) in some texmf directory or just put the **gmutils.sty** somewhere in the `texmf/\:tex/\:latex` branch. Creating a `texmf/\:tex/\:latex/\:gm` directory may be advisable if you consider using other packages written by me.

Then you should refresh your T<sub>E</sub>X distribution’s files’ database most probably.

## Contents of the gmutils.zip archive

The distribution of the **gmutils** package consists of the following three files and a TDS-compliant archive.

```
gmutils.gmd
README
gmutils.pdf
gmutils.tds.zip
```

## Compiling of the documentation

The last of the above files (the `.pdf`, i.e., *this file*) is a documentation compiled from the `.gmd` file by running L<sup>A</sup>T<sub>E</sub>X on the **gmutils.gmd** file twice (`xelatex gmutils.gmd` in the directory you wish the documentation to be in), then `MakeIndex` on the `\jobname.idx` file, and then L<sup>A</sup>T<sub>E</sub>X on `\jobname.\gmdExt` once more.

MakeIndex shell commands:

```
makeindex -r gmutils
```



```
makeindex -r -s gmglo.ist -o gmutils.gls gmutils.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 `gmdoc.cls`), `gmverb.sty`, the `gmutils` bundle, `gmiflink.sty` and also some standard packages: `hyperref.sty`, `color.sty`, `geometry.sty`, `multicol.sty`, `lmodern.sty`, `fontenc.sty` that should be installed on your computer by default.

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 T<sub>E</sub>X distribution's files' database most probably.

If you had not installed the `mwcls` classes (available on CTAN and present in T<sub>E</sub>X 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.

```
777 <*base>
```

`% &utils&command&envir` doesn't make sense since `gmbase` is loaded by them all and sets it properly.

```
781 \RequirePackage {expl3, xparse} % because it's used anyway: by fontspec, and
      thus we avoid name collisions.
```

```
784 \ifx\XeTeXversion\relax
```

```
785 \let\XeTeXversion\@undefined% If someone earlier used
      % \@ifundefined{XeTeXversion} to test whether the engine is XƒTEX, then
      % \XeTeXversion is defined in the sense of  $\varepsilon$ -TEX tests. In that case we \let
      it to something really undefined. Well, we might keep sticking to \@ifun|
      defined, but it's a macro and it eats its arguments, freezing their catcodes,
      which is not what we want in line 12481.
```

```
792 \fi
```

```
795 \ifdefined\XeTeXversion
```

```
796 \XeTeXinputencoding utf-8 % we use Unicode dashes later in this file.
```

```
797 \fi% and if we are not in XƒTEX, we skip them thanks to XƒTEX-test.
```

```
799 </base>
```

```
800 <*utils>
```

## Options

```
804 \unless\ifdefined\Name
```

```
\Name 805 \def\Name#1#2{\expandafter#1\cscname#2\endcscname}
```

```
806 \fi
```

```
809 \unless\ifcscname ifgmu@quiet\endcscname
```

```
810 \Name\newif {ifgmu@quiet}% it has to be at least (at highest) in gmcommand since
      is used by it and not always entire gmutils is loaded.
```

```
813 \fi
```

```
815 \RequirePackage{xkeyval}
```

```
817 \RequirePackage{gmbase}
```

818 </utils>

## The gmbase package<sup>1</sup>

This is the lowest-level package that defines such things as `\gmu@ifempty`, a handful of “if next” tests &c.

826 <\*base>

### A couple of abbreviations

829 `\unless\ifdefined\strcmp`

830 `\let\strcmp\pdfstrcmp`

831 `\fi`

`\@xa` 834 `\let\@xa\expandafter`

`\@nx` 835 `\let\@nx\noexpand`

`\@xanx` 837 `\def\@xanx{\@xa\@nx}`

`\@xade` 839 `\long\def\@xade#1{\@xa\def\@xa#1\@xa}`

`\@xa` 841 `\long\def\@csn#1{\csname #1\endcsname}`

`\@csn`

Note that it differs from L<sup>A</sup>T<sub>E</sub>X’s `\@nameuse` in `\longness`.

`\Name` 844 `\long\def\Name#1#2{\@xa#1\csname #2\endcsname}`

`\@xau` 847 `\long\def\@xau#1{\unexpanded\@xa{#1}}`

Note that there’s only one `\expandafter`: after `\unexpanded`. It’s because `\unexpanded` expands expandable tokens and gobbles `\relaxes` while looking for an opening brace or `\bgroup`.

Note also that (since v0.991) this is a 1-parameter macro so doesn’t expand subsequent tokens until meets a *<balanced text>* but just takes first single token or *<text>*.

`\gmu@firstandspace` 860 `\def\gmu@firstandspace#1{#1 }`

`\strip@bslash` 862 `\long\def\strip@bslash#1{%`

863 `\gmu@ifempty{#1}{}{%`

864 `\gmu@if {cat}{\@nx~\@nx#1}% this is specially for an active backslash (have you ever met it?). Thanks to this special case the inner macro declared for an active \ by \DeclareCommand is \\ not \csname\endcsname\.`

868 `{\string#1}% if #1 is active`

869 `{%`

`%% \@xa@ifnum\@xa\escapechar\@xa=\@xa` % looks great, but what if #1 is 92 (while normal \escapechar)? or \1?`

`%% \if\bslash`

873 `\@xa\@xa\@xa@ifnum\@xa\@xa\@xa\escapechar`

874 `\@xa\@xa\@xa=\@xa\@xa\@xa\@xa\@xa\gmu@firstandspace`

875 `\string#1\@xa@gobble`

876 `\else\@xa\@firstofone \fi`

877 `{\string#1}%`

878 `}% if #1 is not active`

<sup>1</sup> This file has version number v0.996 dated 2011/10/12.

```

879   }% of if #1 not empty
880 }

\slash@or@ac 884 \long\def\slash@or@ac#1{%
    If #1 is a CS or a name, we make it beginning with a backslash. Otherwise we keep it
    only \stringed.

887   \ifcat\@nx~\@nx#1%
888   \else
889     \backslash
890   \fi
891   \strip@backslash{#1}%
892 }

\@xanxcs 895 \long\def\@xanxcs #1{%
    \noexpand indifferent to whether the argument is a name or an active char.

898   \ifcat\@nx~\@nx#1%
899     \@nx#1%
900   \else
901     \@xa\@nx\csname #1\endcsname
902   \fi
903 }

\@xanxcssimple 906 \long\def\@xanxcssimple #1{%
    \noexpand indifferent to whether the argument is a name or an active char.

909     \@xa\@nx\csname #1\endcsname
910 }

    Meaning of a csname protected with \unexpanded

\@xaucs 915 \long\def\@xaucs#1{%
916   \unexpanded\@xa\@xa\@xa{\csname #1\endcsname}%
917 }

\@xanxtri 920 \long\def\@xanxtri#1{%
    \noexpand indifferent to whether the argument is a name, an active char or a CS.
    Warning. It applies \string to the first token of #1 so doesn't expand it if it's a macro.

926   \ifcat\@nx~\@nx#1%
927     \@nx#1%
928   \else
929     \@xa\@nx\csname \strip@backslash{#1}\endcsname
930   \fi
931 }

\pdef 935 \def\pdef{\protected\def}

\lpdef 938 \def\lpdef{\long\protected\def}
939 \let\pldef\lpdef

\gmu@ifdefinable 942 \long\def\gmu@ifdefinable
943 #1% a CS #2 (implicit) what if definable (ifx undefined or relax) #3 (implicit) what if
    not definable (ifx not undefined nor relax)
944 {%
945   \ifx #1\@undefined

```

```

948     \@xa \@firstoftwo
949   \else
950     \@xa \@secondoftwo
951   \fi
952   {\@firstoftwo}%
953   {\ifx #1\relax
954     \@xa \@firstoftwo
955   \else
956     \@xa \@secondoftwo
957   \fi
958   }%
959 }

```

```
\pedef 962 \def\pedef{\protected\edef}
```

```
\pxdef 964 \def\pxdef{\protected\xdef}
```

And this one is defined, I know, but it's not `\long` with the standard definition and I want to be able to `\gobble` a `\par` sometimes.

```
\gobble 971 \long\def\gobble#1{}
```

```
\@gobble 973 \let\@gobble\gobble
```

```
\gobbletwo 974 \let\gobbletwo\@gobbletwo % it's a LATEX's \long macro (in File d: ltdefns.dtx,
Date: 2004/09/18 Version v1.3g l. 939)
```

```
\@gobbleeight 977 \long\def\@gobbleeight#1#2#3#4#5#6#7#8{}
```

```

981 \long\pdef\provide#1{%
982   \ifdefined#1%
983     \ifx\relax#1\afterfifi{\def#1}%
984     \else\afterfifi{\gmu@gobdef}%
985   \fi
986   \else\afterfi{\def#1}%
987   \fi}

```

```

990 \long\def\gmu@gobdef#1#{%
991   \def\gmu@tempa{}}% it's a junk \def-assignment to absorb possible prefixes.
992 \@gobble
993 }

```

```
997 \def\pprovide{\protected\provide}
```

Note that both `\provide` and `\pprovide` may be prefixed with `\global`, `\outer`, `\long` and `\protected` because the prefixes stick to `\def` because all before it is expandable. If the condition(s) is false (`#1` is defined) then the prefixes are absorbed by a junk assignment.

Note moreover that unlike L<sup>A</sup>T<sub>E</sub>X's `\providecommand`, our `\(p)provide` allow any parameters string just like `\def` (because they just *expand* to `\def`).

```

1010 \long\def\@nameedef#1#2{%
1011   \@xa\edef\csname#1\endcsname{#2}}

```

### **\ifs as a four-argument L<sup>A</sup>T<sub>E</sub>X command robust to unbalanced \ifs and \fis**

```

1016 \pdef\gmu@DefSymbol#1{%
1017   \unless\ifdefined#1%
1018     \def#1{#1}%
1019   \fi

```

```

1020 }
\newdef 1023 \newcommand\newdef % sort of newcommand that accepts prefixes.
1024 [1]%
1025 {%
1026   \gmu@ifdefinable #1%
1027   {\pdef #1}
1028   {%
1029     \PackageError {gmbase} {\@nx#1 already defined.}{}%
1030     \gmu@gobdef
1031   }%
1032 }

1034 \protected\newdef \pnew {%
1035   \protected\newdef
1036 }

%% |\pdef\globalize#1{\global#1=#1} % doesn't make sense general enough (2010/11/03)
1042 \long\def\gmu@if #1#2{%

    2011/10/11, 15.23 (GM) we move the special case of \ifincsname to a separate macro
    as used exactly once and only in a very special and personal macro.

1047   \csname if#1\endcsname #2%
1048   \@xa\@firstoftwo
1049   \else\@xa\@secondoftwo
1050   \fi
1051 }

    A little \expandafter tip: to get the effect of \expandafter\ifx<stuff> write
    \gmu@if {x\expandafter\expandafter}, where x stands for any conditional primitive
    suffix.

1057 \long\def\gmu@notif#1#2{%

    We leave the name \gmu@unlessif for analogon of \gmu@if prefixed with \unless,
    which works slightly different than reversion of #3 and #4 (if the tail of #2 remains after test,
    it becomes part of true branch for unprefixd and part of false branch for \unless-prefixed
    version). (2010/7/25)

1063   \gmu@if {#1}{#2}%
1064   \@secondoftwo\@firstoftwo
1065 }

    And simplified versions of the testing macros, for the switches, because I many times
    forgot to add the empty #2 (which of course lead to a disaster at best (at worst—to a
    perfidious bug that remains hidden for years)).

1072 \def\gmu@ifsw #1{\gmu@if {#1}{} }
1073 \def\gmu@notsw #1{\gmu@notif {#1}{} }

1075 \def \gmu@ifincsname

    (implicit) #1 what if in,
    (implicit) #2 what if not in.

1078 {%
1079   \ifincsname
1080   \@xa\@firstoftwo

```

```

1081 \else
1082   \@xa\@secondoftwo
1083 \fi
1084 }

1086 \long\def\gmu@unless #1#2{%
1087   \@xa\unless \csname if#1\endcsname #2%
1088   \@xa\@firstoftwo
1089   \else
1090     \@xa\@secondoftwo
1091   \fi
1092 }

```

For a special case of downright nesting of conditionals let's provide a shorthand. But wait a minute. This special case, which from T<sub>E</sub>Xnical point of view is a tree growing downright, is just a `cases` special form from usual languages. Therefore let's name it `case(s)`.

It has one inconvenience: the last (innermost) false branch (the last case) also has to be preceded with `\gmu@EatDownright` (or just `be` this macro).

```

1107 \lpdef\@iwru@EC#1#2#3{%
1108   \@iwrum{#1»{#2}« »#3«}%
1109   \gmu@passbraced{#1{#2}}{#3}%
1110 }

1113 \long\def\gmu@generalCASE
1114 #1% Testing macro (\gmu@if etc.
1115 #2% #1 of the testing macro
1116 #3% #2 of the testing macro
1117 #4% #3 of the testing macro (what if test satisfied)
      (#4 of the testing macro will always be empty)
1119 {%
1120   #1{#2}{#3}%
1121   {%
1122     \gmu@EatCases{#4}}}%
1123 }

```

If the condition is satisfied, #3 is executed after eating all the stuff succeeding it up to a delimiter CS`\gmu@ESAC`.

```

1127 \long\def\gmu@EatCases
1128 #1%
1129 #2\gmu@ESAC
1130 {#1}

1132 \let\gmu@lastCASE\gmu@EatCases

1134 \def\gmu@CASE {\gmu@generalCASE \gmu@if }
1135 \def\gmu@CASEnot {\gmu@generalCASE \gmu@notif }
1136 \def\gmu@CASExany {\gmu@generalCASE \gmu@ifxany }
1137 \def\gmu@CASExnone {\gmu@generalCASE \gmu@ifxnone }

1140 \long\def\gmu@reserved@firstofmany#1#2\gmu&nil{#1}

```

An expandable test whether argument is a single token or not. Beware: it expands to an open if.

```

1147 \long\def\ifsingletoken#1{%

```

%% \gmu &nil % such a strange delimiter to make it work both in letter and other @ scopes etc. (& seems to me recatcoded rather seldom)

```

1151 \ifnum \strcmp{\unexpanded{#1}}%
1152   {\@xa{\gmu@reserved@firstofmany #1\blekotnizza@ \di \broccoli
1153     \gmu&nil}%
1154   }% of right text of \strcmp
1155   =\z@
1156 }

```

The conditional of this macro turns true if #1 was a single token (of catcode  $\neq$  1, 2) or such a token in curly braces (remember that the braces are stripped also from delimited argument if it consists only of braced text).

Note that single (unbalanced) tokens of catcodes 1 or 2 cannot be passed this macro.

The conditional turns false when #1 (possibly after stripping braces) is of at least 2 tokens or is empty or is a blank space.

The last segment of last disjunction may be a bit confusing. But this macro is intended for determining whether we should pass #1 in braces or not and passing a blank in braces seems reasonable.

A macro that applies \string to its argument if it's not braced. To be expanded by \detokenize.

```

1178 \long\def\gmu@predetokstring #1{%
1179   \gmu@if {num}
1180   {%

```

After a couple of hours of debug I reached the proper test which is given below. The goal is to (expandably!) check whether #1 is braced and/or begins with a blank space. In any of those cases we don't hit it('s first token) with \string.

\@firstofone strips outermost pair of braces if any and gobbles the lading blank(s) if any so the detokenised strings will differ.

```

1189   \strcmp
1190   {\detokenize{#1x}}%
1191   {\detokenize\@xa{\@firstofone #1x}}=\z@
1192   }% of test
1193   {\@xa{\string#1}}
1194   {{#1}}%
1195 }

```

A test for comparison of CS es as macro delimiters (stronger than \ifx). Beware: it expands to an open if. And it has to, to be usable in \gmu@if.

```

1203 \long\def\ifstrings#1#2{%
1204   \ifnum\strcmp
1205     {\detokenize\gmu@predetokstring{#1x}}%
1206     {\detokenize\gmu@predetokstring{#2x}}=\z@

```

We hit both texts with \unexpanded in case they are more than one token. Note \string is not the same as \detokenize because the latter adds a space after a letter CS.

Moreover, a char is added to both texts to assure that \string has sth. to hit not the closing brace (which would lead to a disaster).

```

1214 }

```

As you see, it expands to an open if, but that's OK as far as we use it only via macros, e.g.

```
\gmu@if {strings}{\while\until}...
```

Thanks to this test defining CSeS that are intended only as atoms (symbols) ceases to be necessary. Which is quite an advantage, if we wish to use symbol CSeS such as `\while`, `\until` or `\SameAs` (in `\DeclareCommand`) that with this test may still be available for `\newcommand`.

And another, slightly different: turns true iff the arguments are equal after (stringing and) possible stripping bslash(es), e.g. `par` and `\par` (well, *any* escape char that is currently in charge).

```
1232 \long\def\ifstribs#1#2{%
1233   \ifnum\strcmp{\strip@bslash{#1}}{\strip@bslash{#2}}=\z@
1234 }
```

And a test with a database flavour: for tokens that are defined it's `\ifx`. For tokens `\ifx`-equal `\@undefined`—it's `\ifstrings` (in relational databases any usual comparison of two NULLs returns null so there's a distant resemblance):

```
1244 \long\def\ifStrX#1#2{%
1245   \gmu@CASEnot {x}{#1#2}%
1246   {\iffalse}% if tokens are x-unequal, all is clear.

1248   \gmu@CASEnot x {\@undefined#1}
1250   {\iftrue}%
```

some (i.e. *both*) tokens are-x `\@undefined` or `\relax`, then

```
1254   \gmu@CASE {strings}{#1#2}%
1255   {\iftrue}%
1257   \gmu@lastCASE
1258   {\iffalse}%
1259   \gmu@ESAC
1260 }
```

### `\firstofone` and the queer `\catcodes`

Remember that once a macro's argument has been read, its `\catcodes` are assigned forever and ever. That's what is `\firstofone` for. It allows you to change the `\catcodes` locally for a definition *outside* the changed `\catcodes`' group. Just see the below usage of this macro 'with T<sub>E</sub>X's eyes', as my T<sub>E</sub>X Guru taught me.

```
1273 \long\def\firstofone#1{#1}

1275 \long\def\bracefirstofone#1{#{#1}}

1277 \long\pdef\scantwo#1#2{
1278   \begingroup\endlinechar\m@ne
1279   \@xa\endgroup\scantokens{#1#2}%
1280 }

1282 \long\def\@firstofthree#1#2#3{#1}
1283 \long\def\@secondofthree#1#2#3{#2}
1284 \long\def\@thirdofthree#1#2#3{#3}
1285 \long\def\@twoofthree#1#2#3{#1#2}
1286 \long\def\@secondoffive#1#2#3#4#5{#2}
```

In some `\if[cat?]` test I needed to look only at the first token of a tokens' string (first letter of a word usually) and to drop the rest of it. So I define a macro that expands to the first token (or `{(text)}`) of its argument.

```
1293 \long\def\@firstofmany#1#2\@nil{#1}
```



```

1296 \long\def\@secondofmany#1#2#3\@nil{#2}
1298 \long\def\@allbutfirstof#1#2\@nil{#2}
1304 \long\def\@firstthensecond #1#2{#1#2} % Note this macro strips braces if present.
1306 \long\def\@secondthenfirst #1#2{#2#1} % Note as above.

```

### **\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 `\afterfi` and `pals` are sensitive to `\fis` that may occur in macros' arguments so probably the safest and expandable way is the `\expandafter\@(first|second)oftwo` trick.

```

1330 \def\longafterfi{%
1331   \long\def\afterfi##1##2\fi{\fi##1}}
1332 \longafterfi

```

And two more of that family:

```

1334 \long\def\afterfififi#1#2\fi#3\fi{\fi\fi#1}
1336 \long\def\afteriffififi#1#2\fi#3\fi{\fi#1}

```

Notice the refined elegance of those macros, that cover both 'then' and 'else' cases thanks to `#2` that is discarded.

```

1340 \long\def\afterififfifififi#1#2\fi#3\fi#4\fi{\fi#1}
1341 \long\def\afteriffiffififi#1#2\fi#3\fi#4\fi{\fi\fi#1}
1342 \long\def\afterfififfififi#1#2\fi#3\fi#4\fi{\fi\fi\fi#1}

```

### **\foone**

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

```

1349 \long\def\foone#1{\begingroup#1\relax\egfirstofone}
1351 \long\def\egfirstofone#1{\endgroup#1}
1353 \def\foeatletter{\foone\makeatletter}

```

```

\@empty 1359 \newcommand*\@emptyify[1]{\let#1=\@empty}
\emptyify 1360 \gmu@ifdefinable\emptyify{\let\emptyify\@emptyify}

```

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

```

\g@emptyify 1364 \newcommand*\g@emptyify{\global\@emptyify}
\gemptyify 1365 \gmu@ifdefinable\gemptyify{\let\gemptyify\g@emptyify}

```

```

\@relaxen 1368 \newcommand*\@relaxen[1]{\let#1=\relax}
\relaxen 1369 \gmu@ifdefinable\relaxen{\let\relaxen\@relaxen}

```

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

```

\g@relaxen 1373 \newcommand*\g@relaxen{\global\@relaxen}
\grelaxen 1374 \gmu@ifdefinable\grelaxen{\let\grelaxen\g@relaxen}

```

`\@ifempty, \IfAmong, \IfIntersect \@ifinmeaning`

After a short deliberation I make the `\IfAmong... \among` and `\IfIntersect` macros' names apeless since they are intended for the macro and class writers, at the same level of abstraction as `\DeclareCommand`.

```
1385 \long\def\gmu@ifempty#1{%
      % #1 the token(s) we test, it may be just {},
      % #2 the stuff executed if #1 is empty (is {}),
      % #3 the stuff executed if #1 consists of at least one token.
      %% \def\gmu@ifempty@resa{#1}%
      %% \ifx\gmu@ifempty@resa\@empty
```

```
1393 \ifnum\strcmp{\detokenize{#1}}{ }=\z@
```

Note that now (2010/6/23) it's expandable thanks to `\strcmp` and therefore it's no longer `\protected`. Another argument for `strcmp` is that it detokenizes its text so its robust to `#` es. But it expands all the macros which is not what we intended so we `\detokenize{#1}` anyway.

```
1399 \@xa\@firstoftwo
1400 \else\@xa\@secondoftwo
1401 \fi
1402 }
```

```
1405 \long\pdef\@ifnonempty#1#2#3{\gmu@ifempty{#1}{#3}{#2}}
```

```
1424 \long\def\gmu@ifexempty #1{%
1425 \ifnum \strcmp{#1}{ }=\z@
1426 \@xa\@firstoftwo
1427 \else\@xa\@secondoftwo
1428 \fi
1429 }
```

```
1432 \long\pdef\IfAmong
```

```
1433 #1% the token(s) whose presence we check,
```

```
1434 \among % delimiter of #1
```

```
1435 #2% the list of tokens in which we search #1, #3 (implicit) the 'if found' stuff,
      #4 (implicit) the 'if not found' stuff.
```

```
1438 {% Note this command has to be used with care since it recognizes the token(s) due
      to fitting a delimited macro, so it distinguishes any two different tokens even if
      they are \ifx-equal.
```

```
1444 \long\def\gmu@among@##1#1##2\gmu@among@{%
1445 \gmu@ifempty{##2}\@secondoftwo\@firstoftwo}%
1446 \gmu@among@#2#1\gmu@among@
1447 }
```

```
\ifgmu@ifquant 1450 \newif\ifgmu@ifquant
```

```
1454 \long\pdef\gmu@ifxany
```

```
1455 #1% a single token to be \ifx ed with each of #2
```

```
1456 #2% counterpart to the above—a sequence of tokens to check #1 against. It may contain
      anything including groups since checking is preceded by an assignment.
```

```
1460 {%
```

```
%% \gmu@ifempty{#1}{\PackageError{gmbase}{We don't consider this
%% case}{}}%
```

```
1463 \gmu@ifempty{#2}{\@secondoftwo}{%
```

we wrap the iteration over #2's tokens in `\gmu@ifempty` because we expect many empty #2's in `\DeclareCommand's \loop` arguments (such as `Q` and `U`)

```
1467 \gmu@ifquantfalse
1468 \let\gmu@ifxa@aasiter\@gmu@ifxa@aasiter
1470 \edef\gmu@ifxa@aas{% edef and unexpanded to protect agains #6 token(s) in #1.
1472 \unexpanded{%
1473 \ifx #1\gmu@ifxa@token
1474 \gmu@ifquanttrue
```

We are happy we found what we looked for but anyway we have to iterate to let all the possible groups to the drain.

```
1477 \let\gmu@ifxa@aas\gmu@ifxa@drainer
1478 \else
1479 \ifx \gmu@ifxa@Limit\gmu@ifxa@token
1480 \emptify \gmu@ifxa@aasiter
1481 \fi
1482 \fi
1483 \gmu@ifxa@aasiter
1484 }% of \unexpanded
1485 }% of \gmu@ifxa@aas
1487 \gmu@ifxa@aasiter #2\gmu@ifxa@PreLimit\gmu@ifxa@Limit
1488 \gmu@if {gmu@ifquant}{}%
1489 }% of if #2 nonempty
1490 }
1493 \def\gmu@ifxa@drainer{%
1494 \ifx\gmu@ifxa@Limit\gmu@ifxa@token
1495 \emptify\gmu@ifxa@aasiter
1496 \fi
1497 \gmu@ifxa@aasiter
1498 }
1500 \gmu@DefSymbol \gmu@ifxa@PreLimit
1501 \gmu@DefSymbol \gmu@ifxa@Limit
1504 \def\@gmu@ifxa@aasiter{%
```

It's a pattern CS to restore `\gmu@ifxa@aasiter` in each `\gmu@ifxany`.

```
1506 \afterassignment\gmu@ifxa@aas
1507 \let\gmu@ifxa@token= }% thanks to this space it'll look also at spaces (cat 10)
and = chars.
```

```
\gmu@ifxnone 1512 \long\pdef\gmu@ifxnone
1513 #1% token to be checked against
1514 #2% list of tokens to not find #1 at
1515 {%
1516 \gmu@ifxany{#1}{#2}\@secondoftwo\@firstoftwo
1517 }
```

```
%% \long\pdef\gmu@ifNXany
%% #1% a single token to be noexpanded and \if ed with each of
%% % #2 noexpanded
%% #2% counterpart to the above—a sequence of tokens to check #1
```

```

%% % against. It may contain anything including groups since checking is
%% % preceded by an assignment.
%% %
%% {%
%% \gmu@ifempty{#2}{\@secondoftwo}{%
%% % we wrap the iteration over #2's tokens in \gmu@ifempty because
%% % we expect many empty #2's in \DeclareCommand's \loop
%% % arguments (such as Q and U)
%% \gmu@ifquantfalse
%% \let\gmu@ifNXa@aasiter\@gmu@ifNXa@aasiter
%% %
%% \edef\gmu@ifNXa@aas{% edef and unexpanded to protect agains
%% % #6 token(s) in #1.
%% \unexpanded{%
%% \if \@nx#1\@nx\gmu@ifNXa@token
%% % an „honest” char token, i.e., not an active char neither a
%% % CS, when \let to a CS, keeps its catcode for
%% % \ifcat. An active char however, when let to a CS,
%% % loses its 13 and ifcat-is \relax (a CS). Therefore for
%% % active chars and CS es we use \ifx.
%% %
%% % This is useless: we cannot determine that a CS let
%% \ifnum
%% \numexpr
%% \ifcat \@nx#1\relax 0\else 1\fi *%
%% \ifcat \@nx#1~0\else 1\fi
%% +\ifcat \@nx\gmu@ifNXa@token\relax 0\else 1\fi
%% >\z@
%% % if at least one of compared tokens is not a CS neither
%% % an active char
%%
%% \fi
%% \gmu@ifquanttrue
%% % We are happy we found what we looked for but anyway we have to
%% % iterate to let all the possible groups to the drain.
%% \let\gmu@ifNXa@aas\gmu@ifNXa@drainer
%% \else
%% \ifx \gmu@ifNXa@Limit\gmu@ifNXa@token
%% \emptify \gmu@ifNXa@aasiter
%% \fi
%% \fi
%% \gmu@ifNXa@aasiter
%% }% of \unexpanded
%% }% of \gmu@ifxa@aas
%% %
%% \gmu@ifNXa@aasiter #2\gmu@if@PreLimit\gmu@ifNXa@Limit
%% \gmu@if {gmu@ifquant}{}%
%% }% of if #2 nonempty
%% }
%%
%% \def\gmu@ifNXa@drainer {%

```

```

%% \ifx\gmu@ifNXa@Limit\gmu@ifNXa@token
%% \emptyify\gmu@ifNXa@aasiter
%% \fi
%% \gmu@ifNXa@aasiter
%% }
%%
%%
%% \gmu@DefSymbol\gmu@ifNXa@Limit
%%
%%
%% \def\@@gmu@ifNXa@aasiter{%
%% % It's a pattern CS to restore \gmu@ifNXa@aasiter in each \gmu@ifNXany.
%% \afterassignment\gmu@ifNXa@aas
%% \let\gmu@ifNXa@token= }% thanks to this space it'll look also at
%% % spaces (cat 10) and = chars.
%%
%%
%% %
%% \long\pdef\gmu@ifNXnone
%% #1% token to be checked against
%% #2% list of tokens to not find #1 at
%% {%
%% \gmu@ifNXany{#1}{#2}\@secondoftwo\@firstoftwo
%% }
%%

```

We need a macro that iterates over every token/text on a list. L<sup>A</sup>T<sub>E</sub>X's `\@for` iterates over a list separated with commas so it's not the case. Our macro is much simpler.

```
1607 \long\def\gmu@foreach#1\gmu@foreach@delim#2{%
```

We define the iterator that takes one item from the list, checks it against

```

1611 \long\def\gmu@forer##1{%
1612 \gmu@if {strings} {\gmu@foreach@delim {##1}}%
1613 }% if we've met the delimiter, we stop.
1614 {% otherwise we wrap #1 in a macro to make it available to #2
1615 \edefU\gmu@forarg{##1}%
1616 #2% we execute #2 and probably continue iteration (unless the loop isn't broken
with the next macro).
1618 \gmu@forer
1619 }%
1620 }% of forer.

```

So we apply the defined iterator

```

1623 \gmu@forer#1\gmu@foreach@delim
1624 }

```

A macro to break the loop:

```

1627 \long\def\gmu@foreach@break
1628 #1\gmu@foreach@delim{

```

The “if strings” or “if stribs” test performed with small quantifier. It's not as robust and all-purpose as `\gmu@ifxany` because for obvious reasons we cannot use `\futurelet`.

Note however that thanks to the nature of the test we don't need the sentinel CS to be defined.

And this is expandable

```
1640 \long\def\gmu@ifstrany
1641 #1% kind of test (strings or stribs so far)
1642 #2% token to be checked against #3
1643 #3% the list of tokens to be iterated over
      #4 (implicit) what if found
      #5 (implicit) what if not found
1646 {%
1647 \gmu@ifstrany@{#1}{#2}#3\gmu@ifstrany@end
1648 \@firstoftwo\@secondoftwo
1649 }

1651 \long\def\gmu@ifstrany@
1652 #1% kind of test
1653 #2% left side of comparison
1654 #3% right side of comparison
1655 {%
1656 \gmu@if {#1}{#2#3}%
1657 \gmu@ifstrany@found
1658 {% else
1659 \gmu@if {#1}{#3\gmu@ifstrany@end}%
1660 \@secondoftwo
1661 {\gmu@ifstrany@{#1}{#2}}% if we didn't meet the sentinel, we iterate
1662 }%
1663 }

1666 \long\def\gmu@ifstrany@found
1667 #1\gmu@ifstrany@end
1668 {\@firstoftwo}
```

```
\gmu@ifstrany 1671 \def\gmu@ifstrany{\gmu@ifstrany {strings}}
1673 \def\gmu@ifsbany{\gmu@ifstrany {stribs}}
1675 \def\gmu@ifStrXany{\gmu@ifstrany {StrX}}
```

And the counterparts:

```
1678 \long\def\gmu@ifstrnone#1#2#3#4{%
1679 \gmu@ifstrany{#1}{#2}{#4}{#3}%
1680 }

1682 \long\def\gmu@ifsbnone#1#2#3#4{%
1683 \gmu@ifsbany{#1}{#2}{#4}{#3}%
1684 }

1686 \long\def\gmu@ifStrXnone#1#2#3#4{%
1687 \gmu@ifStrXany{#1}{#2}{#4}{#3}%
1688 }
```

And their downright versions:

```
1692 \lpdef\gmu@CASEstrany {\gmu@generalCASE \gmu@ifstrany }
1694 \lpdef\gmu@CASEstrnone {\gmu@generalCASE \gmu@ifstrnone }
1696 \lpdef\gmu@CASEsbany {\gmu@generalCASE \gmu@ifsbany }
1698 \lpdef\gmu@CASEsbnone {\gmu@generalCASE \gmu@ifsbnone }
```

Note that `\gmu@ifsXXany` iterate hash by hash, so don't distinguish a `CS\par` from a `\stringed` sequence of other chars `\par` passed them in braces. To avoid such a case we provide a `degrouper` with which we may prepare the text for token-by-token `\gmu@ifsXXany` test:

```

\degrouper@toks 1707 \newtoks\degrouper@toks

1709 \long\pdef\gmu@degrouper
1710 #1% text to be degrouper
1711 {\degrouper@toks={}}%
1712 \gmu@degrouper@iter#1\gmu@degrouper@end
1713 }

1715 \long\def\gmu@degrouper@afterlet{%
1716 \gmu@if x{\degrouper@lettoken\bgroup}%
1717 {\degrouper@drainanditer}%
1718 {\gmu@if x{\degrouper@lettoken\egroup}%
1719 {\degrouper@drainanditer}%
1720 {\degrouper@addanditer}%
1721 }%
1722 }

1724 \def\gmu@degrouper@iter{%
1725 \futurelet\degrouper@lettoken\gmu@degrouper@afterlet}

1727 \def\degrouper@drainanditer{%
1728 \afterassignment\gmu@degrouper@iter
1729 \let\gmu@drain=
1730 }

1732 \def\degrouper@addanditer{%
1733 \gmu@if x{\degrouper@lettoken\gmu@letspace}%
1734 {\addtotoks\degrouper@toks{ }}%
1735 \degrouper@drainanditer
1736 }{%
1737 \degrouper@addanditer@i
1738 }%
1739 }

1741 \long\def\degrouper@addanditer@i
1742 #1{%
1743 \gmu@if {strings}{#1\gmu@degrouper@end}%
1744 }% we've reached the end of iteration
1745 {% it's sth. to add
1746 \addtotoks\degrouper@toks{#1}%
1747 \gmu@degrouper@iter
1748 }%
1749 }

1754 \pdef\IfIntersect

```

This is a 4-argument command (not expandable) that checks whether the list of tokens `% #1` and `#2` have nonempty intersection in the sense of `\ifx` and if so it executes `#3` or `#4` otherwise:

- `#1` first list to match,
- `#2` second list to match,
- `#3` if match (nonempty intersection),

```

#4 if not match (empty intersection).
1765 {\gmu@ifintersect \gmu@ifxany}

1766 \lpdef\gmu@ifintersect
1769 #1% an iterating test (\gmu@ifxany, \gmu@ifstrany or \gmu@ifsbany so far)
1771 #2% one list to match
1772 #3% another list to match #4 (implicit) what if intersect
      #5 (implicit) what if don't
1775 {%
1776   \let\IfIntersect@next\@secondoftwo
1777   \gmu@foreach #2\gmu@foreach@delim{%
1778     \@xa #1\gmu@forarg{#3}%
1779     {\let\IfIntersect@next\@firstoftwo
1780       \gmu@foreach@break
1781     }}%
1782   }% of \gmu@foreach's #2.
1783   \IfIntersect@next
1784 }

1786 \pdef\gmu@ifstrintersect
1787 {\gmu@ifintersect\gmu@ifstrany}

1789 \pdef\gmu@ifsbintersect
1790 {\gmu@ifintersect\gmu@ifsbany}

```

A somewhat generalised `\expandafter`:

```

\@XAtoks 1795 \newtoks\@XAtoks

1797 \long\pdef\@XA#1{%
1798   \@XAtoks={#1}%
1799   \@xa\the\@xa\@XAtoks}

1804 \long\pdef\@ifinmeaning#1\of#2{%
      % #1 the token(s) whose presence we check,
      % #2 the macro in whose meaning we search #1 (the first token of this argu-
      %     ment is expanded one level with \expandafter),
      % #3 the 'if found' stuff,
      % #4 the 'if not found' stuff.
1821   \@XA{\IfAmong#1\among}\@xa{#2}}

1823 \gmu@DefSymbol\defNoHash
1824 \gmu@DefSymbol\defHashy
1825 \gmu@DefSymbol\boolean

1827 \def\gmu@geteschar{%

      A macro that edefines detokenised char of the charcode \escapechar

1829   \edef\gmu@xiieschar{%
1830     \gmu@CASE {num} {\escapechar <\z@}
1831     }%
1833   \gmu@CASE {num}{\escapechar <32 }
1834   {\@xa \@firstofmany \string \blekotnizza\@nil}% not firstthreeofmany!!!!

1836   \gmu@CASE {num} {\escapechar=32 }
1837   { }% space cannot be "first of...".
1838   \gmu@lastCASE

```



```

1839     {\@xa \@firstofmany \string \blekotnizza \@nil}%
1840     \gmu@ESAC
1841   }%
1842 }
1845 \long\def\gmu@IfBooleanMacro#1{%
    this macro should provide a yes-no answer (\@firstoftwo/\@secondoftwo).

1847   \gmu@ifedetokens{\meaning#1}{macro:->true}%
1848   {\@firstoftwo}%
1849   {\gmu@ifedetokens{\meaning#1}{macro:->false}%
1850   {\@firstoftwo}%
1851   {\@secondoftwo}%
1852   }%
1853 }

1856 \pdef\IfIs
1857 #1% a CS
1858 #2% \dimen, \long, \toks, \skip, \count, \dimexpr, \numexpr, \glueexpr, \newif
    for \iffalse and \iftrue, \if or \conditional for any Boolean test/switch,
    \def for a macro.

```

This test tells us in particular what kind of assignment may be applied to #1. Therefore it turns true for #1 being e.g. primitive T<sub>E</sub>X's skip registers and #2==\skip.

```

1866 {%
1872   \@tempswafalse
1874   \gmu@CASE x{\defNoHash#2}%
1875   {% case "def no hash" (hashless macro)
1876     \@tempwattrue
1877     \edef\gmu@IfIs@resa{%
1878       \pdef\@nx\gmu@IfIs@resa
1879       #####1\detokenize{macro:->}%
1880       #####2\@nx\@nil{\@ifnonempty{#####2}}%

```

And we prepare applying thus defined macro to #1:

```

1882     \unexpanded{\@xa\gmu@IfIs@resa\meaning#1}%
1883     \detokenize{macro:->}\@nx\@nil
1884   }% of \edef
1885   }% of case hasless macro ("def no hash")

```

if not hashless

```

1888   \gmu@CASE x {\defHashy#2}%
1889   {%

```

case hashy macro

```

1891     \@tempwattrue
1892     \edef\gmu@IfIs@resa{%
1893       \pdef\@nx\gmu@IfIs@resa
1894       #####1\detokenize{macro:}#####2\xiihash1#####3->%
1895       #####4\@nx\@nil{\@ifnonempty{#####4}}%

```

And we prepare applying thus defined macro to #1:

```

1897     \unexpanded{\@xa\gmu@IfIs@resa\meaning#1}%
1898     \detokenize{macro:}\xiihash1->\@nx\@nil

```

```

1899     }% of \edef
1900 }% of case hashy macro

1902 \gmu@CASE x {#2\boolean}
1903 {% case Boolean
1904     \@tempwattrue
1905     \def\gmu@IfIs@resa{\gmu@IfBooleanMacro#1}%
1907 }% of case Boolean

1909 \gmu@CASEstrany #2{\dimexpr \numexpr \glueexpr }%
1910 {% case  $\epsilon$ -TeX expression
1911     \@tempwattrue
1912     \def\gmu@IfIs@resa{% if should be a macro expanding to expression contents
1914         \IfIsExpression #1#2}%
1915 }% of case expression

```

The subsequent part of this huge test refers to `\meanings`. We want to make it robust to possible (although rather seldom) changes of `\escapechar`. Therefore define an aux macro that expands to detokenised escape char.

Assume that letters, including »e«, will not be used as the escape char:

```

1924 \gmu@geteschar

```

Now `\gmu@xiieschar` carries the detokenised escape char (is empty if `\escapechar < 0`).

```

1930 \gmu@CASE x {\dimen#2}%
1931 {% Case dimen. Let's check if it's special (inner TeX's) or normal.
1933     \gmu@ifxany#1{\prevdepth \pagegoal \pagetotal \pagestretch
1934         \pagefilstretch \pagefillstretch \pagefilllstretch
1935         \pageshrink \pagedepth
1936         \hfuzz \vfuzz \overfullrule \emergencystretch \hsize \vsize
1937         \maxdepth \splitmaxdepth \boxmaxdepth \lineskiplimit
1938         \delimitershortfall \nulldelimiterspace \scriptspace
1939         \mathsurround \predisplaysize \displaywidth
1940         \displayindent \parindent \hangindent \hoffset \voffset
1941     }%
1942     {\@tempwattrue\let\gmu@IfIs@resa\@firstoftwo}%
1943     {% subcase normal dimen
1944         \def\gmu@IfIs@resa{%
1945             \gmu@ifexempty {\gmu@xiieschar}%
1946             {%
1947                 \pdef\gmu@IfIs@resa####1####2####3####4####5####6\@nil{%
1948                     \gmu@if {}
1949                     {1%
1950                         \if d####1\else0\fi
1951                         \if i####2\else0\fi
1952                         \if m####3\else0\fi
1953                         \if e####4\else0\fi
1954                         \if n####5\else0\fi
1955                     }%
1956                 }% of inner \gmu@IfIs@resa...
1957             }% ...if eschar negative
1958             {%

```

```

1959         \pdef\gmu@IfIs@resa####1####2####3####4####5####6####7\@nil{%
%
1960         \gmu@if {} {1%
1961         \if \gmu@xiieschar ####1\else0\fi
1962         \if d####2\else0\fi
1963         \if i####3\else0\fi
1964         \if m####4\else0\fi
1965         \if e####5\else0\fi
1966         \if n####6\else0\fi
1967         1%
1968         }%
1969         }% of inner \gmu@IfIs@resa...
1970         }% ...when eschar nonnegative
1971         }% of outer \gmu@IfIs@resa
1972         }% of if special dimen or not
1973     }% of case dimen

1975 \gmu@CASE x {\count#2}%
1976 {% case count
1977     \gmu@ifxany#1{\spacefactor \prevgraf \deadcycles
1978     \insertpenalties
1979     \pretolerance \tolerance \hbadness \vbadness \linepenalty
1980     \hyphenpenalty \exhyphenpenalty \binoppenalty \relpenalty
1981     \clubpenalty \widowpenalty \displaywidowpenalty
1982     \brokenpenalty \predisplaypenalty \postdisplaypenalty
1983     \interlinepenalty \floatingpenalty \outputpenalty
1984     \doublehyphendemerits \finalhyphendemerits \adjdemerits
1985     \looseness \pausing \holdinginserts \tracingonline
1986     \tracingmacros \tracingstats \tracingparagraphs
1987     \tracingpages \tracingoutput \tracinglostchars
1988     \tracingcommands \tracingrestores \language \uchyph
1989     \lefthyphenmin \righthyphenmin \globaldefs
1990     \defaultshyphenchar \defaultskewchar \escapechar \endlinechar
1991     \newlinechar \maxdeadcycles \hangafter \fam \mag
1992     \delimiterfactor \time \day \month \year \showboxbreadth
1993     \showboxdepth \errorcontextlines
1994     \lastlinefit
1995     }%
1996     {% if special count register then:
1997     \@tempwatrue\let\gmu@IfIs@resa\@firstoftwo}%
1998     {%

if normal count register then

2000     \def\gmu@IfIs@resa{%
2001     \gmu@ifexempty \gmu@xiieschar
2002     {\pdef\gmu@IfIs@resa####1####2####3####4####5####6\@nil{%
2003     \gmu@if {} {1%
2004     \if c####1\else0\fi
2005     \if o####2\else0\fi
2006     \if u####3\else0\fi
2007     \if n####4\else0\fi
2008     \if t####5\else0\fi

```

```

2009         1%
2010         }% of test
2011         }% of inner \gmu@IfIs@resa...
2012         }% ... when eschar is negative
2014         {% eschar not empty (nonnegative)
2015
                \pdef\gmu@IfIs@resa####1####2####3####4####5####6####7\@nil{%
                %
2016         \gmu@if {} {1%
2017         \if \gmu@xiieschar####1\else0\fi
2018         \if c####2\else0\fi
2019         \if o####3\else0\fi
2020         \if u####4\else0\fi
2021         \if n####5\else0\fi
2022         \if t####6\else0\fi
2023         1%
2024         }% of test
2025         }% of inner \gmu@IfIs@resa...
2026         }% ... when eschar nonnegative
2027         }% of outer \gmu@IfIs@resa
2028         }% of if normal count register
2029         }% of case count

2031 \gmu@CASE x {\skip#2}%
2032 {% case skip
2033 \gmu@ifxany#1{%
2034 \baselineskip \lineskip \parskip \abovedisplayskip
2035 \abovedisplayshortskip \belowdisplayskip
2036 \belowdisplayshortskip \leftskip \rightskip \topskip
2037 \splittopskip \tabskip \spaceskip \xspaceskip \parfillskip
2038 }%
2039 {% if special skip
2040 \@tempwattrue\let\gmu@IfIs@resa\@firstoftwo}%
2041 {% not special skip
2042 \def\gmu@IfIs@resa{%
2043 \gmu@ifexempty \gmu@xiieschar
2044 {\pdef\gmu@IfIs@resa####1####2####3####4####5\@nil{%
2045 \gmu@if {} {1%
2046 \if s####1\else0\fi
2047 \if k####2\else0\fi
2048 \if i####3\else0\fi
2049 \if p####4\else0\fi
2050 1%
2051 }%
2052 }% of inner \gmu@IfIs@resa...
2053 }% ...when eschar nonnegative

2055 {\pdef\gmu@IfIs@resa####1####2####3####4####5####6\@nil{%
2056 \gmu@if {} {1%
2057 \if \gmu@xiieschar ####1\else0\fi
2058 \if s####2\else0\fi
2059 \if k####3\else0\fi
2060 \if i####4\else0\fi

```

```

2061         \if p###5\else0\fi
2062         1%
2063     }%
2064     }% of inner \gmu@IfIs@resa...
2065     }% ...when eschar nonnegative
2066     }% of outer \gmu@IfIs@resa
2067     }% of if skip but not special
2068 }% of case skip

2070 \gmu@CASE x {\toks#2}
2071 {% case toks
2072     \gmu@ifxany#1{%
2073         \output \everypar \everymath \everydisplay \everyhbox
2074         \everyvbox \everyjob \everycr \errhelp \everyeof}%
2075     {% subcase special toks
2076         \@tempwatrue\let\gmu@IfIs@resa\@firstoftwo}%
2077     {% subcase normal toks
2078         \def\gmu@IfIs@resa{%
2079             \gmu@ifexempty \gmu@xiieschar
2080             {\pdef\gmu@IfIs@resa####1####2####3####4####5\@nil{%
2081                 \gmu@if {} {1%
2082                     \if t###1\else0\fi
2083                     \if o###2\else0\fi
2084                     \if k###3\else0\fi
2085                     \if s###4\else0\fi
2086                     1%
2087                 }%
2088                 }% of inner \gmu@IfIs@resa...
2089                 }% ...when eschar negative
2090             {\pdef\gmu@IfIs@resa####1####2####3####4####5####6\@nil{%
2091                 \gmu@if {} {1%
2092                     \if \gmu@xiieschar ####1\else0\fi
2093                     \if t###2\else0\fi
2094                     \if o###3\else0\fi
2095                     \if k###4\else0\fi
2096                     \if s###5\else0\fi
2097                     1%
2098                 }%
2099                 }% of inner \gmu@IfIs@resa...
2100                 }% ...when eschar nonnegative
2101                 }% of outer \gmu@IfIs@resa
2102                 }% of if toks but not special
2103             }% of case toks
2105         \gmu@CASE x {\long#2}%

```

if a macro is `\long`, then these detokens open its meaning despite of possible `\outer`.

```

2108 {% case long macro
2109     \def\gmu@IfIs@resa{%
2110         \gmu@ifexempty \gmu@xiieschar
2111         {\pdef\gmu@IfIs@resa####1####2####3####4####5\@nil{%
2112             \gmu@if {} {1%
2113                 \if l###1\else0\fi
2114                 \if o###2\else0\fi

```

```

2115         \if n####3\else0\fi
2116         \if g####4\else0\fi
2117         1%
2118     }%
2119 }% of inner \gmu@IfIs@resa...
2120 }% ...when eschar negative
2121 {\pdef\gmu@IfIs@resa####1####2####3####4####5####6\@nil{%
2122     \gmu@if {} {1%
2123         \ifnum \gmu@xiieschar ####1\else0\fi
2124         \if l####2\else0\fi
2125         \if o####3\else0\fi
2126         \if n####4\else0\fi
2127         \if g####5\else0\fi
2128         1%
2129     }%
2130 }% of inner \gmu@IfIs@resa...
2131 }% ...when eschar nonnegative
2132 }% of outer \gmu@IfIs@resa
2133 }% of case \long Note that we also can put here a test whether the meaning begins
        with macro which would mean that a macro is short.
2137 \gmu@CASE x {\newif#2}%
2138 {% case \newif (Boolean switch)
2139     \def\gmu@IfIs@resa{%
2140         \gmu@ifexempty \gmu@xiieschar
2141         {\pdef\gmu@IfIs@resa####1####2####3%
2142             ####4####5####6####7####8\@nil{%
2143             \gmu@unless {} {0% lazy disjunction
2144                 \gmu@if {} {1% lazy conjunction
2145                     \if i####1\else0\fi
2146                     \if f####2\else0\fi
2147                     \if f####3\else0\fi
2148                     \if a####4\else0\fi
2149                     \if l####5\else0\fi
2150                     \if s####6\else0\fi
2151                     \if e####7\else0\fi
2152                     1%
2153                 }%
2154                 }{0}%
2155                 \gmu@if {} {1% lazy conjunction
2156                     \if i####1\else0\fi
2157                     \if f####2\else0\fi
2158                     \if t####3\else0\fi
2159                     \if r####4\else0\fi
2160                     \if u####5\else0\fi
2161                     \if e####6\else0\fi
2162                     \if \relax####7\else0\fi
2163                     1%
2164                 }%
2165                 }{0}%
2166                 0%
2167             }%
2168         }% of inner \gmu@IfIs@resa...
2169     }% ...when eschar nonnegative

```

```

2171 {\pdef\gmu@IfIs@resa####1####2####3%
2172   ####4####5####6####7####8####9\@nil{%
2173   \gmu@unless {} {0%
2174     \gmu@if {} {1%
2175       \if \gmu@xiieschar ####1\else0\fi
2176       \if i####2\else0\fi
2177       \if f####3\else0\fi
2178       \if f####4\else0\fi
2179       \if a####5\else0\fi
2180       \if l####6\else0\fi
2181       \if s####7\else0\fi
2182       \if e####8\else0\fi
2183       1%
2184     }%
2185   }{0}%
2186   \gmu@if {} {1%
2187     \if \gmu@xiieschar ####1\else0\fi
2188     \if i####2\else0\fi
2189     \if f####3\else0\fi
2190     \if t####4\else0\fi
2191     \if r####5\else0\fi
2192     \if u####6\else0\fi
2193     \if e####7\else0\fi
2194     \if \relax####8\else0\fi
2195     1%
2196   }%
2197   }{0}%
2198   0%
2199   }%
2200   }% of inner \gmu@IfIs@resa...
2201   }% ...when eschar nonnegative
2202   }% of outer \gmu@IfIs@resa
2203 }% of case \newif (Boolean switch)

2205 \gmu@CASE {x\@xa\@xa} {\csname if\endcsname#2}%
2206 {% case \if test
2207   \def\gmu@IfIs@resa{%
2208     \gmu@ifexempty \gmu@xiieschar
2209     {\pdef\gmu@IfIs@resa####1####2####3\@nil{%
2210       \gmu@if {} {1%
2211         \if i####1\else0\fi
2212         \if f####2\else0\fi
2213         1%
2214       }%
2215       }% of inner \gmu@IfIs@resa...
2216     }% ...when eschar negative

2218   {\pdef\gmu@IfIs@resa####1####2####3####4\@nil{%
2219     \gmu@if {} {1%
2220       \if \gmu@xiieschar ####1\else0\fi
2221       \if i####2\else0\fi
2222       \if f####3\else0\fi
2223       1%
2224     }%

```

```

2225     }% of inner \gmu@IfIs@resa...
2226     }% ...when eschar nonnegative
2227     }% of outer \gmu@IfIs@resa
2228 }% of case \if

2230 \gmu@lastCASE %
2231 {false}{}{}%
2232 \gmu@ESAC

```

For the cases 1– $n$  we just launch their auxiliary macro:

```

2234 \if@tempswa
2235   \@xa\gmu@IfIs@resa
2236 \else

```

For the other cases their auxiliary macro defines “inner @resa” (protected) which perform the test on the meaning of #1.

```

2240   \gmu@IfIs@resa

now (new) \gmu@IfIs@resa it's defined as delimited with #2 and \@nil (if #1 may be
one of kinds checked letter by letter).

2246   \edef\gmu@IfIs@resb{%
2247     \gmu@IfIs@resa\meaning#1%
2248     \relax\relax\relax\relax\relax\relax\relax\relax % to provide some-
        thing for gobbling tests.
2250     \@xa\detokenize\@xa{\string#2}\@nx\@nil}%
2251   \@xa\gmu@IfIs@resb
2252 \fi
2253 }% of \IfIs

2256 \unless\ifdefined\@tempskipa\newskip\@tempskipa\fi
2257 \unless\ifdefined\@tempmuskipa\newmuskip\@tempmuskipa\fi

2260 \long\def\IfIsExpression
2261 #1% the stuff to be examined
2262 #2% \dimexpr, \glueexpr, \numexpr or \muexpr
2263 {%
2264   \ifx#2\numexpr\let\next\@tempcnta\fi
2265   \ifx#2\glueexpr\let\next\@tempskipa\fi
2266   \ifx#2\dimexpr\let\next\@tempdima\fi
2267   \ifx#2\muexpr\let\next\@tempmuskipa\fi
2268   \afterassignment\gmu@testtopenalty
2269   \next=#2#1\penalty
2270 }

2272 \def\gmu@testtopenalty#1\penalty{%
2273   \gmu@ifempty{#1}}

```

### Global Boolean switches

The `\newgif` declaration’s effect is used even in the L<sup>A</sup>T<sub>E</sub>X 2<sub>ε</sub> source by redefining some particular user defined ifs (UD-ifs henceforth) step by step. The goal is to make the UD-if’s assignment global. I needed it at least twice during `gmdoc` writing so I make it a macro. It’s an almost verbatim copy of L<sup>A</sup>T<sub>E</sub>X’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 2287 \pdef\newgif#1{%
```



```

2288  {\escapechar\m@ne
2289   \global\let#1\iffalse
2290   \@gif#1\iftrue
2291   \@gif#1\iffalse
2292  }}

```

‘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.

```

2298  \def\@gif#1#2{%
2299   \protected\@xa\gdef\csname\@xa\@gobbletwo\string#1%
2300   g% the letter g for ‘\global’.
2301   \@xa\@gobbletwo\string#2\endcsname
2302   {\global\let#1#2}}

```

```

2304  \pdef\newif#1{% We not only make \newif \protected but also make it to define
        \protected assignments so that premature expansion doesn’t affect \if...\fi
        nesting.

```

```

2311  \count@\escapechar \escapechar\m@ne
2312  \let#1\iffalse
2313  \@if#1\iftrue
2314  \@if#1\iffalse
2315  \escapechar\count@}

```

```

2317  \def\@if#1#2{%
2318   \protected \@xa\def\csname\@xa\@gobbletwo\string#1%
2319   \@xa\@gobbletwo\string#2\endcsname
2320   {\let#1#2}}

```

```

2323  \pdef\hidden@iffalse{\iffalse}
2324  \pdef\hidden@iftrue{\iftrue}

```

After `\newgif\iffoo` you may type `{\foogtrue}` and the `\iffoo` switch becomes globally equal `\iftrue`. Simili modo `\foogfalse`. Note the letter *g* added to underline globalness of the assignment.

If for any reason, no matter how queer ;-) may it be, you need *both* global and local switchers of your `\if...`, declare it both with `\newif` and `\newgif`.

Note that it’s just a shorthand. `\global\ifswitch(true|false)` *does* work as expected.

There’s a trouble with `\refstepcounter`: defining `\@currentlabel` is local. So let’s `\def` a `\global` version of `\refstepcounter`.

Warning. I use it because of very special reasons in `gmdoc` and in general it is probably not a good idea to make `\refstepcounter` global since it is contrary to the original L<sup>A</sup>T<sub>E</sub>X approach.

```

\grefstepcounter 2345  \pdef\grefstepcounter#1{%
2346   {\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`:

```
2360 \pdef\hgrefstepcounter#1{%
2361   \ifhmode\leavevmode\fi\grefstepcounter{#1}}
```

By the way I read some lines from *The T<sub>E</sub>X book* 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

```
2368 \pdef\hunskip{\ifhmode\unskip\fi}
```

Note the two macros defined above are `\protected`. I think it's a good idea to make `\protected` all the macros that contain assignments. There is one more thing with `\ifhmode`: it can be different at the point of `\edef` and at the point of execution.

Another shorthand. It may decrease a number of `\expandafters` e.g.

```
\glet 2378 \def\glet{\global\let}
```

And for use in the very document,

```
\addtomacro 2394 \lpdef\addtomacro
2395 #1% macro (has to be parameterless)
2396 #2% stuff to be added
2397 {\edef #1{\@xau #1\unexpanded{#2}}}
```

We use unexpanded `edef` to allow `#` tokens in `#2`. Note that L<sup>A</sup>T<sub>E</sub>X's `\g@addto@macro` uses analogous trick from the pre- $\epsilon$ -T<sub>E</sub>X era (scratch toks register and `\edef`).

Note that `\addtomacro` loses the possible prefixes of the previous version of `#1` but may be prefixed on its own (anyway, what's the use of `\long` for a macro with no parameters? And who on Earth uses `\outer`?).

Note moreover it works fine also for `#1`'s that are `\protected` because `\expandafter` hits first and always works.

A `\global` version:

```
2412 \pdef\gaddtomacro{\global\addtomacro}
```

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

```
2416 \long\def\prependtomacro#1#2{%
2418   \edef#1{\unexpanded{#2}\@xa\unexpanded\@xa{#1}}}
```

Note that `\prependtomacro` can be prefixed.

```
\addtotoks 2422 \lpdef\addtotoks#1#2{%
2423   #1=\@xa{\the#1#2}}
```

```
\gmu@prependtoks@aux 2425 \newtoks\gmu@prependtoks@aux
```

```
2429 \lpdef\gmu@prependtotoks@ambig
```

```
2430 #1% scope
```

```
2431 #2% toks register
```

```
2432 #3% text of prependement
```

```
2433 {%
```

```
2437   \iffalse {\fi % hack to balance braces in definition
```

```
2438     \@XA{%
```

```
2439       #1#2=\bgroup#3%
```

```
2440     }% during execution, this brace closes the \@XA's argument...
```

```
2441     \the#2}% and this one closes the text opened by \bgroup, i.e., the text of \toks
assignment.
```

```
2447 }
```

```
\prependtotoks 2450 \lpdef\prependtotoks
```

the "loocal" version (2010/11/10)

```
2452 #1% toks register
2453 #2% text to be prepended
2454 {\gmu@prependtotoks@ambig {}{#1}{#2}}%
2457 \lpdef\gprependtotoks
```

the "global" version (2010/11/10)

```
2459 #1% toks register
2460 #2% text to be prepended
2461 {\gmu@prependtotoks@ambig \global{#1}{#2}}%
```

Adding an element to a list iterated by by \@for

```
\addto@forlist 2467 \long\def\addto@forlist
2468 #1% a comma-separated list
2469 #2% the element(s) added
2470 {\ifx#1\@empty
2471   \@xa\@gobble
2472   \else\@xa\@firstofone
2473   \fi
2474   {\addtomacro#1{,}}%
2475   \addtomacro#1{#2}}%
2476 }
```

```
2479 \lpdef\eaddtomacro
2480 #1% a macro
2481 #2% stuff to be added (will be fully expanded)
2482 {\edef#1{\@xau#1#2}}
```

### **\gmu@ifundefined—a test that doesn't create any hash entry unlike \@ifundefined**

I define it under another name not redefine \@ifundefined because I can imagine an odd case when something works thanks to \@ifundefined's 'relaxation' effect.

```
2492 \long\def\gmu@ifundefined #1{% not \protected because expandable.
2499   \gmu@CASEnot {cname} {#1\endcname}% defined...
2500   {\@firstoftwo}%
2502   \gmu@CASE {x\@xa\@xa} {\cname #1\endcname\relax}% but only as \re|
2504   {\@firstoftwo}%
```

defined and not \relax—then 3rd argument.

```
2506   \gmu@lastCASE
2507   {\@secondoftwo}%
2508   \gmu@ESAC
2509 }
```

```
2511 \long\def\gmu@ifdefined #1#2#3{%
2512   \gmu@ifundefined {#1}{#3}{#2}}
```

While \gmu@if(un)defined are intended for csnames and any macros present in their #1 are expanded, the next two are intended for #1 being a single CS or an active char.

It's the active char's case why we write all *da capo*.

```
2521 \long\def \gmu@ifCSdefined#1{%
```

```

2522 \gmu@CASEnot {defined} {#1}
2523 {\@secondoftwo}%
2525 \gmu@CASE x {\relax #1}
2526 {\@secondoftwo}%
2528 \gmu@EatCases
2529 {\@firstoftwo}%
2530 \gmu@ESAC
2531 }

2534 \long\def\CSNameIf
2535 #1% the name to check and probably execute

2 stuff when the CS#1 defined
3 stuff when the CS#1 not defined
2540 {%
2541 \ifcsname #1\endcsname
2542 \@xa\@twoofthree
2543 \else\@xa\@thirdofthree
2544 \fi
2545 {\csname #1\endcsname}%
2546 }

```

### Some ‘other’ and active stuff

Here I define a couple of macros expanding to special chars made ‘other’. It’s important the CS are expandable and therefore they can occur e.g. inside `\csname...\endcsname` unlike e.g. CS’es `\chardefed`.

```

2557 \foone{\catcode`\_ =8 }
\subs 2558 {\let\subs=_}

2561 \foone{\catcode`\^ =7 }
\sup 2562 {\let\sup =^}

2564 \foone{\@makeother\_}
2565 {\def\xiiunder{_}}

2567 \let\all@unders\xiiunder
2568 \foone{\catcode`\_ =8 }
2569 {\addtomacro\all@unders{_}}
2570 \foone{\catcode`\_ =11 }
2571 {\addtomacro\all@unders{_}}
2572 \foone{\catcode`\_ =13 }
2573 {\addtomacro\all@unders{_}}
Now \all@unders bears underscores of categories 8, 11, 12 and 13.

2577 \foone{\@makeother\^^M}{\def\xiiM{
2578 }}%

2580 \def\all@stars{*}
2581 \foone{\catcode`\* =11 }
2582 {\addtomacro\all@stars{*}}
2583 \foone{\catcode`\* =13 }
2584 {\addtomacro\all@stars{*}}
And \all@stars bears stars of categories 11, 12 and 13.

2588 \def\all@spaces{ }
2589 \foone{\@makeother\ }{%

```

```

2590 \addtomacro\all@spaces{ }%
2591 }
2592 \foone{\obeyspaces}{%
2593 \addtomacro\all@spaces{ }%
2594 }

2596 \foone\obeylines{%
2597   \def\two@Ms{
2598   }}
2599 \foone{\@makeother\^^M}{%
2600   \addtomacro\two@Ms{
2601   }}

2603 \edef\spaces@and@Ms{\@xau{\all@spaces}\@xau{\two@Ms}}

2605 \ifdefined\XeTeXversion
2606   \chardef\_="005F
2607   \fi

2609 \foone{\@makeother\`}{%
2610 {\def\backquote{`}}

2613 \foone{\catcode`\[=1 \@makeother\{
2614   \catcode`\]=2 \@makeother\}}%
2615 [%
2616   \def\xiilbrace[{}]{%
2617   \def\xiirbrace[]]{%
2618 ]% of \firstofone

```

Note that L<sup>A</sup>T<sub>E</sub>X's `\@charlb` and `\@charrb` are of catcode 11 ('letter'), cf. The L<sup>A</sup>T<sub>E</sub>X 2<sub>ε</sub> Source file k, lines 129–130.

Now, let's define such a smart `_` (underscore) which will be usual `_s` in the math mode and `_12` ('other') outside math.

```

2629 \foone{\catcode`\_=\active}
2630 {%
\smartunder 2631   \pdef\smartunder{%
2632     \catcode`\_=\active
2633     \def_{%
2634       \ifincsname\xiiunder
2635       \else
2636         \ifmmode\subs
2637         \else\xiiunder %
2638         \fi
2639       \fi}}}% We define it as \_ not just as \xiiunder because some font encodings
              don't have _ at the \char`\_ position.

2645 \foone{\catcode`\!=0
2646   \@makeother\!}
\xiibackslash 2647 {\!newcommand*!\xiibackslash{\!}}

\bslash 2651 \let\bslash=\xiibackslash

2655 \foone{\@makeother\%}
2656 {\def\xiipercent{}}

2659 \foone{\@makeother\&}{%
2660 {\def\xiiand{&}}

```

```

2662 \foone{\@makeother\ }%
2663 {\def\xiispace{ }}
2665 \foone{\@makeother\#}%
2666 {\def\xiihash{#}}

```

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 recognised the difference between `\xiispace` which has to be plain 'other' char (used in `\xiistring`) and something visible to be printed in any font.

```

2675 \AtBeginDocument{%
2676   \ifdefined\xxt@visiblespace
2677     \let\visiblespace\xxt@visiblespace
2678     \def\xxt@visiblespace@fallback{%
2679       \fontspec{Latin Modern Mono}\textvisiblespace}%
2680   \else
2681     \let\visiblespace\xiispace
2682   \fi}
2685 \foone\obeyspaces{\def\gmu@activespace{ }}
2687 \foone\obeylines{\def\activeM{^M}}
2691 \pdef\makeblanksignored{%
2692   \catcode`\^^M=9\relax
2693   \catcode`\^^I=9\relax
2694   \catcode`\ =9\relax}
2696 \pdef\fooblanksignored{%
2697   \foone{\makeblanksignored}%
2698 }

```

**`\@ifnextcat`, `\@ifnextac`, catcode-independent `\gm@ifstar`, `\@ifnextnotgroup`, `\@ifnextgroup`**

As you guess, we `\def \@ifnextcat` à la `\@ifnextchar`, see L<sup>A</sup>T<sub>E</sub>X<sub>2</sub> $\epsilon$  source dated 2003/12/01, file `d`, lines 253–271. The difference is in the kind of test used: while `\@ifnextchar` does `\ifx`, `\@ifnextcat` does `\ifcat` which means it looks not at the meaning of a token(s) but at their `\catcode`(s). As you (should) remember from *The T<sub>E</sub>X book*, the former test doesn't expand macros while the latter does. But in `\@ifnextcat` the peeked token is protected against expanding by `\noexpand`. Note that the first parameter is not protected and therefore it shall be expanded if it's expandable. Because an assignment is involved, you can't test whether the next token is an active char. But you can test if the next token is `{1 or }2`: `\@ifnextcat\bgroup...`, `\@ifnextcat\egroup...`

```

2717 \long\pdef\edefU#1#2{%

```

This is to allow passing the hashes.

```

2719   \edef#1{\unexpanded{#2}}%
2720 }

```

```

2723 \long\pdef\xdefU#1#2{%

```

This is to allow passing the hashes.

```

2725   \xdef#1{\unexpanded{#2}}%
2726 }

```

```

2729 \long\pdef\@ifnextcat#1#2#3{%
2732   \edefU\reserved@d{#1}%
2733   \edefU\reserved@a{#2}%
2734   \edefU\reserved@b{#3}%
2735   \futurelet\@let@token\@ifncat}

2737 \def\@ifncat{%
2738   \ifx\@let@token\@sptoken
2739     \let\reserved@c\@xifncat
2740   \else
2741     \ifcat\reserved@d\@nx\@let@token
2742       \let\reserved@c\reserved@a
2743     \else
2744       \let\reserved@c\reserved@b
2745     \fi
2746   \fi
2747   \reserved@c}

2749 {\def\:\{\let\@sptoken= } \global\: % this makes \@sptoken a space token.
2752 \def\:\{\@xifncat} \@xa\gdef\:\{\futurelet\@let@token\@ifncat}}

```

Note the trick to get a macro with no parameter and requiring a space after it. We do it inside a group not to spoil the general meaning of \: (which we extend later).

The next command provides the real \if test for the next token. *It* should be called \@ifnextchar but that name is assigned for the future \ifx text, as we know. Therefore we call it \@ifnextif.

Having \@ifnextcat defined, let's apply it immediately. Similar thing does probably the `xspace` package.

```

2766 \pdef\spifletter{\@ifnextcat a{\space}{}}
2769 \long\pdef\@ifnextif#1#2#3{%
    (the future token in \noexpanded, unlike #1)

2776   \def\reserved@d{#1}%
2777   \def\reserved@a{#2}%
2778   \def\reserved@b{#3}%
2779   \futurelet\@let@token\@ifnif}

\@ifnif 2782 \def\@ifnif{%
2783   \ifx\@let@token\@sptoken
2784     \let\reserved@c\@xifnif
2785   \else
2786     \if\reserved@d\@nx\@let@token
2787       \let\reserved@c\reserved@a
2788     \else % #1 of \@ifnextif is not \if-equivalent the future token. But this may
           be because the future token is active so it would be \if-equivalent if not
           passed through \futurelet. Let's manage this case.

2792     \begingroup
2793     \edef\gmu@tempa{%
2794       \lccode`\@nx~=`\reserved@d
2795     }\gmu@tempa
2796     \lowercase{\endgroup
2797     \ifx~}\@let@token

```

```

2798     \let\reserved@c\reserved@a
2799     \else
2800     \let\reserved@c\reserved@b
2801     \fi
2802 \fi
2803 \fi
2804 \reserved@c}

2807 {\def\:\{\let\@sptoken= } \: % this makes \@sptoken a space token.
2809 \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.

```

2820 \long\pdef\@ifnextac#1#2{%
2821 \@ifnextnotgroup
2822 {\gmu@ifnac{#1}{#2}}%
2823 {#2}}

2825 \long\def\gmu@ifnac#1#2#3{%
2826 \ifcat\@nx~\@nx#3%
2827 \@xa\@firstoftwo
2828 \else\@xa\@secondoftwo
2829 \fi{#1#3}{#2#3}}

```

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` not just recatcode it to `_1`?)

Having defined such tools, let's redefine `\gmu@ifstar` to make it work with whatever catcode `*` may be. make a version of `\gmu@ifstar` that would work with `*11`.

```

2843 \pdef\gmu@lowstar{%
2844 \iffontchar\font"22C6 \char"22C6 \else\gmu@lowstarfake\fi
2845 }% we could define it to be expandable (with ^^22c6) but the only goal of it would
    be allowing this low star in \csname...\endcsname. But it would be misleading
    (not everyone can easily distinguish * from *) so IMHO it's better to raise an
    error should such an active occur in a \csname. This macro is intended to be
    \let to an active star only in verbatims. For usual text there's

```

Commands around making star low (via `\activeation`) are defined in line 13111 because they use `\DeclareCommand`.

```

2858 \def\gmu@tempa{*}
2859 \foone{\catcode`*\=\active}
2860 {\def\gmu@tempb{*}% it's defined in line ?? to make * defined (when it was undefined,
    % \newcommand's \gm@ifstar test turned true, the next undefined token was
    % gobbled and raised an error).
2864 \let*=\gmu@lowstar}

2867 \edef\gmu@tempa{%
2868 \long\pdef\@nx\gmu@ifstar##1##2{%
2871 \@nx\@ifnextif\gmu@tempa%
2872 {\@nx\@firstoftwo{##1}}}% it's a bit hacky but O.K.: if the condition is not
    satisfied, the following brace is taken

```



```

2874     {%
2875     \@nx\@ifnextchar\@xa\@nx\gmu@tempb
2876     {\@nx\@firstoftwo{##1}}% and again, if the condition is not satisfied, the
           second brace is taken.
2878     {##2}%
2879     }% of if not *12
2880     }% of \gmu@ifstar.
2881 }\gmu@tempa

```

A test whether we can pick a single token. We have to check whether we are not next to { and whether we are not next to }.

```

2889 \long\pdef\@ifnextnotgroup#1#2{% This macro checks whether the next token is
           able to be picked or is it a braced list of tokens or is it a group closer so there's
           no token to be picked.
2893 \@ifnextcat\bgroup{#2}{%
2894 \@ifnextcat\egroup{#2}%
2896 {#1}}}}
2898 \long\pdef\@ifnextgroup#1#2{% Note this macro turns true both before a group
           opener and before a group closer.
2900 \@ifnextnotgroup{#2}{#1}}

```

now let's apply this to sth. useful (used in **gmverse** and in some typesetting, e.g. for prof. JSB).

```

2905 \pdef\ignoreactiveM{%
2906 \@ifnextgroup{}{\gmu@checkM}%
2907 }
2909 \foone\obeylines{% we know it's a single token since we use this macro only in \@
           ifnextgroup's 'else'.
2911 \long\pdef\gmu@checkM#1{%
2912 \ifx
2913 #1\@xa\ignoreactiveM%
2914 \else\@xa#1\fi %
2915 }%
2916 }

```

Now, define a test that checks whether the next token is a genuine space, <sub>10</sub> that is. First define a CS let such a space. The assignment needs a little trick (*The T<sub>E</sub>X book* appendix D) since \let's syntax includes one optional space after =.

```

2925 \let\gmu@reserveda\*%
2926 \def\*{%
2927 \let\*\gmu@reserveda
2928 \let\gmu@letspace= }%
2929 \* %

```

*The T<sub>E</sub>X book* chapter 8, 10th double bend says, if we read thoroughly (or meet this perversity in our daily T<sub>E</sub>Xing), that a blank line (<sup>^^M^^M</sup> of catcode 5, the two subsequent chars of catcode 5 (preceded with sth. else)) are transformed in \par—a blank space (cat. 10) followed by \par. This strange case we *don't* want to treat as 'next is space', as T<sub>E</sub>X itself doesn't (a letter CS gobbles such a space).

```

2939 \lpdef\gmu@peep@next #1{%

```

This macro performs \futurelet to \@let@token, checks if we are in this strange case of a blank space before \par (and fixes \@let@token if so), moreover, if \@let@token turns

out to be undefined or `\relax`, it grabs the next token (thus surely single and not `\outer`) and passes as the contents of `\@def@token` (which is un-defined each time).

Therefore #1 for it may be branched with `\ifdefined\@def@token`.

```

2950 \let\@def@token\@undefined
2952 \edefU\gmu@peepnext@inner {%
2953   \gmu@CASE x {\@let@token\gmu@letspace}%
2954   {\@ifnextchar\par
2955     {#1}
2956     {\let\@let@token= \gmu@letspace % note = and blank space
2957       \@XA{#1}\space
2958     }%
2959   }% of if \futurelet detected a blank space

2961   \gmu@CASE {AnyClause} % if any of the conditions below:
2962   {{ x {\@let@token\@undefined}
2963     x {\@let@token\relax}}} % it's arg of disjunction then we grab the next
2964     token into the \@def@token macro.
2965   {\gmu@peep@hash{#1}}%
2966   \gmu@lastCASE
2967   {#1}%
2968   \gmu@ESAC
2969   }%
2970 \futurelet\@let@token\gmu@peepnext@inner
2971 }

2974 \lpdef\gmu@peep@hash
2975 #1% stuff that probably tests #2 somehow
2976 #2% a single token, which we are sure is undefined or \relax (and thus not \outer
2977     neither a blank space)
2978 {%
2979   \def\@def@token{#2}%
2980   #1%
2981   #2%
2982 }

2985 \lpdef\gmu@ifpeeped
2986 #1% kind of comparison
2987 #2% stuff to compare
2988 {%
2989   \gmu@if {defined} {\@def@token}
2990   {\@XA{\gmu@if {#1}}\@xa{\@def@token #2}}
2991   {\gmu@if {#1} {\@let@token #2}}%
2992 }

@ifnextspace 2996 \long\pdef\@ifnextspace
2997 #1% if yes
2998 #2% if not
2999 {%

Note that this macro doesn't gobble space(s)

3001 \gmu@peep@next
3002 {\gmu@if x {\@let@token\gmu@letspace}{#1}{#2}}%
3003 }

```

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.

The next—in the `gmdoc` bundle not to gobble spaces following `%`, which is crucial for determining whether there is a `DocStrip` directive or not. But this is done with a wrapper for both `\ifnextspace` and `\ifnextchar`:

“if next char” that respects spaces

```
\ifnextcharRS 3015 \long\pdef\ifnextcharRS
3016 #1% a single token (will be \ifxed)
3017 #2% what if yes
3018 #3% what if not
3019 {%
3020   \gmu@peep@next
3021   {\gmu@if {defined} {\@def@token}
3022     {\@XA{\gmu@if {StrX}}\@xa {\@def@token #1}}% of the special “null” case
3025     {\gmu@if x {\@let@token #1}}%
3026     {#2}{#3}%
3027   }% of \gmu@peep@next
3028 }
```

“if next any” that respects spaces

```
\ifnextanyRS 3033 \long\pdef\ifnextanyRS
3034 #1% list of tokens (any balanced text, will be \let token after token
3035 #2% what if next is one of listed #1
3036 #3% what if not
3037 {\gmu@peep@next
3038   {\gmu@if {defined} {\@def@token}
3039   {\@xa \gmu@ifStrXany \@def@token }
3041   {\gmu@ifxany {\@let@token}}%
3042   {#1}{#2}{#3}%
3043   }% of peep
3044 }
```

Some (quite large) chunks of commented out code were here till v0.240.

```
3048 \long\def\gmu@ifnextStrXany
3049
3050 We call this macro only in the true branch of \ifx\@let@token#1
3051 #1% outer macro’s tested list of tokens
3052 #2% “if found” branch
3053 #3% “if not found” branch
3054 {\gmu@notif {AnyClause}
3055   {{ x {\@let@token\@undefined} x {\@let@token\relax}}}%
3056   {#2}%
```

Some (i.e. *both*) `\@undefined` or `\relax`, then we compare strings

```
3060   {\gmu@futureifany {#2}{#3}{#1}}%
3061 }
3063 \long\def\gmu@futureifany
```

As above.

3067 #1% what if OK  
 3068 #2% what if not OK  
 3069 #3% list of tokens  
 3070 #4% token ot be checked against #3 (and put back on the input) Note we use this macro  
       only where we know #4 is either undefined or `\relax`.  
 3073 `\gmu@ifstrany #4 {#3}{#1}{#2}#4`

“if next any” that ignores space(s)

```
\@ifnextanyIS 3078 \long\pdef\@ifnextanyIS
3079 #1% list of tokens (any balanced text, will be \let token after token
3080 #2% what if next is one of listed in #1
3081 #3% what if not
3082 {%
3083   \edefU\gmu@ifna@afterlet{%
3084     \gmu@if x{\@let@token\gmu@letspace}%
3085     {% if next is blank space, we drop it and retry:
3086       \edefU\gmu@ifna@resa{\@ifnextanyIS{#1}{#2}{#3}}%
3087       \afterassignment\gmu@ifna@resa
3088       \let\gmu@drain=
3089     }%
3090     {% else we perform \@ifnextanyIS
3091       \gmu@ifxany{\@let@token}{#1}{%
3092         \gmu@ifnextStrXany {#1}{#2}{#3}%
3093       }%
3094       {#3}%
3095     }%
3096   }% of the after-let macro
3097   \futurelet\@let@token\gmu@ifna@afterlet
3098 }% of \@ifnextanyIS
```

```
\@ifnextnoneRS 3102 \long\pdef\@ifnextnoneRS
3103 #1% list of tokens (any balanced text, will be \let token after token
3104 #2% what if next is one of listed #1
3105 #3% what if not
3106 {%
3107   \@ifnextanyRS{#1}{#3}{#2}%
3108 }
```

If-next-group respecting spaces

```
3112 \long\pdef\@ifnextgroupRS #1#2{%
        Note that this macro doesn't gobble space(s)
3114   \gmu@peep@next
3115   {\gmu@if x{\@let@token\bgroup}{#1}%
3116    {\gmu@if x{\@let@token\egroup}{#1}{#2}}%
3117   }%
3118 }
3122 \long\pdef\@ifnextnotgroupRS#1#2{%
3123   \@ifnextgroupRS{#2}{#1}}
```

What seems worth noticing is that my if-nexts don't use `\reserved@{a|...|d}` and set `\@let@token` properly.

Now a test if the next token is an active line end. I use it in **gmdoc** and later in this package for active long dashes.

```
3133 \foone\obeylines{%
3134   \long\pdef\@ifnextMac#1#2{%
3135     \@ifnextchar^^M{#1}{#2}}}
```

Standard `\string` command returns a string of ‘other’ chars except for the space, for which it returns `_`. In **gmdoc** I needed the spaces in macros’ and environments’ names to be always `_`, so I define

```
\xiistring 3145 \long\def\xiistring#1{%
3146   \if\@nx#1\xiispace
3147     \xiispace
3148   \else
3149     \afterfi{\string#1}% to make the same error as bare \string would cause in
                          case of empty #1.
3151   \fi}
```

The next macro is applied to a `\detokenized` nonempty string to convert the spaces into ‘other’.

```
3155 \def\@xiispaces#1 #2\@nil{%
3156   #1%
3157   \ifx\@xiispaces#2\@xiispaces
3158   \else
3159   \xiispace
3160   \afterfi{\@xiispaces#2\@nil}%
3161   \fi}
```

```
3163 \long\pdef\xiiEdetoke
3164 #1% a scratch CS
3165 #2% stuff to be fully expanded and turned to catcode 12 including spaces.
3166 {%
3167   \edef#1{#2}%
3168   \edef#1{%
3169     \@xa\@xa\@xa\@xiispaces
3170     \@xa\detokenize\@xa{#1} \@nil}%
3171   }
3172 }
```

```
3175 \long\def\@ifEUnextchar#1#2#3{%
    ‘if Edefed Unexpanded next char’
3177   \let\reserved@d=#1%
3178   \edefU\reserved@a{#2}%
3179   \edefU\reserved@b{#3}%
3180   \futurelet\@let@token\@ifnch}
```

### Storing and restoring the catcodes of specials

```
\gmu@storespecialchars 3187 \newcommand\gmu@storespecialchars[1][[]]{% we provide a possibility of adding stuff.
                          For usage see line ??.
```

```
3189   \def\do##1{\catcode`\@nx##1=\the\catcode`##1\relax}%
\gmu@restorespecials 3190   \edef\gmu@restorespecials{%
3191     \dospecials\do^^M#1}

3193 \pdef\gmu@septify{% restoring the standard catcodes of specials. The name is the
                          opposite of ‘sanitize’ :-). It restores also the original catcode of ^^M.
```

```

3196 \def\do{\relax\catcode`}%
3197 \do\ 10\do\0\do\{1\do\}2\do\3\do\&4%
3198 \do\#6\do\^7\do\_8\do\%14\do\~13\do\^^M5\relax
%% \let\do\@makeother
%% \do0\do1\do2\do3\do4\do5\do6\do7\do8\do9\relax
3201 }

```

## Storing and restoring the meanings of CSes

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

```

\ifgmu@SMglobal 3208 \newif\ifgmu@SMglobal
3210 \pdef\SMglobal{\gmu@SMglobaltrue}
3212 \def\MakePrivateLetters{\makeatletter}

```

The subsequent commands are defined in such a way that you can ‘prefix’ them with `\SMglobal` to get global (re)storing.

A command to store the current meaning of a CS in another macro to temporarily redefine the CS and be able to set its original meaning back (when grouping is not recommended):

```

\StoreMacro 3224 \pdef\StoreMacro{%
3225 \begingroup\MakePrivateLetters
3226 \gmu@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 3455.

```

3231 \lpdef\egStore@Macro#1{\endgroup\Store@Macro{#1}}
3232 \lpdef\egStore@MacroSt#1{\endgroup\Store@MacroSt{#1}}
3234 \pdef\StoreMacro@nocat

```

It’s version of `\StoreMacro` to be used in arguments of other macros, where recatcoding wouldn’t change anything anyway.

```

3237 {%
3238 \gmu@ifstar \Store@MacroSt \Store@Macro
3239 }
3242 \def\gmu@storeprefix{/gmu/store}
3245 \lpdef\Store@Macro#1{%
3246 \escapechar92
3247 \ifgmu@SMglobal\afterfi\global\fi
3248 \@xa\let\csname \gmu@storeprefix/\bslash@or@ac{#1}\endcsname#1%
3249 \global\gmu@SMglobalfalse
3250 }
3253 \lpdef\Store@MacroSt#1{%
3254 \edef\gmu@smtempa{%
3255 \ifgmu@SMglobal\@xa\global\fi
3256 \let\@xa\@nx\csname\gmu@storeprefix/\bslash@or@ac{
#1}\@xa\endcsname% we add backslash because to ensure compatibility
between \ (Re)StoreMacro and \ (Re)StoreMacro*, that is. to allow

```

```

        writing e.g. \StoreMacro\kitten and then \RestoreMacro*{kitten} to
        restore the meaning of \kitten.
3262   \ifcsname #1\endcsname % If the argument CS is undefined, we undefine the
        storage macro too.
3267   \@xa\@nx\csname#1\endcsname
3268   \else\@nx\@undefined
3269   \fi
3270   \global\gmu@SMglobalfalse}% we wish the globality to be just once.

3272   \gmu@smtempa
3273 }

```

We make the `\StoreMacro` command a three-step to allow usage of the most inner macro also in the next command.

The starred version, `\StoreMacro*` works with csnames (without the backslash). It's first used to store the meanings of robust commands, when you may need to store not only `\foo`, but also `\csname foo \endcsname`.

The next command iterates over a list of CSeS and stores each of them. The CS'es may be separated with commas but they don't have to.

```

\StoreMacros 3288 \lpdef\StoreMacros{\begingroup\MakePrivateLetters\egStore@Macros}
3290 \lpdef\egStore@Macros#1{\endgroup
3291   \Store@Macros{#1}%
3292 }

3294 \lpdef\Store@Macros #1{%
3295   \gmu@setsetSMglobal
3296   \let\gml@StoreCS\Store@Macro
3297   \gml@storemacros#1.}

3300 \def\gmu@setsetSMglobal{%
3301   \ifgmu@SMglobal
3302     \let\gmu@setSMglobal\gmu@SMglobaltrue
3303   \else
3304     \let\gmu@setSMglobal\gmu@SMglobalfalse
3305   \fi}

```

And the inner iterating macro:

```

3308 \lpdef\gml@storemacros#1{%
3309   \def\gmu@storemacros@resa{\@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.
3312   \if\gmu@storemacros@resa.% a dot finishes storing.
3313     \global\gmu@SMglobalfalse
3314   \else
3315     \if\gmu@storemacros@resa,% The list this macro is put before may contain
        commas and that's O.K., we just continue the work.
3317     \afterfifi\gml@storemacros
3318   \else% what is else this shall be stored.
3319     \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.
3322     \afterfifi{\gmu@setSMglobal\gml@storemacros}%
3323     \fi
3324   \fi
3325 }

```

And for the restoring

```
\RestoreMacro 3331 \lpdef\RestoreMacro{%
3332
           \begingroup\MakePrivateLetters\gmu@ifstar\egRestore@MacroSt\egRestore@Macro
3334 \lpdef\egRestore@Macro#1{\endgroup\Restore@Macro{#1}}
3335 \lpdef\egRestore@MacroSt#1{\endgroup\Restore@MacroSt{#1}}
3337 \lpdef\Restore@Macro#1{%
3338   \escapechar92
3339   \gmu@ifstored#1{%
3340     \ifgmu@SMglobal\afterfi\global\fi
3341     \@xa\let\@xa#1\csname \gmu@storeprefix/\bslash@or@ac{#1}\endcsname
3342     \global\gmu@SMglobalfalse}%
3343   {\unless\ifgmu@quiet
3344     \PackageWarning{gmutils}{\@nx#1 is not stored, I do nothing with
3345       it}%
3346     \fi
3347   }%
3348 }
3350 \long\def\gmu@ifstored#1#2#3{%
3351   \gmu@ifundefined{\gmu@storeprefix/
3352     \bslash@or@ac{#1}}{#3}{#2}%
3353 }
3355 \lpdef\gmu@storeifnotyet#1{%
3356   \gmu@if {} {\relax\@nx#1}% we check if it's a CS
3357   {\gmu@ifstored{#1}}{\StoreMacro@nocat#1}}%
3358   {}%
3359 }
3362 \lpdef\Restore@MacroSt#1{%
3363   \gmu@ifundefined{\gmu@storeprefix/\bslash@or@ac{#1}}%
3364   {\unless\ifgmu@quiet
3365     \PackageWarning{gmutils}{\bslash#1 is not stored. I~do nothing}%
3366     \fi}%
3367   {\edef\gmu@smtempa{%
3368     \ifgmu@SMglobal\global\fi
3369     \@nx\let\@xa\@nx\csname#1\endcsname
3370     \@xa\@nx\csname\gmu@storeprefix/\bslash@or@ac{#1}\endcsname}% cf. the
           commentary in line 3256.
3372     \gmu@smtempa}%
3373   \global\gmu@SMglobalfalse
3374 }
\RestoreMacros 3377 \lpdef\RestoreMacros{%
3378   \begingroup\MakePrivateLetters
3379   \egRestore@Macros}
3381 \lpdef\egRestore@Macros#1{\endgroup
3382   \Restore@Macros{#1}%
3383 }
3385 \lpdef\Restore@Macros #1{%
3386   \gmu@setsetSMglobal
```



```

3387 \let\gml@storeCS\Restore@Macro% we direct the core CS towards restoring and
      call the same iterating macro as in line 3297.
3390 \gml@storemacros#1.)

```

As you see, the `\RestoreMacros` command uses the same iterating macro inside, it only changes the meaning of the core macro.

```

3395 \pdef\ResetMacro{% restore possibly to \@undefined
3397
      \begingroup\MakePrivateLetters\gmu@ifstar\egReset@MacroSt\egReset@Macro}
3399 \lpdef\egReset@Macro#1{\endgroup\Reset@Macro{#1}}
3400 \lpdef\egReset@MacroSt#1{\endgroup\Reset@MacroSt{#1}}
3402 \lpdef\Reset@Macro#1{%
3403   \escapechar92
3404   \ifgmu@SMglobal\@xa\global\fi
3405   \ifcsname \gmu@storeprefix/\backslash@or@ac{#1}\endcsname
3406     \@xa\let\@xa#1\csname \gmu@storeprefix/\backslash@or@ac{#1}\endcsname
3407   \else
3408     \let#1\@undefined
3409   \fi
3410   \global\gmu@SMglobalfalse
3411 }%
3414 \lpdef\Reset@MacroSt#1{%
3415   \ifcsname \gmu@storeprefix/\backslash@or@ac{#1}\endcsname
3416     \ifgmu@SMglobal\@xa\global\fi
3417     \@xa\let\csname#1\@xa\endcsname
3418     \csname\gmu@storeprefix/\backslash@or@ac{#1}\endcsname % cf. the commen-
      tary in line 3256.
3420   \else
3421     \@xa\let\csname#1\endcsname\@undefined
3422   \fi
3423   \global\gmu@SMglobalfalse
3424 }

```

```

\ResetMacros 3427 \lpdef\ResetMacros{\begingroup\MakePrivateLetters\Reset@Macros}
3429 \lpdef\Reset@Macros#1{\endgroup
3430   \gmu@setsetSMglobal
3431   \let\gml@storeCS\Reset@Macro% we direct the core CS towards restoring and call
      the same iterating macro as in line 3297.
3434   \gml@storemacros#1.)

```

As you see, the `\ResetMacros` command uses the same iterating macro inside, it only changes the meaning of the core macro.

And to restore *and* use immediately:

```

3441 \pdef\StoredMacro{\begingroup\MakePrivateLetters\Stored@Macro}
3442 \lpdef\Stored@Macro#1{\endgroup\Restore@Macro#1#1}

```

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

```

3445 \long\def\storedcsname#1{%
3446   \ifcsname \gmu@storeprefix/\backslash@or@ac{#1}\endcsname
3447     \afterfi{%
3448       \csname \gmu@storeprefix/\backslash@or@ac{#1}\endcsname}%

```

```

3449 \else \@xa \@undefined
3450 \fi
3451 }

```

2008/08/03 we need to store also an environment.

```

3455 \pdef\StoreEnvironment#1{%
3457 \Store@MacroSt{#1}\Store@MacroSt{end#1}}

3459 \pdef\RestoreEnvironment#1{%
3461 \Restore@MacroSt{#1}\Restore@MacroSt{end#1}}

```

It happened (see the definition of `\@docinclude` in `gmdoc.sty`) that I needed to `\relax` a bunch of macros and restore them after some time. Because the macros were rather numerous and I wanted the code more readable, I wanted to `\do` them. After a proper defining of `\do` of course. So here is this proper definition of `\do`, provided as a macro (a declaration).

```

\StoringAndRelaxingDo 3476 \pdef\StoringAndRelaxingDo{%
3477 \gmu@SMdo@setscope
3478 \long\def\do##1{%
3479 \gmu@SMdo@scope
3480 \@xa\let\csname \gmu@storeprefix/\bslash@or@ac{##1}\endcsname##1%
3481 \gmu@SMdo@scope\let##1\relax}}

3483 \pdef\gmu@SMdo@setscope{%
3484 \ifgmu@SMglobal\let\gmu@SMdo@scope\global
3485 \else\let\gmu@SMdo@scope\relax
3486 \fi
3487 \global\gmu@SMglobalfalse
3488 }

```

And here is the counter-definition for restore.

```

\RestoringDo 3497 \lpdef\RestoringDo{%
3498 \gmu@SMdo@setscope
3499 \long\def\do##1{%
3500 \gmu@SMdo@scope
3501 \@xa\let\@xa##1\csname
3502 \gmu@storeprefix/\bslash@or@ac{##1}\endcsname}%
3503 }

```

Note that both `\StoringAndRelaxingDo` and `\RestoringDo` are sensitive to the `\SM|global` ‘prefix’.

(Preliminary:)

```

3509 \pdef\gmu@MakeScopePrefix
3510 #1% CS to be let \global or \relax
3511 #2% a sequence of tokens
3512 {%
3513 \let#1\relax
3514 \gmu@ifxany{\global}{#2}%
3515 {\let#1\global}}%
3516 }

```

And to store a cs as explicitly named cs, i.e. to `\let` one csname another (`\n@melet` not `\@namelet` because the latter is defined in Till Tantau’s `beamer` class another way) (both arguments should be text):

```

3523 \lpdef\gmu@namelet
3524 #1% scope prefix (to be honest, any sequence of tokens that may be passed as an
      argument: \gmu@ifxany will parse it)
3526 #2% left side of the assignment
3527 #3% right side of the assignment
3528 {%
3529   \gmu@MakeScopePrefix\gmu@namelet@scpref{#1}%
3530   \gmu@if {csname} {#3\endcsname}%
3531   {%
3532     \@xa\gmu@namelet@scpref\@xa\let\csname#2\@xa\endcsname
3533     \csname#3\endcsname}%
3534   {%
3535     \@xa\gmu@namelet@scpref
3536     \@xa\let\csname#2\endcsname\@undefined}%
3537 }

```

```

3540 \pdef\n@melet{\gmu@namelet\relax}
3552 \pdef\gn@melet{\gmu@namelet\global}

```

```

3554 \long\pdef\tri@let
3555 #1% scope prefix(es)
3556 #2% left side
3557 #3% right side of the assignment
3558 {%
3559   \gmu@MakeScopePrefix\gmu@tri@let@scpref{#1}%

```

both s.o.a. can be names, CS es or active chars.

```

3561   \ifcat\@nx~\@nx#2%
3562     \def\next{\gmu@tri@let@scpref\let#2}%
3563   \else \edef\next{\gmu@tri@let@scpref\let\@xanxtri{#2}}%
3564   \fi

```

Thus the left argument of the assignment is handled and of the assignment prepared.

```

3567   \ifcat\@nx~\@nx#3%
3568     \next#3%
3569   \else
3570     \edef\next{%
3571       \@xau\next
3572       \ifcsname \strip@bslash{#3}\endcsname
3573       \@xanxtri{#3}%
3574       \else\@nx\@undefined
3575       \fi
3576     }% of next's edef
3577     \next
3578   \fi
3579 }%

```

```

3583 \long\pdef\envirlet#1#2{% for \letting environments.
3584   \n@melet{#1}{#2}%
3585   \n@melet{end#1}{end#2}%
3586 }

```

```

3588 \long\pdef\glenvirlet#1#2{% for \letting environments.
3589   \gn@melet{#1}{#2}%

```

```
3590 \gn@melet{end#1}{end#2}%
3591 }
```

\@ifprevenvir are defined in **gmenvir**

### Setting for X<sub>Y</sub>TeX

```
3598 \def\@ifXeTeX{% two-argument command
3599 \ifdefined\XeTeXversion
3600 \unless\ifx\XeTeXversion\relax\afterfifi\@firstoftwo\else\afterfifi\@secondoftwo
3601 \else\afterfi\@secondoftwo\fi
3602 }
```

```
\ifgmuXeTeX 3608 \newif\ifgmuXeTeX
3609 \@ifXeTeX{\gmuXeTeXtrue}{}%
```

\XeTeXthree is defined with \DeclareCommand so occurs yet in **gmcommand**.

```
3616 \@ifXeTeX{%
3617 \pdef\textbullet{%
3620 \iffontchar\font"2022 \char"2022 \else\ensuremath{\bullet}\fi}%
3622 \pprovide\glyphname#1{%
3624 \XeTeXglyph \numexpr\XeTeXglyphindex "#1"\relax\relax}% since XYTeX
... \numexpr is redundant.
3626 }
3627 {\def\textbullet{\ensuremath{\bullet}}}}
3629 \def\if@XeTeX {\@ifXeTeX {\iftrue}{\iffalse}}
```

### Expandable turning stuff all into ‘other’

A shorthand. Note that it takes an undelimited argument not requires *<balanced text>*.

```
3638 \long\def\detoken@xa#1{\detokenize\@xa{#1}}
```

The next macro originates from the ancient era when I didn’t know about  $\epsilon$ -TeX’s \detokenize. A try to redefine it to \detokenize\@xa{#1} resulted in error so (v0.991) I leave it and use as is.

Note however it acts different than \detoken@xa for a macro with parameters: while \detoken@xa produces and ‘extra ’ error, \all@other expands to the detokenised meaning.

```
\all@other 3648 \long\def\all@other#1{\@xa\gmu@gobmacro\meaning#1}
```

The \gmu@gobmacro macro above is applied to gobble the \meaning’s beginning, long macro:-> all ‘other’ that is.

```
3653 \edef\gmu@tempa{%
\gmu@gobmacro 3654 \def\@nx\gmu@gobmacro##1\@xa\@gobble\string\macro:##2->{}}
3655 \gmu@tempa
```

### Show must go on

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,

```
3665 \def\gmshowlists{%
```

```

3666 \tracingonline=1
3667 \showboxdepth=1 \showboxbreadth=1000000 \showlists}

3670 \def\gmshowbox{%
3671 \tracingonline=1
3672 \showboxdepth=10000 \showboxbreadth=10000 \showbox}

3674 \def\gmtracingoutput{%
3675 \tracingoutput\@ne
3676 \tracingonline=\@ne
3677 \showboxdepth=1
3678 \showboxbreadth=1000000
3679 }

\ifgmu@debug@msgs 3682 \newif\ifgmu@debug@msgs

3685 \def\gmtron{%
3686 \tracingonline=\@M
3687 \tracingmacros=\@M
3688 \tracingassigns=\@M
3689 \tracingcommands=\@m
3690 \gmu@debug@msgstrue
3691 \let\let\let
3692 }

3694 \def\gmtroff{%
3695 \tracingonline=\m@ne
3696 \tracingmacros\m@ne
3697 \tracingassigns=\m@ne
3698 \tracingcommands=\m@ne
3699 \tracingoutput=\m@ne
3700 \gmu@debug@msgsfalse
3701 \let\let\let
3702 }

\nameshow 3705 \newcommand\nameshow[1]{%
3706 \ifcsname #1\endcsname
3707 \@xa\show\csname#1\endcsname
3708 \else \show\@undefined
3709 \fi}

\nameshowthe 3711 \newcommand\nameshowthe[1]{%
3712 \ifcsname #1\endcsname
3713 \@xa\showthe\csname#1\endcsname
3714 \else \showthe\@undefined
3715 \fi}

Note that to get proper \showthe\my@dimen14 in the ‘other’ @’s scope you write
\nameshowthe{my@dimen}14.

\namemeaning 3719 \newcommand\namemeaning[1]{%
3720 \ifcsname #1\endcsname
3721 \@xa\typeout{\@xa\meaning\csname#1\endcsname}%
3722 \show\relax
3723 \else \typeout{\meaning\@undefined}\show\relax
3724 \fi}

Note that to get proper \showthe\my@dimen14 in the ‘other’ @’s scope you write
\nameshowthe{my@dimen}14.

```

## Second class document class

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`.

```
3736 \def\secondclass{%
\ifSecondClass 3737   \newif\ifSecondClass
3738   \SecondClasstrue
3739   \@fileswithoptions\@clsextension}% [outeroff,gmeometric]{gmdocc} it's load-
      ing gmdocc.cls with all the bells and whistles except the error message.

3744 \AtBeginDocument{%
\@parindent 3745   \unless\ifdefined\@parindent
3746     \newskip\@parindent
3747     \@parindent=\@parindent
3748   \fi
3749 }

3754 \def\balsmiley#1 {}% to balance parentheses and brackets in smileys. ;-) \balsmiley(
      % ;-).

3759 \long\def\scantnoline#1{% ‘rescan tokens without adding line at the end’
3760   {\endlinechar\m@ne\scantokens{#1}}}

3765 \pdef\getprevdepth{% to pass last depth through a group (e.g. \end{envir.})
3766   \endgraf
3767   \xdef\setprevdepth{\prevdepth=\the\prevdepth\relax}%
3768 }

3771 \pdef\getprevdepthlocal{%
3772   \endgraf
3773   \edef\setprevdepth{\prevdepth=\the\prevdepth\relax}%
3774 }
```

## Storing the catcode of line end

```
3778 \def\StoreCatM{%
3779   \protected\edef\RestoreCatM{%
3780     \catcode`\@nx^^M=\the\catcode`\^^M\relax}%
3781 }

3783 \pdef\RestoreCatM{\PackageE{gmutils}{first store the catcode of
3784   ^\empty^\empty M with \string\StoreCatM.}%
3785 }
```

## \resizegraphics

```
3790 \RequirePackage{graphicx}

3792 \pdef\resizegraphics#1#2#3{% 2009/11/17 works bad with a file whose name con-
      tains spaces so I return \XeTeXpicfile
3797   \resizebox{#1}{#2}{%
3798     \edef\gmu@tempa{\@nx\csname XeTeX\@nx\@ifendswithpdf{%
3799       \@xa\string\csname#3\endcsname}{pdf}{pic}file\@nx\endcsname}%
3800     \gmu@tempa "#3"\relax}}

3802 \edef\gmu@tempa{%
```

```

3803 \def\@nx\@ifendswithpdf##1{%
3804   \unexpanded{%
3805     \ifnum
3806     \if\relax\gmu@pdfdetector}##1%
3807   \detokenize{pdf}\unexpanded{\relax0\else1\fi}% we expand to 1 if #1 ends
      with lowercase 'pdf' of cat. 12
3809   \unexpanded{\if\relax\gmu@PDFdetector}##1%
3810   \detokenize{PDF}\unexpanded{\relax0\else1\fi}% we expand to 1 if #1 ends
      with uppercase 'PDF' of cat. 12
3812   >0
3813   \unexpanded{\@xa\@firstoftwo\else\@xa\@secondoftwo\fi}%
3814 }% of \@ifendswithpdf

3818 \def\@nx\gmu@pdfdetector##1\detokenize{pdf}{}%
3819 \def\@nx\gmu@PDFdetector##1\detokenize{PDF}{}%
3820 }\gmu@tempa

```

Paragraph with last or first line centered: `\lastcentered` declaration and `lastcentered` environment

```

3826 \def\lastcentered{%
3827   \lastlinefit\z@
3828   \parindent0sp\relax
3829   \leftskip\dimexpr1\leftskip\relax plus 1fil\relax
3830   \rightskip\dimexpr1\rightskip\relax plus -1fil\relax
3831   \parfillskip0sp plus 2fil\relax
3832 }

3834 \def\endlastcentered{\par\@endptrue}

3837 \def\firstcentered{%
3838   \lastlinefit\z@
3839   \parfillskip 0sp\relax
3840   \rightskip0sp plus 1fil\relax
3841   \leftskip0sp plus -1fil\relax
3842   \parindent 0sp
3843   \addtotoks\everypar{\hskip0sp plus 2fil\relax}%
3844 }

3846 \let\endfirstcentered\endlastcentered

3849 \def\gmu@measurewd#1{%
3850   \edef\gmu@tempa{\the\fontcharwd\font`#1}%
3851   \settowidth{\@tempdimb}{% to preserve kerning
3852     \char`#1\char`#1\char`#1\char`#1\char`#1\char`#1%
3853     \char`#1\char`#1\char`#1\char`#1\char`#1\char`#1%
3854     \char`#1\char`#1\char`#1\char`#1\char`#1\char`#1%
3855     \char`#1\char`#1\char`#1}%
3856   \edef\gmu@tempb{\the\dimexpr(\@tempdimb-\gmu@tempa)/20}%
3857 }

3860 \def\@xa@three#1#2{% reverses expansion of three tokens, two given as arguments.
      (third may be {})
3863   \@xa\@xa\@xa \@xa\@xa\@xa \@xa #1%
3864   \@xa\@xa\@xa #2%
3865   \@xa }

```

```

3868 \def\@xa@four#1#2#3{% reverses expansion of four tokens, three given as arguments.
      (fourth may be {})
3871 \@xa\@xa\@xa\@xa \@xa\@xa \@xa
3872 \@xa\@xa\@xa\@xa \@xa\@xa\@xa\@xa #1%
3873 \@xa\@xa\@xa\@xa \@xa\@xa \@xa #2
3874 \@xa\@xa\@xa #3%
3875 \@xa }

```

### Comparison of detokenised strings

A (not expandable) macro that checks whether current environment is as given in #1. Why is this macro `\long`?—you may ask. It’s `\long` to allow environments such as `\string%` `\par`.

```

3883 \long\pdef\@ifenvir#1{%
      % #1 enquired environment name which will be confronted with \@currenvir
      % #2 what if true (if the names are equivalent2)
      % #3 what if false
3896 \gmu@ifedetokens{\@currenvir}{#1}%
3897 }

```

```

3900 \pdef\@ifjobname#1{\gmu@ifedetokens{\jobname}{#1}}

```

(2010/09/23, v0.993:) since `\gmu@ifedetokens` turned to be expandable, we make this macro `\protected`

2010/09/23 introduced as a common part of the three then four then... of the below.

```

3906 \long\def\gmu@ifstrcmp
3907 #1% wrapper for left side of comparison
3908 #2% wrapper for right side of comparison
3909 #3% left side of comparison (list of tokens)
3910 #4% right —”—
3911 {%
3912 \ifnum\strcmp{#1{#3}}{#2{#4}}=\z@
3913 \@xa\@firstoftwo
3914 \else
3915 \@xa\@secondoftwo
3916 \fi
3917 }

```

```

3919 \def\gmu@ifdetokens
      test if two list of tokens agree when decategorised (without expansion)
      implicit #1: left tokens
      implicit #2: right tokens
      (implicit #3: if agree)
      (implicit #4: if disagree
3927 {%
3928 \gmu@ifstrcmp\detokenize\detokenize
3929 }% of \gmu@ifdetokens

```

```

3932 \def\gmu@ifutokens
      test if two list of tokens agree when decategorised (without expansion)

```

---

<sup>2</sup> The names are checked whether they produce the same `\csname`. They don’t have to have the same catcodes.



```

3936 {%
3937 \gmu@ifstrcmp\unexpanded\unexpanded
3938 }

```

```

3940 \def\gmu@ifedetokens{%

```

basically a wrapper for the basic IMHO use of `\strcmp`. Won't work the same in some extremely perverse cases I can imagine. But with `\@firstofone` won't work either, only another way.

```

3945 \gmu@ifstrcmp \@firstofone \@firstofone
3946 }

```

And the `\protected` versions of those macros.

```

3949 \@XA{\pdef\gmu@pifdetokens}\@xa{\gmu@ifdetokens}
3951 \@XA{\pdef\gmu@pifutokens}\@xa{\gmu@ifutokens}
3953 \@XA{\pdef\gmu@pifedetokens}\@xa{\gmu@ifedetokens}

```

(2010/09/23, v0.993:) redefined deeply and made expandable thanks to `\strcmp`. Also renamed from `\gmu@ifedetokens` (the `\gmu` prefix added)

```

%% \lpdef\@ifedetokens#1#2{%
%% %
%% % #1 first list of tokens to be expanded and detokenized
%% % #2 second list
%% % #3 if agree
%% % #4 else
%% %
%% \edef\gmu@edetoka{#1}% to get #1 fully expanded.
%% \edef\gmu@edetokb{\@xa\detokenize\@xa{\gmu@edetoka}}% with our
%% % brave new \begin, \@currenvir is fully expanded,
%% % remember?
%% \edef\gmu@edetoka{#2}% to get #2 fully expanded.
%% \edef\gmu@edetokc{\@xa\detokenize\@xa{\gmu@edetoka}}%
%% \ifx\gmu@edetokb\gmu@edetokc\@xa\@firstoftwo
%% \else\@xa\@secondoftwo
%% \fi
%% }

```

## Hashes for meta-defining macros

In this section we use an expandable loop described in The  $\varepsilon$ -TeX Manual p. 9. In the `gmampulex` package we construct a more general definer for such loops.

```

3996 \def\gmu@HHashes#1#2{% this is a fully expandable loop analogous to that of The
    \varepsilon-TeX Manual p. 9.
3998 \ifnum#1<#2 %
3999 #####\number#1
4000 \expandafter\gmu@HHashes
4001 \expandafter{\number\numexpr#1+1\expandafter}%
4002 \expandafter{\number#2\expandafter}%
4003 \fi}% of \gmu@HHashes.

4005 \def\gmu@Hashes#1#2{% this is a fully expandable loop analogous to that of The
    \varepsilon-TeX Manual p. 9. expanding in an \edef to #####1...#####h.$2-1$ (quadruple
    hashes' sequence)

```

```

4008 \ifnum#1<#2 %
4009 #####\number#1
4010 \expandafter\gmu@HHashes
4011 \expandafter{\number\numexpr#1+1\expandafter}%
4012 \expandafter{\number#2\expandafter}%
4013 \fi}% of \gmu@hashes.

4015 \def\gmu@hashes#1#2{% this is a loop analogous to that of The  $\epsilon$ -TeX Manual p. 9.,
    that expands to a sequence of double hashes  $\langle\#1\rangle-\langle\#2-1\rangle$ , useful in edefing a
    definition of macros.

4020 \ifnum#1<#2%
4021 ####\number#1
4022 \expandafter\gmu@hashes
4023 \expandafter{\number\numexpr#1+1\expandafter}%
4024 \expandafter{\number#2\expandafter}%
4025 \fi
4026 }% of \gmu@hashes.

4030 \def\gmu@HHashesbraced#1#2{%
4031 \ifnum#1<#2%
4032 {#####\number#1}%
4033 \expandafter\gmu@HHashesbraced
4034 \expandafter{\number\numexpr#1+1\expandafter}%
4035 \expandafter{\number#2\expandafter}%
4036 \fi}% of \gmu@hashesbraced.

4038 \def\gmu@Hashesbraced#1#2{%
4039 \ifnum#1<#2%
4040 {#####\number#1}%
4041 \expandafter\gmu@HHashesbraced
4042 \expandafter{\number\numexpr#1+1\expandafter}%
4043 \expandafter{\number#2\expandafter}%
4044 \fi}% of \gmu@hashesbraced.

4047 \def\gmu@hashesbraced#1#2{%
4048 \ifnum#1<#2%
4049 {####\number#1}%
4050 \expandafter\gmu@hashesbraced
4051 \expandafter{\number\numexpr#1+1\expandafter}%
4052 \expandafter{\number#2\expandafter}%
4053 \fi
4054 }% of \gmu@hashesbraced.

4057 \def\gmu@hashesOut#1#2{%
4058 \ifnum#1<#2%
4059 \space\space\space\space
4060 »\@nx\unexpanded{####\number#1}«%
4061 \expandafter\gmu@hashesOut
4062 \expandafter{\number\numexpr#1+1\expandafter}%
4063 \expandafter{\number#2\expandafter}%
4064 \fi
4065 }% of \gmu@hashesbraced.

4068 \def\gmu@hashesOutU#1#2{%
4069 \ifnum#1<#2%

```

```

4070 \space\space\space\space
4071 »\@nx\unexpanded{###\number#1}«%
4072 \expandafter\gmu@hashesOut
4073 \expandafter{\number\numexpr#1+1\expandafter}%
4074 \expandafter{\number#2\expandafter}%
4075 \fi
4076 }% of \gmu@hashesbraced.

4083 \@tempcnta=1
4084 \@whilenum\@tempcnta<11\do{% 2010/4/15
4085 \@nameedef{gmu@hashes@\the\numexpr\@tempcnta-1\relax}%
4086 {\gmu@hashes1\@tempcnta}%
4088 \@nameedef{gmu@Hashes@\the\numexpr\@tempcnta-1\relax}%
4089 {\gmu@Hashes1\@tempcnta}%
4091 \@nameedef{gmu@HHashes@\the\numexpr\@tempcnta-1\relax}%
4092 {\gmu@HHashes1\@tempcnta}%
4094 \@nameedef{gmu@hashesbraced@\the\numexpr\@tempcnta-1\relax}%
4095 {\gmu@hashesbraced1\@tempcnta}%
4097 \@nameedef{gmu@Hashesbraced@\the\numexpr\@tempcnta-1\relax}%
4098 {\gmu@Hashesbraced1\@tempcnta}%
4100 \@nameedef{gmu@HHashesbraced@\the\numexpr\@tempcnta-1\relax}%
4101 {\gmu@HHashesbraced1\@tempcnta}%
4103 \edef\gmu@eloops@resa{%
4104 \long\def\@xanxcs{%
4105 gmu@TOhashes@\romannumeral\numexpr\@tempcnta-1\relax}%
4106 \gmu@hashes1\@tempcnta{%
4107 \@nx\TypeOut{%
4110 \gmu@hashesOut %
4111 1 \@tempcnta}%
4112 }% of \gmu@TOhashes@viii etc.

4120 }% of temporary macro

4122 \gmu@eloops@resa

```

Generates nine pairs of macros `\gmu@TOhashes@i`, `\gmu@TOUhashes@i`, `\gmu@TOhashes@ii`, `\gmu@TOUhashes@ii` etc. that print (type out) their arguments on the terminal, unexpanded

```

4129 \advance\@tempcnta\@ne
4130 }% of \@whilenum

```

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 `\activate` the space but also will (re)define it as the `\` primitive. So define `\gmobeyspaces` that obeys this requirement.

(This definition is repeated in `gmverb`.)

```

4143 \foone{\catcode`\ \active}%
\gmobeyspaces 4144 {\newcommand*\gmobeyspaces{\let \ \catcode`\ \active}}

```

And a macro to forbid hyphenation of the next word:

```

\nohy 4148 \newcommand*\nohy{\leavevmode\kern0sp\relax}
\yeshy 4149 \newcommand*\yeshy{\leavevmode\penalty\@M\hskip\z@skip}

```

In both of the above definitions ‘0sp’ not `\z@` to allow their writing to and reading from files where @ is ‘other’.

```

4154 \long\pdef\gmu@EdefCurrnames#1{%
4155   \xiiEdetoke\gmu@EdefCurrnames@resa{#1}%
4156   \@xa \gmu@EdefCurrnames@ \gmu@EdefCurrnames@resa.tex.\@nil

    if no extension is present, tex is assumed. This couple of macros won't work well for
    files with dots in their names.

4159 }

4161 \pdef\gmu@EdefCurrnames@ #1.#2.#3\@nil{%
4162   \def\@currname{#1}%
4163   \def\@currentx{#2}%
4164 }

4167 \def\NamedInput@prepare#1{% we wrap in a macro to use also in \DocInput
4168   \@pushfilename
4169   \gmu@EdefCurrnames {#1}%
4170 }

4172 \let\NamedInput@finish=\@popfilename

4174 \pdef\NamedInput#1{% useful e.g. in error handling
4175   \NamedInput@prepare {#1}%
4176   \@@input #1\relax
4177   \NamedInput@finish
4178 }

4182 \def\gmu@ifdim
4183 #1% dimen specification
4184 #2% comparison
4185 #3% dimen specification
4186 {%
4187   \ifnum0\ifx#2≤\fi\ifx#2≥\fi\ifx#2≠\fi=\@ne
4188   \afterfi\unless
4189   \fi
4190   \ifdim#1\ifx#2≤\fi\ifx#2≥\fi\ifx#2≠\fi
4191   \ifx#2>\fi\ifx#2<\fi\ifx#2==\fi
4192   \dimexpr(#3)*1\relax% parentheses are for closing all possible  $\epsilon$ -TeX expressions
      not to gobble that \relax by them but only by the outermost \dimexpr to
      avoid premature expansion of the following \expandafter. (2010/6/14)
4197   \@xa\@firstoftwo
4198   \else\@xa\@secondoftwo
4199   \fi
4200 }

4203 \def\gmu@ifskip
4204 #1% glue specification
4205 #2% comparison for natural part
4206 #3% comparison for stretch part
4207 #4% comparison for shrink part
4208 #5% glue specification #6 (implicit) what if all conditions satisfied
      #7 (implicit) what if any condition unsatisfied.
4211 {\ifnum
4212   0\gmu@ifdim{1\glueexpr#1}#2{1\glueexpr#5}10%
```

```

4213     \gmu@ifdim{\gluestretch\glueexpr#1}#3{\gluestretch\glueexpr#5}10%
4214     \gmu@ifdim{\glueshrink\glueexpr#1}#4{\glueshrink\glueexpr#5}10=111
4215     \@xa\@firstoftwo
4216     \else\@xa\@secondoftwo
4217     \fi
4218 }% of \gmu@ifskip.

```

```

4222 \def\gmu@ifbox

```

We provide an expandable macro for comparing boxes' dimensions. We handle the registers' numbers not any boxes just to allow expandability.

```

4226 #1% a box register number (e.g. \copy\z@)
4227 #2% comparison for heights
4228 #3% comparison for depths
4229 #4% comparison for widths
4230 #5% a box register number
4231 {%
4232     \gmu@ifskip
4233     {\glueexpr (\ht#1 plus \dp#1 minus\wd#1 )*1\relax}%
4234     #2#3#4%
4235     {\glueexpr (\ht#5 plus \dp#5 minus\wd#5)*1\relax}%
4236 }
4240 \def\greater@dim#1#2{%
4241     \ifdim\dimexpr#1>\dimexpr(#2)*1\relax
4242         #1%
4243     \else #2%
4244     \fi
4245 }

```

Removal of an element from a comma-separated list macro (such as used in the L<sup>A</sup>T<sub>E</sub>X's \@for loops). The removed element becomes macro-meaning of #3.

```

4251 \long\pdef\gmu@removeelement
4252 #1% element to be removed
4253 #2% macro carrying a comma-separated list
4254 #3% CS to carry removed element.
4255 {%
4256     \let#3\@undefined%
4257     \def\gmu@removeelement@resa##1,#1,##2\@nil{%
4258         \gmu@ifempty{##2}%
4259         {\edefU#2{##1}}%
4260         {\edefU#2{##1,##2}}%
4261         \def#3{#1}%
4262         \@xa\gmu@removeelement@resa#2\@nil
4263     }%

```

If ##2 is not empty, then we know #1 was in the list so we have to remove its copy from the end of the list.

```

4266     }% of \gmu@removeelement@resa
4267     \@xa\gmu@removeelement@resa\@xa,#2,#1,\@nil

```

We gobble the beginning comma

```

4269     \@xa\@xa\@xa@ifx\@xa\@firstofmany#2\relax\@nil,%
4270     \edef#2{\unexpanded

```

```

4271     \@xa\@xa\@xa{\@xa\@gobble#2}}%
4272     \fi
4273 }

```

A macro for edefs of csnames: expands to a csname hit by `\noexpand`.

```

4276 \long\def\@nxcn#1{%
4277   \@xa\@nx\csname #1\endcsname}

4291 \def\@listbegvskipping{%
4292   \@topsepadd=\topsep
4293   \ifvmode
4294     \advance\@topsepadd by\partopsep
4295   \fi
4296   \par
4297   \addvspace\@topsepadd
4298 }

```

Remember that proper vskips at the end of an environment will be put by `\@endparenv`

```

4303 \def\foolc#1#2{%
4304   \begingroup\lccode`#1=`#2\relax
4305   \lcfirstofone
4306 }

4308 \long\def\lcfirstofone#1{%
4309   \lowercase{\endgroup#1}%
4310 }

```

Just to make it `\long`

```

4313 \long\def\PackageWarning#1#2{%
4314   \GenericWarning{%
4315     (#1)\@spaces\@spaces\@spaces\@spaces
4316   }{%
4317     Package #1 Warning: #2%
4318   }%
4319 }

```

long version of `\typeout` (`\par` occurs quite often)

```

4322 \long\def\TypeOut#1{%
4323   \edef\TO@resa{#1}%
4324   \edef\TO@resa{\@xa\detokenize\@xa{\TO@resa}}%
4325   \typeout{\TO@resa}}

4327 \long\def\ShowOut#1{%
4328   \TypeOut{#1}%
4329   \show\TO@resa
4330 }

```

A definition that makes its `#1` a stringed version of itself if in `\csname... \endcsname`

```

4336 \long\pdef\incsddef
4337 #1% a CS or active char
4338 #2% parameters string
4339 #3% definition's body
4340 {%
4341   \def#1#2{\gmu@ifincsdname {\string#1}{#3}}%
4342 }

```

## Deducing whether hash was braced

Since not built-in  $\TeX$ 's test can distinguish  $\{_1$  from  $\backslash\mathbf{group}$ , there seemed not to be a way to deduce whether the stuff we are passed as an argument was braced or not.

In general it still seems so to me, but in the case of undelimited arguments a partial solution seems to exist: count the tokens and assume they were braced if they are they and don't bother if they are just one.

In fact, this partial solution seems quite satisfactory to avoid bracing single tokens when passing them further as arguments.

```
4360 \gmu@DefSymbol\gmu@CountTokens@end
\c@gmu@TokensCount 4361 \newcount\c@gmu@TokensCount
4363 \lpdef\gmu@CountTokens
```

```
%% #1% upper bound? No.
```

Imposing numeric upper bound doesn't make sense since anyway we have to  $\backslash\mathbf{let}$  each token to balance the braces and/or not to bother with possibly unbalanced ifs:

If we'd stop at reaching the bound, we could throw the tail of tokens to nonexistence by putting a macro with a parameter delimited with the counting's sentinel. But that wouldn't work for unbalanced braces. On the other hand, wrapping the tail in an  $\backslash\mathbf{iffalse}$ , would release us from bothering with unbalanced braces, but in such case unbalanced ifs could occur so both ways are not satisfactory.

Note moreover that  $\{_1$  is indistinguishable from  $\backslash\mathbf{group}$  so we can't count braces' nesting to put as many as needed.

Counting groups and opening a junk box with this many groups is absurd because would lead to execution of all those tokens and that's what we cannot allow. BTW it'd be prone to unbalanced ifs, too.

```
4385 #1% the tokens to be counted.
4386 {%
4387   \c@gmu@TokensCount=\m@ne
4388   \let\gmu@CountToken@token\@undefined
4389   \gmu@CountToken@iter
4390   #1\gmu@CountTokens@end
4391 }
4393 \def\gmu@CountToken@iter{%
4394   \gmu@if x{\gmu@CountToken@token\gmu@CountTokens@end}%
4395   }% we've reached the end of iteration
4396   {%
```

We increase the counter and throw the iterator after next assignment

```
4398   \advance\c@gmu@TokensCount\@ne
4399   \afterassignment\gmu@CountToken@iter
4400   \let\gmu@CountToken@token=
4401   }%
4402 }%
```

Now we are ready to define a brace-wrapper:

```
4406 \long\def\gmu@passbraced
```

This macro checks whether its  $\#2$  was a balanced text (in explicit braces) or a  $\backslash\mathbf{group}$  token. Well, actually it checks whether  $\#2$  consists of not one token (0 or  $> 1$ ) and if so, wraps it in braces before putting it right next to  $\#1$ .

```
4412 #1% the stuff to be put before #2
```

```

4413 #2% the stuff we check and pass unbraced if single or braced otherwise
4414 {%
4415   \gmu@CountTokens{#2}%
4416   \gmu@if {num}{\c@gmu@TokensCount=\@ne}%

```

If we have only one token, we have to check whether it's space or not: the 'blank space' token could never become a hash without braces.

```

4421   {\gmu@if x{#2 }% if #2 is single token of blank space then we don't consider it
         single since it can't be an argument to a macro if without braces.
4424     \@secondoftwo\@firstoftwo
4425   }%

```

Otherwise (not one token, incl. the possibility of 0 tokens)

```

4427   \@secondoftwo
4428   {#1#2}{#1{{#2}}}%
4429 }% of \gmu@passbraced Note that this works also for #2 being empty: it will be
         considered not single and passed in braces.

```

```

4434 \long\def\gmu@passbracedNotSp

```

This macro works as the above except it doesn't embrace a blank.

```

4436 #1% the stuff to be put before #2
4437 #2% the stuff we check and pass unbraced if single or braced otherwise
4438 {%
4439   \gmu@CountTokens{#2}%
4440   \gmu@if {num}{\c@gmu@TokensCount=\@ne}%
4441   {#1{#2}}%
4442   {#1{{#2}}}%
4443 }% of \gmu@passbracedNotSp

```

```

4449 \long\def\MeaningOrUnex#1{%
4450   \gmu@if {singletoken}{{#1}}%
4451   {\meaning#1}{\unexpanded{#1}}%
4452 }

```

```

4455 \pldef\@iwru#1{%

```

as simple as possible not to put much output on the terminal if tracing is on.

```

4460   \immediate \write \@unused {1.\the\inputlineno:\space #1}%
4461 }

```

```

4463 \pldef\@iwruJ{\@iwru{^^J}}

```

```

4465 \pldef\@iwrum#1{%
4466   \@iwru{»}\unexpanded{#1}« is »\MeaningOrUnex{#1}«}%
4467 }

```

```

4469 \pldef\@iwruU #1{\@iwru{\unexpanded{#1}}}

```

```

4471 \pldef\@iwruif#1{%
4472   \gmu@if {gmu@debug@msgs}{}
4473   {\@iwru{#1}}{}%
4474 }

```

```

4478 \long\pdef\IgnInfo
4479 #1% package name
4480 #2% description

```



```

4481 #3% stuff we announce as ignored
4482 {%
4483   \PackageInfo{#1}{Item »\unexpanded{#3}« (\MeaningOrUnex{#3})
         ignored^^J%
4484   #2}%
4485 }

```

```

4488 \pldef\@iwma{%
      (Added 2010/10/19)
      Note it takes a text not an argument.

```

```

4492   \immediate \write \@mainaux
4493 }

```

```

4497 \def\stepnummacro
4498 #1% a macro that expands to some numerical stuff
4499 #2%
4500 {\edef#1{\the\numexpr #1+#2}}

```

An expandable macro that expands to `\numexpr` containing conversion of given sequence of switches to a binary number, presented in its Horner's schema.

```

\boolstobin 4507 \def\boolstobin
4508 #1% a sequence of Boolean switches' names without »if«
4509 {%
4510   \numexpr \boolstobin@iter 0 #1 {} \gmu@delim %
4511 }

```

```

4513 \def\boolstobin@iter
4514 #1% expression so far
4515 #2% the name of current switch (without »if«)
4516 #3% tail of switches
4517 \gmu@delim
4518 {%
4519   \gmu@ifempty{#3}%
4520   {#1\relax}% \relax to close the num expression
4521   {\boolstobin@iter
4522     {(#1)*2+\csname if#2\endcsname 1\else 0\fi}%
4523     #3\gmu@delim
4524   }%
4525 }

```

```

4528 \long\def\condstobin
4529 #1% a sequence of conditionals' names without »if« followed by the condition
4530 {%
4531   \numexpr \condstobin@iter 0 #1 {}{} \gmu@delim %
4532 }

```

```

4534 \long\def\condstobin@iter
4535 #1% expression so far
4536 #2% the name of current conditional (without »if«)
4537 #3% the condition for #2
4538 #4% tail of switches
4539 \gmu@delim
4540 {%
4541   \gmu@ifempty{#3}%

```

```

4542  {#1\relax}% \relax to close the num expression
4543  {\condstobin@iter
4544    {(#1)*2+\csname if#2\endcsname #31\else 0\fi}%
4545    #4\gmu@delim
4546  }%
4547  }
4562  \long\def\gmu@ifQUANT@iter
      (it's left-to-right and lazy)
4565  #1% the (binary) value that terminates calculation: 0 for AND and 1 for OR (it has to
      be a single token due to \expandafter in line 4583)
4568  #2% the (binary) value so far;
4569  #3% the name of current conditional (without »if«)
4570  #4% the condition for #2
4571  #5% tail of condition(al)s
4572  \gmu@delim
4573  {%
4574    \gmu@ifempty{#5}%
      if #5 is empty, we expand to the most recent (previous) value.
4576    {#2}%
4577    {% or else we check whether #2 is terminating
4578      \gmu@if {num}{#2=#1 }% with a space and expand to it and finish calculation if
      so...
4580      {#1}%
      or iterate. This case happens where the value so far is #1, so the future of the conjunc-
      tion doesn't depend on it.
4583      {\@xa\gmu@ifQUANT@iter \@xa #1%
4584        \the\numexpr %
          %% 1 * % \numexpr is used here to get the full expansion in one step. No need of
          superposing with identity.
4586          (\gmu@if {#3}{#4 } {1} {0}) +\z@\relax
4587          #5\gmu@delim
4588        }%
4589      }%
4590  }% of \gmu@ifQUANT@iter.
      Disjunction of conditions (finite Existential Quantifier). First as a pseudo-conditional.
      It expands to an open \if but that's OK if we use it only inside macros or with \gmu@if.
4598  \long\def \ifAnyClause
4599  #1{%
4600    \ifnum
4601      \gmu@ifQUANT@iter
4602      1% for the Existential Quantifier 1 terminates calculation (an(y) example has
      just been found)
4604      0% To make any calculation sense we assume 0 at the beginning (i.e., “we haven't
      found an example yet”)
4606      #1
4607      {false}{} {false} {}% the sentinel(s)
4608      \gmu@delim % the delimiter

```

```

4609     =\@ne % right side of \ifnum
4610 }

```

```

4613 \long\def\gmu@OR
4614 #1% as above

```

```

%% #2% (impl.) True branch
%% #3% (impl.) False branch

```

```

4617 {%
4618   \ifAnyClause {#1}%
4619     \@xa\@firstoftwo
4620   \else
4621     \@xa\@secondoftwo
4622   \fi
4623 }

```

Now the dual, i.e., conjunction of conditions. First as a pseudo-conditional (as above)

```

4626 \long\def\ifAllClauses
4627 #1% a sequence of pairs {<conditionals' name without »if«>}{<the condition>}. (Don't
      delimit the conditions, it's been taken care of!) (it has to be braced).

```

```

4631 {%
4632   \ifnum
4633     \gmu@ifQUANT@iter
4634     0% for the General Quantifier 0 terminates calculation (when a counter-example
      has just been found)
4635     1% To make any calculation sense we assume 1 at the beginning
4636     #1
4637     {true}{} {true} {}% the sentinel(s)
4638     \gmu@delim % the delimiter
4639     =\@ne % right side of \ifnum
4640   }

```

```

4643 \long\def\gmu@AND
4644 #1%
4645 {%
4646   \ifAllClauses {#1}%
4647     \@xa\@firstoftwo
4648   \else
4649     \@xa\@secondoftwo
4650   \fi
4651 }

```

A shorthand: as \Name but with the second parameter delimited with a space (for less tokens)

```

4657 \long\def\sName #1#2 {\@xa#1\csname #2\endcsname}

```

and a version that detokenises its #2

```

4660 \long\def\sdName #1#2 {\@xa#1\csname \detokenize{#2}\endcsname}

```

```

4662 \long\def\dName #1#2{\@xa #1\csname \detokenize{#2}\endcsname}

```

```

4664 \long\def\@sN #1 {\csname #1\endcsname}

```

```

4665 \long\def\@sdN #1 {\csname \detokenize{#1}\endcsname}

```

## Some typesetting macros

A centered overlap

```
4671 \pdef\clap #1{\hbox to \z@{\hss #1\hss}}
```

```
4673 \pdef\hsizecline #1{\hbox to\hsize{\hss #1\hss}}
```

because `\centerline` uses `\textwidth` which is not always what we want.

```
4677 \long\def\gmu@extreme
```

Note it's expandable and expands to the smallest (for #2 being <) or largest (for #2 being >) number/dimen of #3 and #4. May be used recursively (tree-way).

```
4682 #1% kind of test (num or dim)
```

```
4683 #2% inequality sign: < for minimum, > for maximum.
```

```
4684 #3% left side of comparison
```

```
4685 #4% right side of comparison
```

But *in fact* this macro eats a sequence of numexprs or dimexprs that must be terminated by `\relax` (or sth. `\ifx-equal`) and expands to the extreme value of that sequence.

```
4691 {%
```

```
4692 \gmu@if {x\@xa\@xa}{\@firstofmany#4\@nil\relax} % this complicated test is  
to allow arguments beginning with some \if<...> as in \possvfil e.g.
```

```
4695 {#3}%
```

```
4696 {%
```

```
4697 \gmu@if {#1} {\csname #1expr\endcsname#3\relax
```

```
4698 #2\csname #1expr\endcsname #4\relax }
```

```
4699 {\gmu@extreme {#1}#2{#3}}%
```

```
4700 {\gmu@extreme {#1}#2{#4}}%
```

```
4701 }%
```

```
4702 }% of \gmu@extreme
```

Two expandable macros that expect a sequence of undelimited num(expr)s terminated with `\relax` and expand to the biggest or the smallest one.

```
4707 \def\gmu@maxnum{\gmu@extreme {num}>}
```

```
4708 \def\gmu@minnum{\gmu@extreme {num}<}
```

And the same for dim(expr)s:

```
4711 \def\gmu@maxdim{\gmu@extreme {dim}>}
```

```
4712 \def\gmu@mindim{\gmu@extreme {dim}<}
```

And if we wish to assign the larger/smaller value in an iteration:

```
4716 \pdef\gmu@fitto
```

```
4717 #1% scope (nothing, \relax or \global.
```

```
4718 #2% left side of the assignment (must be a dimen able to be "passive")
```

```
4719 #3% the comparison (if #2#3#4 then reassign #2)
```

```
4720 #4% right side of the assignment—any correct text for \dimexpr.
```

```
4721 {%
```

```
4722 \gmu@if {dim}{#2#3\dimexpr (#4)+\z@\relax\relax}%
```

```
4723 {#1#2=\dimexpr (#4)+\z@\relax}%
```

```
4724 {}%
```

```
4725 }
```

```
4728 \pdef\gmu@g@enlargeto
```

```
4729 #1% #2 of the above
```

```
4730 #2% #4 of the above
4731 {\gmu@fitto\global{#1}<{#2}}
```

Let's define three auxiliary macros analogous to `\dywiz` from `polski.sty`: a shorthand for `\discretionary` that'll stick to the word not spoiling its hyphenability and that'll won't allow a line break just before nor just after themselves. The `\discretionary` T<sub>E</sub>X primitive has three arguments: #1 'before break', #2 'after break', #3 'without break', remember?

```
\discre 4743 \pdef\discre#1#2#3{\leavevmode\kern\z@
4744 \discretionary{#1}{#2}{#3}\penalty\@M\hskip\z@skip}
```

```
\discret 4746 \pdef\discret#1{\discre{#1}{#1}{#1}}
```

A discretionary hyphen that allows other (automatic) break-points in the word.

```
4750 \pdef\gmu@flexhyphen{%
4751 \discre{% before break
4752 \ifnum\hyphenchar\font>\z@
4753 \char\hyphenchar\font
4754 \fi
4755 }% end of before break
4756 {}% after break
4757 {}% without break
4758 }
```

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

```
4763 \def\:%
4764 \ifmmode\afterfi{\mskip\medmuskip}%
4765 \else\afterfi{\discre{\null}{}}}% \null to get \hyphenpenalty not \ex|
hyphenpenalty.
4767 \fi
4768 }
```

```
4773 \lpdef\hboxreflected#1{%
4774 \hbox{%
4775 \reflectbox{#1}%
4776 }%
4777 }
```

```
4780 \pdef\gmu@ifSystemX{% the 'If file exists' test is NOT expandable since it involves
opening some streams. Therefore we define this macro as protected.
4783 \IfFileExists{/etc/passwd}%
4784 }
```

```
4788 \def\hrule@zero{\hrule height\z@ width\z@ depth\z@}
4789 \def\vrule@zero{\vrule height\z@ width\z@ depth\z@}
```

```
4792 \def\do#1#2{% (2010/10/11) Define \gmu@iflast<sth.> as a two-argument expand-
able macro-conditional.
4794 \Name \def{gmu@iflast#1}{%
4795 \ifnum #2=\lastnodetype
4796 \@xa\@firstoftwo
4797 \else
4798 \@xa\@secondoftwo
4799 \fi
```

```

4800 }%
4801 }

4804 \do {glue}{11} \do{skip}{11}
4806 \do{kern}{12}
4808 \do {penalty}{13}

4811 \def\gmu@dimratio
4812 #1% numerator dim(en/expr)
4813 #2% denominator dim(en/expr)
4814 {\strip@pt
4815 \dimexpr 1pt *
4816 \numexpr\dimexpr (#1)*1\relax\relax /
4817 \numexpr \dimexpr (#2)*1\relax \relax
4818 \relax
4819 }

```

2010/10/21

```

4823 \def\MakeFalseIfDefined #1{%
4824 \ifcsname if#1\endcsname
4825 \csn{#1false}%
4826 \fi
4827 }

```

```

4829 \def\MakeTrueIfDefined #1{%
4830 \ifcsname if#1\endcsname
4831 \csn{#1true}%
4832 \fi
4833 }

```

2010/10/28

```

4836 \def\gmu@pageremain{\dimexpr
4837 \ifdim\pagegoal=\maxdimen \textheight \else \pagegoal \fi
4838 -\pagetotal
4839 \relax
4840 }

```

### Absolute values of dimens & nums

Strictly for use in the “absoluters”

```

4845 \def\gmu@GobbleMinus #1{\if-#1\else#1\fi}

4847 \def\gmu@absExpr
4848 #1%  $\varepsilon$ -TEX's expression primitive, num or dim so far (2010/12/17, 11.46)
4849 #2% expression of respective kind
4850 {%
4851 \@xa\gmu@GobbleMinus\the #1#2\endexpr
4852 }

4854 \def\gmu@absdim {\gmu@absExpr \dimexpr }
4855 \def\gmu@absnum {\gmu@absExpr \numexpr }

```

2010/10/29

```

4859 \pdef\gmu@SetPagegoal #1{%

```

This macro is L<sup>A</sup>T<sub>E</sub>X-specific in the sense that it assumes resetting of `\pagegoal` by `\output` at the beginning of a new page and excludes that case.

```
4863 \gmu@if {dim} {\pagegoal=\maxdimen}%
4864 {}% in this case we do nothing, as explained above
4865 {\pagegoal = \dimexpr (#1)\relax}%
4866 }
```

2010/11/10

```
4870 \pdef\gmu@SetPagegoalGlobal #1{%
```

This macro is L<sup>A</sup>T<sub>E</sub>X-specific in the sense that it assumes resetting of `\pagegoal` by `\output` at the beginning of a new page and excludes that case.

```
4874 \gmu@if {dim} {\pagegoal=\maxdimen}%
4875 {}% in this case we do nothing, as explained above
4876 {\global\pagegoal = \dimexpr (#1)\relax}%
4877 }
```

```
4881 \pdef\gmu@AdvancePagegoal #1{% the arg. should be a proper stuff for a sub-
      dimexpr (i.a., no spurious \relaxes are allowed).
```

```
4885 \gmu@SetPagegoal {\pagegoal +#1}%
4886 }
```

```
4889 \def\pagegoalortextheight{% as in the name:
```

(Note that „it’s alive, it’s alive!!!” but behaves as a dimen (except left side of assignments)). In particular, it’s a subject to `\the` and thus can fully expand to a “dead” dimen spec. in one step.

```
4895 \dimexpr
4896 \ifdim \pagegoal=\maxdimen
4897 \textheight
4898 \else \pagegoal
4899 \fi
4900 \relax
4901 }
```

```
4904 \def\gmu@totalht #1{% the arg. should be a box register.
4905 \dimexpr \ht #1+\dp#1\relax
4906 }
```

```
4909 \def\vbadness@M {%
4910 \unless\ifnum\vbadness=\@M
4911 \edef\gmu@VeryBadBadness {\the\vbadness}%
4912 \vbadness\@M
4913 \fi
4914 }
```

```
4916 \def\vbadness@Restore {%
4917 \ifdefined \gmu@VeryBadBadness
4918 \vbadness \gmu@VeryBadBadness \relax
4919 \else
4920 \PackageError{gmbase}{You try to gm-restore \vbadness\space
      where
4921 it is not gm-stored}{}%
4922 \fi
```

```

4923 }
    For proper indentation of  $\varepsilon$ -TeX's expressions let's introduce an alias of \relax
4928 \relaxen\endexpr
    2010/11/22 the same for TeX's (primitive) rules:
4931 \relaxen\endrule
    2010/11/22
4936 \def\gmu@ifpageodd {%
    %% #1% (implicit) what if page number is odd
    %% #2% (implicit) what if page number is even
4939 \ifodd \c@page
4940 \@xa\@firstoftwo
4941 \else
4942 \@xa\@secondoftwo
4943 \fi
4944 }
    2010/12/08, 15.14
4949 \gmu@DefSymbol\gmu@Fake
4951 \def\gmu@FakeLoaded
4952 #1% extension
4953 #2% name
4954 {\ifcsname ver@#2.#1\endcsname
4955 \@xa\ifx\csname ver@#2.#1\endcsname \relax
4956 \@xa\@xa\@xa \@firstofone
4957 \else
4958 \@xa\@xa\@xa \@gobble
4959 \fi
4960 \else
4961 \@xa\@firstofone
4962 \fi
4963 {\Name\def {ver@#2.#1}{\gmu@Fake }}%
4964 }
4966 \def\gmu@FakeUnloaded
4967 #1% extension
4968 #2% name
4969 {\ifcsname ver@#2.#1\endcsname
4970 \@xa\ifx\csname ver@#2.#1\endcsname\gmu@Fake
4971 \Name\let {ver@#2.#1}\@undefined
4972 \fi
4973 \fi
4974 }
    And an immediate use of the above. Because the polski package recatcodes a bunch of
    chars that are Polish diacritics in some ancient TeX encoding but in Unicode include the
    guillemots &al. that should rather remain "other".
4982 \pdef\LoadPackagePolski {%
4983 \gmu@FakeLoaded \@pkgextension {inputenc}%
4984 \RequirePackage {polski}%

```



```

4985 \gmu@FakeUnloaded \@pkgextension {inputenc}%
4986 }
4989 \def\hrule@zero{\hrule height\z@ width\z@ depth\z@}

```

## Combining accents (in X<sub>Y</sub>TeX only)

Added 2011/07/13, 11.56. Useful as delimiters of macros' arguments robust to writes. First used in \addcontentsline of a nonstandard toc.

```

4995 \pdef \protected@nil {}
4996 \let\protected@empty\protected@nil
4999 \ifgmuXeTeX

```

2011/02/03, 12.53 definitions of the macros for combining accents have been moved here from my personal package since they are needed in **gmdoc** (to typeset typewriter Ū (absent in TeX Gyre Cursor) but used as one of argument types in \DeclareCommand).

```

5007 \def\gmu@nound
typesets a char with the bound ##2 stripped off.
5009 #1% a char
5010 #2% bound number
5011 {%
5012 \gmu@unless {}
5013 {0%
5014 \gmu@if {num}{#2=\@ne }{0}{}%
5015 \gmu@if {num}{#2=13 } {0}{}%
5016 1}
5017 {}
5018 {%
5019 \leavevmode
5020 \kern -\glyphbound #1 1%
5021 }%
5022 #1%
5023 \gmu@unless {}
5024 {0%
5025 \gmu@if {num}{#2=\thr@@ }{0}{}%
5026 \gmu@if {num}{#2=13 }{0}{}%
5027 1}
5028 {}
5029 {%
5030 \kern -\glyphbound #1 3%
5031 }%
5032 }
5034 \def\glyphbound
5035 #1% char (name)
5036 #2% number of the bound
5037 {%
5038 \XeTeXglyphbounds #2 \XeTeXglyphindex "#1"
5039 }
5041 \def\gmu@halfcomb
5042 #1% the “basic” char

```

```

5043 #2% the accent char
5044 #3% emergency [fontspec] font spec
5045 {%
5046   #1\kern-\gmu@halfwd #1%
5047   \clap{%
5048     \gmu@unless {fontchar}{\font `#2 }
5049     {#3}{}}%
5050     #2}%
5051   \kern\gmu@halfwd #1%
5052 }
5054 \def\gmu@halfwd#1{0,45\fontcharwd\font`#1 }
5056 \def\gmu@llapcomb
5057 #1% the “basic” char
5058 #2% the accent char
5059 #3% emergency [fontspec] font spec
5060 {%
5061   \gmu@nobound #13%
5062   \llap{%
5063     \gmu@unless {fontchar}{\font `#2 }
5064     {#3}{}}%
5065     #2}%
5066   \kern \XeTeXglyphbounds 3 \XeTeXglyphindex "#1"
5067 }
5087 \fi % of if XYTEX
5092 \pdef\@rmfromreset
    Added 2011/06/01, 10.57 (GM)
5094 #1% counter to be freed, e.g., figure,
5095 #2% counter from whose power we free #1, e.g., chapter.
5096 {%
5097   {%
5098     \def\@elt ##1{%
5099       \gmu@ifdetokens{#1}{##1}%
5100       }% then we remove the element, otherwise
5101       {\@nx\@elt {##1}}%
5102     }%
5104     \Name\xdef{cl@#2}{\csname cl@#2\endcsname}%
5105   }%
5106 }
    2011/08/12, 13.06 (GM)
5126 \long\def \gmu@iflist {%
    #1 (implicit) what if in a list environment #2 (implicit) what if not in a list env.
5129   \ifnum \@listdepth>\z@
5130     \@xa\@firstoftwo
5131   \else
5132     \@xa\@secondoftwo
5133   \fi
5134 }
    2011/10/05, 16.17 (GM)

```

```

5138 \foone {\@makeother \^^I }{%
5139   \def\xiitab{\^^I}%
5140 }
5142 \foone {\catcode 9=\active } {%

```

Totally perverse, but seems to be useful with DocStrip.

```

5144   \def\gmu@maketabtab {%
5145     \def ^^I{\xiitab}%
5146   }%
5147 }
5150 </base>
5152 <*utils>

```

## The (gmutils package) options

```

quiet 5157 \DeclareOptionX{quiet}{\gmu@quiettrue
5158   \PassOptionsToPackage{quiet}{gmtypos}%
5159 }

```

The packages of this bundle may be loaded as options of the **gmutils** package. Here is how we provide it.

We define a requirer:

```

5165 \def\gmu@PackOptionX
5166 #1% name of a package with or without leading "gm".
5167 {%

```

So we declare an OptionX that by default loads this package thanks to a special CS having been defined to load it or do nothing.

```

#1 5170   \DeclareOptionX{#1}[on]{%
      %%   \ifcsname gmu@Require@#1\endcsname
      %%       \PackageError{gmutils}{Value clash for the ***#1*** package
option}{}%
      %%   \fi
5174   \lowercase{\@xa\if@gobble ##1\relax}% "off" given as the value
5175   \@namedef{gmu@Require@#1}{}%
5176   \else % "on"
5177   \afterfi{%
5178     \@namedef{gmu@Require@#1}{%
5179       \IfFileExists{gm#1.sty}%
5180       {\RequirePackage{gm#1}}% if there's a gm package, we load it, else we load
5182       {\RequirePackage{#1}}%
5183     }% of namedef
5184   }% of afterfi
5185   \fi
5186 }% of \DeclareOptionX
5187 \IfFileExists{gm#1.sty}%
gm#1 5188 {\DeclareOptionX{gm#1}[on]{%
5189   \ExecuteOptionsX{#1=###1}%
5190 }%
5191 }% of if yes. Else:

```

```

5192   {}%
5194 }
5196 </utils>

```

### `\DeclareCommand` and `\DeclareEnvironment`—the `gmcommand` package<sup>3</sup>

```

5201 <utils>   \gmu@PackOptionX{command}
5203 <*command>

```

(2010/07/26, v0.993:) Incompatibilities with earlier versions: because of making all the specifiers “case-insensitive”, i.e. aliasing the uppercases as lowercase, the `S` spec. ceased to be a `Single-token-catcher` and became a `Star-token-catcher`.

Moreover, the parameters for *all* the one-parameter specifiers became optional (not only for the lowercase as earlier).

Moreover, catcher names of CS specifiers are now created by `\stringing` the CS and stripping its backslash (earlier: by crude detokenising it).

```

5220 \RequirePackage{gmbase} % we require the low-level macros.

```

```

5222 \unless\ifcsname ifgmu@quiet\endcsname

```

```

5223 \Name\newif {ifgmu@quiet}% it has to be at least (at highest) in gmcommand since
      is used by it and not always entire gmutils is loaded.

```

(2010/09/06, v0.993:) moved here from `gmutils.sty` (a bug fix)

```

5228 \fi

```

The code of this section is based on the `xparse` package version 0.17 dated 1999/09/10, the version available in T<sub>E</sub>X Live 2007-13, in Ubuntu packages at least. Originally considered a stub ‘im Erwartung’ (Schönberg) for the L<sup>A</sup>T<sub>E</sub>X3 bundle, it evolved to quite a nice tool I think.

After a short deliberation I rename the command to `\DeclareCommand` which is much shorter than original `\DeclareDocumentCommand` and more adequate at least in my case: I don’t only use this powerful tool for ‘document’ commands.

```

\DeclareCommand<a CS><prefs.>{\<args' spec>}{<body, using #1...#2>}

```

The `<prefs.>` may be any (incl. empty) subset of

```

\global\protected\outer\long\relax!L.qQiIwW\sphack

```

first for for effect analogous to that of prefixing `\def` of a macro, `\relax` for the case when `\DeclareCommand` is used automatically, `!Ll` aliases for `\long`, `.qQ` to suppress placing of the diagnostic message in the definition `iIwW\sphack` to make the command `Invisible` (with `\@bsphack`—`\@esphack`).

(One more possible all-command prefix, `\envhack`, used to inform the machine the command will be used as an environment, is placed automatically when `\DeclareEnvironment` is used. You may use it however at your own risk if you read the code of this package and want to hack it.)

In the `<body>` you use single `#`es as a refernce to the parameters specified in `<args' spec>`, `##` (double hashes) as the parameters of macros defined by your command and so on.

The `<args' spec>` consists of the specifier tokens optionally preceded by a `>`{`<prefixex>`} sign and prefixes. Anything else is ignored, so you can write e.g., `#1...#9` for better readability if you like.

<sup>3</sup> This file has version number v0.996 dated 2011/10/12.

Before we go further with details, a word about general idea. `\DeclareCommand` defines a `\protected` macro named  $\langle a CS \rangle$  that takes no arguments and expands to a bunch of *argument catchers*. The notion of an “argument catcher” seems crucial for understanding how does this machinery work. Each specifier, that is a possibly-prefixed-specifier-token, is translated to proper argument catcher in  $\langle a CS \rangle$ ’es meaning. Then each of those catchers (postpones remaining list of catchers and) tries to catch the bunny, i.e., for the optional arguments performs respective `\@ifnext...` test and takes an argument if present and adds it to a dedicated toks (or adds the default value if suitable argument is absent (except for mandatory arguments that are scanned for anyway and whose absence results with an error)).

The  $\langle body \rangle$  becomes body of a macro with undelimited parameters of the number corresponding to the number of non-ignored  $\langle args' spec \rangle$ ’s specifiers.

Moreover, `\DeclareCommand` defines a macro carrying as its meaning the (parsed and simplified) specifiers’ sequence to allow e.g. repeating it in another `\DeclareCommand` thanks to `\SameAs` special specifier.

The  $\langle prefixes \rangle$  are:

- `P` or any of `p!lL\long\par` to make subsequent argument `\long` (some arguments are *always* `\long` which will be remarked);
- `\@xa`, `\expandafter` for one-level expansion of (the first token of) the first and all possible further specifier’s argument(s) before actual declaring;
- `\@nx@xa`, `\@nxxa`, `\nxxa` to leave first specifier’s argument intact and one-level expand the second;
- `\@xa@nx`, `\@xanx`, `\xanx` reverse of the above;
- `\@xaeacher`, `\xaeacher`, `\xaEacher`, `\xaE` for one-level expansion of the “each” token of the interating specifiers `\loop`, `U|u` and/or `W|w` (see the description of those specifiers);
- `\GroteNegere`, `\GrossIgnore`, `\GroteN`, `\GrossI` to ignore *all* the specifiers starting from this prefix and stopping at (any of) the following prefixes:
- `\GroteNegereStop`, `\GrossIgnoreStop`, `\GroteNStop`, `\GrossIStop`
- `\GroteLang`, `\GrossLong`, `\GroteL`, `\GrossL` to make all the specifiers following this prefix `\long` up to the (any of) following prefixes:
- `\GroteLangStop`, `\GrossLongStop`, `\GroteLStop`, `\GrossLStop`

The accepted argument specifiers are (case of the single letters doesn’t matter):

- `m|M` for mandatory argument (braced or not) (undelimited macro parameter in the sense of Chapter 20 of *The T<sub>E</sub>X book*),
- `(o|O|\NoValue)[\langle default \rangle]` for a L<sup>A</sup>T<sub>E</sub>X’s optional argument (in square brackets) with default value  $\langle default \rangle$  (which is passed as  $\# \langle n \rangle$  if the argument is missing),
- `(s|S|\NoValue)[\langle default \rangle]` for optional star of the catcode any of {11,12,13} (if is present, it becomes respective  $\# \langle n \rangle$ , or else `\NoValue` is at the  $\# \langle n \rangle$  (unlike in `xparse!`)).
- `(t|T|\NoValue){\langle list \rangle}[\langle default \rangle]` for a single Token, checks whether on the respective position there’s an unbraced token one of  $\langle list \rangle$  and returns it on  $\# \langle n \rangle$  if present or  $\{ \langle default \rangle \}$  if absent. The  $\{ \langle default \rangle \}$  is a braced optional. If absent, `\NoValue` is the default.

Almost like in `xparse`, `s` is almost a shorthand for `T{\star}`, but *unlike* in `xparse`, in the parsed arguments you get `\star` or `\NoValue` as  $\# \langle n \rangle$ . You can now declare

```
\DeclareCommand\mimbla{T{+-}m}{%
  \leavevmode
  \ifx#1+\raise\fi
  \ifx#1-\lower\fi
  \ifx#1\NoValue\@tempdima=\fi
  7pt \hbox{\#2}%
}
```

and after `\mimbla+{raised}` `\mimbla-{lowered}` `\mimbla {untouched}` get:

$$\begin{array}{ccc} & & \text{untouched} \\ & \text{lowered} & \\ \text{raised} & & \end{array}$$
 This example is rather silly but shows that you spend only one # for two symbols while in `xparse` you would spend two. What if you wanted 10 options for the optional symbol? Here it's no problem. And with any number  $k$  of tokens on the list the next # is always  $n + 1$  not  $n + k + 1$ .

- `(t|T|\NoValue){⟨list⟩}[⟨default⟩]` as above only respecting blank spaces
- `(q|Q|\NoValue){⟨list⟩}[⟨default⟩]` as ‘sequence’, a sequence of symbols from `⟨list⟩`; for the fundamental usage see line 7796. More clearly, the catcher specified with `Q{⟨tokens⟩}` catches a word over the alphabet `⟨tokens⟩ ∪` where  is ignored and shall not be passed in the respective `#⟨n⟩`.
- `(d|D|\NoValue)[⟨default⟩]` the **D**ecimal (expansion of an integer) argument, equivalent to `Q{+−0123456789}`. If no `⟨default⟩` is given, 0 is assumed.
- `(d|D|\drs)[⟨default⟩]` (d/D-haček): decimal integer respecting space (`d` and `Đ` are made available only in `XqTeX`). Works as the above only doesn't ignore/gobble spaces.:
 

```

\DeclareCommand \foo{ D }{\romannumeral #1\relax}
\DeclareCommand \fóó{ Đ }{\romannumeral #1\relax}

:\foo 17 17 :\quad :\fóó 17 17 :
```

results in:

```

:mcccxvii: :xvii 17 :
```

- `(c|C|\NoValue)[⟨default⟩]` for an optional argument in parentheses. Historically it comes from the ‘coordinate’ argument and may serve as such: if you declare `\DCcoordinate`, then its catcher will be redefined to check for presence of a comma and pass the argument as `{{⟨before comma⟩}{⟨after comma⟩}}` if comma present. `\NoValue` is passed if absent. By default `\DCnocoordinate` is executed that defines the `c` type argument as just another optional in parentheses.
- `(b|B|\NoValue)[⟨default⟩]` for for an *optional argument in curly braces*. That's strange for anyone acquainted with `LATeX` and contrary to its basic convention, but practised by Til Tantau in the `beamer` class. If missing, `\NoValue` is passed as `#⟨n⟩` and therefore all the tests `\If[No]Value(T|F|TF)` apply.
- `K{⟨#-string⟩}[⟨replacement⟩]` for a *mandatory Knuthian delimited or undelimited macro parameters*. This concept comes from Stephen Hicks' suggestion at `BachTeX 2009` of implementing arguments delimited with `#{`. So, in the mandatory first argument to `K` you give an arbitrary parameters string as described in Chapter 20 of *The TeX book*. If you don't provide the replacement, only `#1` of those parameters will be passed to the core macro of your command as `#⟨n⟩`. In both arguments you use single # char.
- `(g|G|\NoValue){⟨pair of tokens⟩}[⟨default⟩]` a **G**eneral catcher of an optional. For instance, to declare an optional in angles (used e.g. in Til Tantau's `beamer` class), declare `G{<>}`. Again, if the second argument is absent, `\NoValue` is assumed.
- `(a|A|\NoValue)[⟨default⟩]` (for ‘angles’): a shorthand for `G{<>}`
- `=[⟨assignment-stuff⟩]` a pseudo-argument that in fact *interrupts* parsing arguments to execute an assignment to `⟨assignment-stuff⟩`. The default `⟨assignment-stuff⟩` is `\@tempcnta`. For example,
 

```

\DeclareCommand\llf{=}{\lastlinefit\@tempcnta\par}
```

 defines `\llf` to be a command that expects a value for a numerical assignment, assigns it to `(\@tempcnta` and then to) the `\lastlinefit` special register and executes `\par` with that setting inside a group.

Actually, *<assignment-stuff>* may be empty {} and then each time our command is used the left side of the assignment has to be given explicitly and may even be different each time.

I call it “pseudo-argument” because it doesn’t add anything to the arguments’ list (toks).

*This pseudo-argument type should be considered experimental.*

- `\loop(\any|\none){<list to match for/against iteration>}`  
`[(\drop|\lost)(\any|\none){<list of dropped>}`  
`[(\count|\ubound)][<decimal>`  
`[(\default|\Default)][{<default>}]`  
`[(\each|\Each)][{<each>}]` for a catcher that iterates *while* (for *<“direction”>* being one of `\any`, `\W`, `w`, `\W`, `\w`) or *until* (for *<“direction”>* being one of `\none`, `U`, `u`, `\U`, `\u`) it meets tokens listed on *<list to match for/against iteration>*.

During this iteration the catcher drops the tokens listed on *<list of dropped>* if `\any` precedes that list or adds *only* those if the list is preceded with `\none`.

The number of iterations may be upper-bound with *<decimal>*, therefore you may write `\count` or `\ubound` before it for readability.

*<default>* is almost self-explaining: we have only explain when applies and what if absent. So, it applies when the parsed sequence of tokens/texts is empty—then *<default>* substitutes that emptiness. By default, i.e., when you specify no *<default>*, `\NoValue` is assumed.

*<each>* is a token (or a list of tokens) added before each token/text caught. An example is given at the `u|U` specifier’s description.

- `w|W` a shorthand for `\loop w`: a catcher iterating **While**.
- `u|U` a shorthand for `\loop u`: a catcher iterating **Until**. Let’s

```
\def\EmergencyFont#1{%
  \ifcat a\noexpand#1%
    \iffontchar\font`#1
      #1%
    \else{\fontspec{FreeSerif}#1}%
  \fi
\fi
}
```

```
\DeclareCommand\foo{
  u{\par<\bgroup} \each{\EmergencyFont}
  w{\<} \count 1
}{#1}
```

and then

```
\fontspec{HerculanumLTStd} \foo Pójdź, kińże tę chmurość w~głęb flaszy{}
```

(the Herculanum STD font hasn’t Polish diacritics) results in

P Ó J D Ż K I ń Ż Ę T ę C H M U R O Ś Ć W G ł ą B F L A S

- `\lostwax{<list to match against iteration>}`  
`[\drop](\any|\none){<list of dropped>}`  
`[[(\count|\ubound)]<decimal>`  
`[(\default|\Default)][{<default>}]`  
`[(\each|\Each)][{<each>}]`  
`[\lost|\Lost]{<lost-list>}[[(\count|\ubound)]<decimal>`  
`[\endlost|\EndLost]` for a catcher that iterates *until* as the one specified with `u|U` and *loses* (drops) its delimiter(s) as specified in the `\lost` part of specification, i.e., the ones present on *<lost-list>*, or *any* delimiter (cf. `\grab@lostwax`).

— `\SameAs`(*command*) “mimetic” specifier (cf. René Girard, James Alison: »mimetic desire«;-)). Repeats current specifiers of *command*.

Note that you could limit acceptable values of a mandatory or an optional-in-brackets argument to a list inside definition of the command, using `\IfAmong... \among...` defined in line 1438.

The `S/T`, `s` and `Q` arguments are always ‘long’, they allow `\par` as their value that is. They have to be unbraced to be parsed so there’s no danger of “runaway argument”.

`\long!LL` By default, all the `c`, `C`, `o`, `O`, `m`, `b` and `B` are ‘short’, they don’t allow `\par` in them that is. Note however that `\par` is allowed in the default values. If you wish to allow `\par` for all the arguments, you can say `\DeclareCommand\mycommand!...` — the optional `!` makes all the arguments ‘long’. Instead of `!` you can use `L` or `l` for ‘long’ or just `\long` itself.

In the arguments specification string you can write `>[{}<prefix>[]]` to make subsequent argument ‘long’ or ignored:

`Pp!LL\long\par` Any of `Pp!LL\long\par` to make a particular argument ‘long’, allowing `\par` in it that is, and/or any of `iI` to make the argument ignored (just gobbled).

(Note that also the `c` and `C` arguments may be made ‘long’. That’s because I use them not as coordinates but as just another kind of optional argument.

The concept of ignored arguments came to my head when I was declaring a command with three braced optionals and put optional stars only to distinguish the braced optionals.

For example, after

```
\DeclareCommand\GLBTQKi{%
  G{&&}{\#1 default}
  >LB{\#2 default}
  >iT{* \ht}
  Q{0123456789}{0}
  K{\#1\par\#2\par}{\text{\#1}\over\text{\#2}}
```

and you get `\GLBTQKi` 4-argument with:

`&\#1& G{&&}` optional short in a pair of `&` with `\NoValue` as the default,

`{\#2} >{L}B` optional long in curly braces,

*<ign.>* `>{i}T{* \ht}` star or `\ht` control sequence,

`\#3 Q{0123456789}{0}` optional sequence of decimal digits with default 0,

`\#4 K{\#1\par\#2\par}{\text{\#1}\over\text{\#2}}` mandatory sequence of two arguments both delimited with `\par`, that will be passed the inner macro as `\text{\#1}\over\text{\#2}`.

`\IfLong` The arguments may be tested inside the command with `\IfLong{\#<n>}{<what if long>}{<what if short>}`. The test looks for `\par` at any level of nesting (`\par`—`\par` in braces will not hide) since it uses the `\gmu@ifxany` test that iterates token by token not hash by hash.

`\global\outer` If you wish to define your command `\globally`, you can specify `\DeclareCommand% \mycommand\global`. If you wish to forbid usage of your command in arguments of macros, add the `\outer` prefix. As with original `TeX`’s `\def` and like, the prefixes are allowed in any order and in any number only here they come between the command’s name and the arguments specification. You can also add `\long`, as we mentioned above, and, for the symmetry, also `\protected`, although the latter is *always* added since the command is not expandable.

Handling of white spaces with optionals seems to me too complicated compared to the estimated weight of the problem and I haven’t faced it so far so I don’t provide anything. But!—but there are some commands that should be invisible in the typeset text, such as indexing commands and font declarations. For those there is a working `LATEX` mechanism



`iIwW\sphack` of `\@bsphack—\@esphack` and to use it I provide yet another ‘prefix’: if you type `W` or `w` (for ‘white’) or `I` or `i` (for ‘invisible’) or, if you prefer a prefix-like, `\sphack`<sup>4</sup> between the CS to be defined and arguments spec, `\@bsphack` and `\@esphack` will be added in proper places.

The original inner macros of the ancient `xparse` had names like `\@dc@o` etc. According to my  $\TeX$  Guru’s advice I changed them to `\ArgumentCatcher@letter(s)` to make the error messages less confusing. Well, I don’t know if they are but `\ArgumentCatcher@PO` looks better than `\@dc@PO` doesn’t it?

`\IfDCMessages` Talking of messages, there’s a Boolean switch `\IfDCMessages`. Its default setting is `\DCMessagesfalse` but if you set `\DCMessagestrue`, every `\command` created with my `\DeclareCommand` will issue a message “Parsing arguments for `\command`” at the beginning of its execution.

`.qQ` If you are positive that no such message will ever be useful, you can suppress the very placing of it in the command’s definition with an optional argument to `\Declarecommand` with all other ‘prefixes’, `.` or `q` or `Q` for ‘quiet’.

To sum up, `\DeclareCommand` takes the following arguments:

- #1 `>{P}m` the command to be defined (can be even `\par` if you really wish),
- #2 `Q{\long\global\outer\protected!lL.qQiIwW\sphack}{}` optional  $\varepsilon$ - $\TeX$ ’s prefix(es) and/or symbols for making all the arguments long (`!`, `l` or `L`) and/or to suppress placing of the diagnostic message in the definition (`.`, `q`, `Q`) and/or for placing `\@sphack—\@esphack` (`i`, `I`, `w`, `W`, `\sphack`),
- #3 `>{P}m` the arguments specification (can contain `\par` as you see),
- #4 `>{P}m` the definition body. You refer to the arguments with `#<n>` and can test their presence and absence with `\If[No]Value(T{F}{TF}){#<n>}` and if the argument was specified with the `>P` prefix (allows `\par` in itself), you can test it with `\IfLong{#<n>}{if long}{if short}`. You are also provided the `\IfAmong...among` and `\IfIntersect` tests defined earlier in this package to process the arguments, especially of the `S/T` and `Q` type.

`\DeclareEnvironment` There is also the `\DeclareEnvironment` command to define environments with sophisticated optionals. It takes the arguments analogous to those of `\DeclareCommand`.

The `iIwW\sphack` specifier however acts different: it doesn’t add `\@bsphack` nor `\@esphack` but only `\@ignoretrue` to the end macro so the spaces following `\end{myenvir}` will be ignored. (I tried the space hack but it’s problematic (“bad space factor” error) if an environment begins in the vertical mode and ends in horizontal.

So the arguments to `\DeclareEnvironment` are:

- #1 `>{P}m` the environment’s name; you may wonder why it allows `\par`; it’s to allow environments like `\string\par—I` met such an environment once.
- #2 `Q{\long\outer\global\protected!lL.qQiIwW\sphack}{}` to prefix the command (note that `\outer` prefix will actually not work since the command is called with `\csname`), make all the arguments ‘long’ (`\long`, `!`, `l` or `L`), not to place the message issuer in the command (`.`, `q`, `Q`) or to make the environment ignores spaces following its end (`iIwW\sphack`).
- #3 `>{P}m` the arguments specification (both for the begin and for the end macros)
- #4 `>{P}m` the begin definition,
- #5 `>{P}m` the end definition; it can use the same parameters (`#<n>`’s) as the begin definition. Note however that there is only one specification of the arguments and both begin and end have to have 9 parameters in total at most.

<sup>5737</sup> `\unless\ifdefined\@temptokenb`

<sup>4</sup> I don’t define the `\sphack` control sequence and don’t assume it’s defined. I use it only as a marker and my use of it doesn’t create an entry in the hash table.

```

\@temptokenb 5738 \newtoks\@temptokenb
5739 \fi

\if@dc@alllong@ 5743 \newif\if@dc@alllong@ % for an option of all arguments long (it's stronger than
GroteLang defined later (the latter is \let it).
\if@dc@quiet@ 5746 \newif\if@dc@quiet@ % for suppressing the message of using of declared command.
\ifDCMessages 5749 \newif\ifDCMessages % a global switch to suppress the message about parsing.
\@dc@arguments 5754 \newtoks\@dc@arguments % the register for storing parsed \\command {#1}...{\langle n\rangle}.
\@dc@catchernum 5759 \newcount\@dc@catchernum
\@dc@argnum 5760 \newcount\@dc@argnum

\if@dc@ignore@ 5762 \newif\if@dc@ignore@
\if@dc@GroteNegere@ 5763 \newif\if@dc@GroteNegere@ % for "gross" ignoring
\if@dc@GroteLang@ 5764 \newif\if@dc@GroteLang@ % for "gross" longness

\if@dc@long@ 5767 \newif\if@dc@long@
\if@dc@BareSpace@ 5769 \newif\if@dc@BareSpace@

%% \gmu@DefSymbol\BareSpace % no need: the test checks bebackslashed strings.

5775 \def\@dc@long@letter{%
5777 \if@dc@long@ P\fi
5778 }

\if@dc@xadefault 5780 \newif\if@dc@xadefault

This is a switch whether a default value of an argument should be one-expanded (\ex|
pandaftered) before adding it to the catchers toks. This switch may be set true in a
prefix and to make it safe it has to be set false after parsing of each argument. That's why
it's set false in so many places. Maybe in the future we'll add analogous switch whether
to \edef default value, then they both will have to be switched in the same places.

And three swith macros for the specifiers with two defaults.

5792 \def\@nx@xa{\@nx\@xa}
5793 \def\@xa@nx{\@xa\@nx}
5794 \def\@xa@xa{\@xa\@xa}

\if@dc@xadefault@i@ 5797 \newif\if@dc@xadefault@i@
\if@dc@xadefault@ii@ 5798 \newif\if@dc@xadefault@ii@
\if@dc@xaeacher@ 5799 \newif\if@dc@xaeacher@

5802 \def\@dc@falsify@xadefaults{%
5803 \@dc@xadefaultfalse
5804 \@dc@xadefault@i@false
5805 \@dc@xadefault@ii@false
5806 }

5810 \lpdef\DeclareCommand#1#2#3{% 7604.

#1 command to be defined,
#2 arguments specification,
#3 definition body.

\@dc@alllong@true 5827 \@dc@alllong@true
5828 \@dc@ResetParseAuxilia
5830 \@dc@ParseNextSpecifier #2%

```

```

5831 \@dc@buckstopshere% it's the sentinel of parsing. Doesn't have to be defined since
        the test performs a strings comparison.
5833 \protected\edef#1{%
5834   \@nx\@dc@   {\the\toks@}%
5835   \@xanxcs{\@dc@InnerName{#1}}%
5836   \@nx #1%
5837 }%
5839 \edef\gmu@DeclareCommand@resa{%
5840   \long\def\@xanxcs{\@dc@InnerName{#1}}%
5841   \@xau\@dc@innerhashes{%
5842     \unexpanded{#3}}%
5843 }% of resa
5844 \gmu@DeclareCommand@resa
5845 }% of the 'draft' \DeclareCommand.

5848 \long\def\@dc@
5849 #1% the argument catchers in a pair of braces (their arguments may contain \par so
        the macro is long).
5851 #2% \thecommand\
5852 #3% \thecommand
5853 {%
5854 \ifx\protect\@typeset@protect
5855   \@xa\@firstofone
5856 \else
5857   \protect#3\@xa@gobble
5858 \fi
5859 {\@dc@arguments{#2}#1\the\@dc@arguments}%
5860 }

\@dc@ParsedSpecs 5863 \newtoks \@dc@ParsedSpecs

5866 \def\@dc@ResetParseAuxilia{%
5867   \@dc@ParsedSpecs={}%
5870   \c@dc@catchernum\z@ % the count of catchers
5871   \c@dc@argnum\z@ % the count of arguments (inner arity) (it may be smaller than
        the above if we ignore some arguments)
5874   \toks@{}}% in this register we will store the created sequence of argument catchers.

        And for the  $\mathring{M}$  specifiers:

5878   \@dc@Mmode@false
5880   \emptify\@dc@innerhashes % in this macro we will store the sequence of #1#2...

5883   \@dc@GroteNegere@false
5884   \let\if@dc@GroteLang@\if@dc@alllong@
5885 }

\if@dc@Mmode@ 5888 \newif \if@dc@Mmode@
5889 \def\@dc@Ms@num {\z@}%

5891 \def\@dc@Ms@init{%
5894   \@dc@Mmode@true
5895   \def\@dc@Ms@num {\z@}%

```

We memorise the state of longness and ignorance at the beginning of  $\mathring{M}$ 's collection (encoded as a numexpr consisting of binary vector of the (standard conversion to  $\{0,1\}$ ) of the switches `\if@dc@long@` and `\if@dc@ignore@`).

```

5901 \edef\@dc@Ms@initial@listate{%
5902   \@dc@Ms@listate
5903 }% of edef
5904 }

5906 \def\@dc@Ms@listate{%
5907   \boolstobin{ {@dc@long@}{@dc@ignore@}}%
5908 }

5910 \def\@dc@Ms@shipout {%
    This macro adds the number of collected M|m's after the catcher and ends the M-mode:

5914   \gmu@if {num} {\@dc@Ms@num>\z@}%
5915   {\@xa@dc@addtoParsed@\@xa{\@dc@Ms@num}%
5916     \def\@dc@Ms@num{\z@}%
5917     \@dc@Mmode@false
5918   }%
5919   {}%
5920 }

5923 \def\@dc@ResetStepAuxilia {%
    We set the local ignorance switch to its "gross" value

5925   \let\if@dc@ignore@ =\if@dc@GroteNegere@
5926   \let\if@dc@long@ = \if@dc@GroteLang@

    and reset the expandaftering switches

5928   \@dc@falsify@xdefaults
5929   \@dc@BareSpace@false
5931 }

5934 \def\@dc@argtypes{%
5935   =\gobblespace \loop \lostwax \SameAs \Scope
5936   AaBbCcDdGgKkMmOo%
5937   QqSs*TtUuWw%
5938   \drs
5939 }

5941 \def\@dc@drs{\drs} % decimal respecting space

5943 \@ifXeTeX
5944 {\addtomacro\@dc@argtypes{dđİtÜüÔô}%
5945   \addtomacro\@dc@drs{dđ}%
5946 }
5947 {}

5950 \def\@dc@ParseSpecifier #1{% The main inner macro parsing argument specifiers
    in \DeclareCommand.

    %% \@dc@ResetStepAuxilia % putting it here is wrong andd leads to disaster

5966   \gmu@CASE {sstribs}{#1}\@dc@buckstophere}%

    If we meet the sentinel we stop.

5969   {%

```

There could have been caught some M|m's in the previous steps and they are not put immediately, so now we shipout their number, if any:

```
5972 \@dc@Ms@shipout
```

And edefine the macro carrying the inner hashes:

```
5975 \edef\@dc@innerhashes {%
5976 \gmu@hashes 1{\numexpr\c@dc@argnum +\@ne}%
5977 }%
5978 }%
```

Else we check if we've met the prefixer

```
5982 \gmu@CASE {stribs}{\#1>}%
5983 {%
```

%% \@dc@ResetStepAuxilia % No! the prefix-switches are reset at \@dc@ParseNextSpecifier and here we may be after previous prefix.

```
5986 \grab@prefix }%
```

Else we check if #1 is a known arg specifier...

```
5990 \@XA{%
5991 \gmu@CASEsbnone{\#1}\@xa{%
```

Here is the list of arg specifiers (arg types) recognised by \DeclareCommand.

Anything else (if not parsed as a specifier's parameter or prefix) is just ignored. For instance, you may write #1...#9 to make your code more readable.

```
6000 \@dc@argtypes
6001 }%
```

...and ignore it and iterate further if not...

```
6004 {\IgnInfo{gmcommand}{while parsing arg specifiers}{\#1}%
6005 \@dc@ParseNextSpecifier }%
```

...or else (#1 is known and not stop and not prefixer) we check if it's the \SameAs special specifier.

If so, then we apply special parser \@dc@SameAs (quite simple in fact).

```
6013 \gmu@CASE {stribs}{\#1}\SameAs}%
```

In this case we just expand the specifiers of "\SameAs arguments"

```
6016 {\@dc@SameAs}%
```

Here #1 is not stop neither prefixer nor the special specifier \SameAs, so we add its catcher possibly with »P« for longness, and possibly prefixed with ignorance.

If we parsed sth. else than M|m, we possibly ship the m's collected formerly:

```
6024 \gmu@ifsbnone {\#1}{Mm}%
6025 {\@dc@Ms@shipout }%
6026 {}%
```

But only if not in the M-mode (about special treatment of M|m's see line 6095).

```
6031 \gmu@if {@dc@Mmode@}{}%
6032 }%
6033 {%
```

Now. If we parse the assignment pseudo-argument, we don't want to add anything to the arguments' toks register so we apply general mechanism of ignoring an argument. The same for the pseudo-argument that causes ignoring spaces.

```
6040 \gmu@ifsbany{#1}{=\gobblespace}{\@dc@ignore@true}{}%
```

and add the catcher of #1 type.

The catcher's CS is `\ArgumentCatcher@P`(*arg. type*):

```
6046 \gmu@ifsw {@dc@BareSpace@}%
6047 {\@dc@addtoparsed@BareSpace }
6048 {}%
6054 \gmu@ifsw {@dc@long@}%
6055 {\@dc@addtospecs@bare{>P}}
6056 {}%
6058 \gmu@ifsw {@dc@ignore@}%
6059 {\@dc@addtoparsed@Ignore }%
6060 {}%
6062 \@dc@addtospecs@bare {#1}%
6064 \@xa\addtotoks\@xa\toks@\@xa{%
6065 \csname ArgumentCatcher@\@dc@long@letter
6066 \strip@bslash{#1}%
6067 \endcsname
6068 }% of toks' text.
```

Now we step the counters of catchers and argums (they'll be unequal if some argument is ignored or if there are multiple subsequent m's) and probably add #*n* tokens to respective toks.

```
6074 \advance\c@dc@catchernum\@ne
6075 }% of if not M-mode
```

But we step the number of arguments (of the inner macro) also in the **M**-mode:

```
6079 \gmu@notif {@dc@ignore@}{}%
6080 {\@dc@addinnerhash }%
6081 {}% (empty "not-else" branch of ignoring)
```

(the ignorance switch will be reset at the beginning of next step of parsing, just like other step-local switches).

Now we parse the parameter(s) of #1 where we should (usually the default value)

The **M**m catcher(s) are treated specially, for legacy and performance reasons: when we meet an **m** or **M**, we set our parser to the **M**-mode if not yet, to look if it's not a beginning of a longer sequence of such specifiers.

```
6095 \gmu@CASEsbany {#1}{Mm}%
```

If so, we process it appropriately:

```
6098 {\@dc@process@ms}%
```

The **K** catcher also requires special treatment since the parameters are not just passed to it but a particular macro is defined with use of them.

```
6104 \gmu@CASEsbany {#1}{Kk}%
6105 {\@dc@define@K}%
```

Else (not mandatory(s) neither Knuthian) we check if parameterless, one- or two-parameter.

```
6110 \gmu@CASEsbany {#1}{\gobblespace}%
```

For the parameterless specifiers we did all and we just continue parsing.

```
6115 {\@dc@ParseNextSpecifier}%
```

Now the one-parameter specifiers (the parameter is optional, although (necessarily) in curly braces). For these we look for the parameter and provide the default if not provided by the user.

```
6121 \gmu@CASEsbany{#1}{=\Scope AaBbCcDdOoSs*}%
6122 {\grab@optparam{#1}}%
6125 \@XA{\gmu@CASEsbany{#1}}\@xa{\@dc@drsres}%
6126 {\grab@optparam{#1}}%
```

For the loop catchers, `Uu` so far, maybe `Qq` in the future:

```
6130 \gmu@CASEsbany{#1}{UuÜÿWw\lostwax}%
6131 {\@dc@grab@Loop #1}% putting the specifier's letter as #1 for the grabber sup-
presses adding it to the parsed specifiers' list.
```

Else we've met a specifier with first parameter mandatory and second optional (the latter has to be in curly braces):

```
6138 \gmu@lastCASE
6139 {%
6140 \grab@twoparams{#1}}%
6141 \gmu@ESAC
6143 }% of \@dc@ParseSpecifier

6146 \def\@dc@addtoparsed@Ignore{%
6147 \addtotoks\toks@\@dc@ignorearg}%
6148 \@dc@addtospecs@bare{>i}%
6149 }

6152 \def\@dc@addinnerhash{%
6153 \advance\c@dc@argnum\@ne
6154 }

6157 \def\@dc@addtoparsed@BareSpace{%
6158 \addtotoks\toks@\@dc@BareSpaceArg}%
6159 \@dc@addtospecs@bare{>\BareSpace}%
6160 }

6163 \def\@dc@process@ms{%
6164 \gmu@if {@dc@Mmode@}{}%
```

If in `M`-mode, we compare current state of ignorance and longness with their initial values

```
6168 {\gmu@if {num}{\@dc@Ms@initial@listate=\@dc@Ms@listate}}%
```

If the state hasn't changed, we look if the optional number of `m`'s is provided and proceed in any case appropriately.

```
6172 {\@dc@process@ms@grabnum}% of if the state has'nt been changed
```

Else, i.e. when we have an `M|m` specifier but of another longness or ignorance, we ship the `m`'s collected so far and *close* the `M`-mode...

```
6177 {\@dc@Ms@shipout
```

...and *reparse* this `M|m` without changing current ignorance and longness, to let the new instance of parser add now closed number of `m`'s and the new catcher.

Remember however that we've increased number of arguments so now we set it back:

```
6185 \advance\c@dc@argnum\m@ne
```

6186        \@dc@ParseSpecifier m%

Note it's not **Next**, which means we don't reset the switches. And first of all, it's not **process@ms**, because we have to add new catcher etc.

6190        }% of if the state has been changed

6192        }% of if M-mode

If we are not in M-mode but have parsed an M|m, now we set us to be in the mode and look for (optional) number of m's.

6196        {\@dc@Ms@init

(This macro sets the number of m's found to 1 and turns the switch \@if@dc@Mmode@true).

6201        \@dc@process@ms@grabnum

6202        }% of not if M-mode.

6203 }% of dc@process@ms

We peep the next char and if it looks as an argument

6209 \def \@dc@process@ms@grabnum {%

6210    \@ifnextanyIS {\bgroup \@ne \tw@ \thr@@ 123456789}%

6211    {%

6212      \@dc@process@ms@fin}%

6213    {%

6214      \@dc@process@ms@fin \@ne}%

6215    }

6217 \def \@dc@process@ms@fin #1{%

6218    \edef\@dc@process@ms@hash {\the\numexpr#1}%

6219    \edef\@dc@process@ms@left {\the\numexpr 9-\c@dc@argnum+\@ne}%

+1 because we've increased the #es number in line 6080.

6223    \gmu@AND {%

6224      {num} {\@dc@process@ms@hash >\z@}

6225      {num} {\@dc@process@ms@hash <\numexpr

6226        \@dc@process@ms@left+1 \relax }%

6227    }%

6228    {\stepnummacro \@dc@Ms@num \@dc@process@ms@hash

And we add proper number of #es

6231    \edef\@dc@hashgoal {%

6232      \the\numexpr\c@dc@argnum +%

6233      \@dc@process@ms@hash-1}% minus 1 because one has (one has already been added in line 6080).

6236    \@whilenum \c@dc@argnum<\@dc@hashgoal \do{%

6237      \@dc@addinnerhash

6238    }%

6240    \@dc@ParseNextSpecifier

6241    }% of if all conds. satisfied

6242    {% some of the conds. not satisfied

6243      \PackageError{gmcommand}{%

6244        The argument to the M or m specifier, if any, has to be ^^J%

6245        a number between 1 and \@dc@process@ms@left ^^J%



```

6246     (there is/are \the\c@dc@argnum\space arg(s) declared
6247     already).^J%
6248     I ignore this m/M.}%
6249     }%
6250     \gmu@passbraced \@dc@ParseNextSpecifier {#1}%
6251     }%
6252 }% of \@dc@process@ms

6255 \def\@dc@ParseNextSpecifier{%
6256   \@dc@ResetStepAuxilia
6257   \@dc@ParseSpecifier
6258 }

6261 \lpdef\@dc@AddAndParse#1{%
6262   \dc@addtoParsed@{#1}%
6263   \@dc@falsify@xadefaults
6265   \@dc@ParseNextSpecifier
6266 }

```

The eating M’s uses the same concept of a “while” iterator as the `\loop` catcher but we prefer to define here a simplified version for this particular purpose.

```

6273 \def\grab@optparam #1{% arg specifier
6274   \@ifnextchar\bgroup{\@dc@AddAndParse}%
6275   {%

```

If the default is not provided in the command’s declaration, then we check whether it’s provided particularly for this specifier or put `\NoValue`.

```

6279   \edef\@dc@tempa{%
6280     \gmu@if {csname}%
6281     {\@dc@optparam@deft@\strip@bslash{#1}\endcsname}%
6282     {\@xaucs {\@dc@optparam@deft@\strip@bslash{#1}}}%
6283     {\@nx \NoValue}%
6284   }%
6285   \@xa\@dc@AddAndParse\@xa{\@dc@tempa}%
6286 }%
6287 }

```

Default default values for some argument types:

The declaring macro may be used with any-case version of the specifier, because both the upper- and lowercase defaults are defined, however this works only for letter specifiers. The caseness of CS specifiers is not modified.

```

6297 \long\def\@dc@DeclareDefault #1#2{%
6298   \uppercase{%
6299     \Name\edefU{\strip@bslash{\@dc@optparam@deft@\strip@bslash{#1}}{%
6300       #2}}%
6301   \lowercase{%
6302     \Name\edefU{\@dc@optparam@deft@\strip@bslash{#1}}{#2}}%
6303 }

```

The default default of “decimal” argument is 0.

```

6306 \@dc@DeclareDefault D{0}
6307 \@dc@DeclareDefault d{0}

6309 \@ifXeTeX {%

```

```

6310 \dc@DeclareDefault D{0}%
6311 }
6312 {}

```

The default default delimiter of the “Scope” argument’s parsing is a star of any catcode of {11, 12, 13}

```

6319 \xa\dc@DeclareDefault \xa\Scope\xa{\all@stars}
6321 \dc@DeclareDefault = {\tempcnta}

```

The “sequence” and “Until” arguments have `\NoValue` as the default default values (which hasn’t to be announced so explicitly since it’s the default setting):

```

6329 \dc@DeclareDefault Q {\NoValue}

```

```

6331 \ifXeTeX {%
6332 \dc@DeclareDefault O{\NoValue}%
6333 }
6334 {}

```

```

6337 \dc@DeclareDefault U {\NoValue}

```

```

6340 \long\def\dc@maybe@expandafter #1{%
6341 \gmu@if {\dc@xadefault}{}%
6342 {\gmu@ifutokens{#1}{\NoValue}%
6343 {\@secondoftwo}{\@firstoftwo}%
6344 }% if we are to expandafter #1.
6345 {\@secondoftwo}% if we don’t expandafter #1
6346 }

```

```

6348 \long\def\dc@addtotoks@ #1{%
6349 \dc@maybe@expandafter {#1}%
6351 {\toks@\xa\xa\xa{\xa\the\xa\toks@\xa{#1}}}%
6352 {\toks@\xa{\the\toks@{#1}}}%
6353 }

```

```

6355 \long\def\dc@addtospecs #1{%
6356 \dc@maybe@expandafter {#1}%
6357 {\xa \addtotoks \xa\dc@ParsedSpecs\xa{\xa{#1}}}%
6358 }%
6359 {\addtotoks\dc@ParsedSpecs{#1}}}%
6360 }

```

```

6362 \long\def\dc@addtoParsed@ #1{%
6363 \dc@addtotoks@{#1}%
6364 \dc@addtospecs{#1}%
6365 }

```

```

6367 \long\def\dc@addtospecs@bare #1{%

```

note it also doesn’t respect `xadefault` switch.

```

6369 \addtotoks\dc@ParsedSpecs{#1}%
6370 }

```

```

6372 \long\def\dc@addtotoks@bare #1{%
6373 \addtotoks\toks@ {#1}%
6374 }

```

```

6376 \long\def\dc@addtoParsed@bare #1{%

```

note it also doesn't respect `xadefault` switch.

```

6378 \@dc@addtotoks@bare {#1}%
6379 \@dc@addtospecs@bare {#1}%
6380 }

6383 \long\def\grab@twoparams
6384 #1% the specifier
6385 #2% first parameter, which is mandatory
6386 {% default default (when default is absent)
6387 \if@dc@xadefault@i@\@dc@xadefaulttrue\fi
6388 \dc@addtoParsed@{#2}%
6389 \if@dc@xadefault@ii@
6390 \@dc@falsify@xadefaults\@dc@xadefaulttrue
6391 \fi
6392 \grab@optparam{#1}%
6393 }

6396 \long\def\@dc@addargum#1{%
6397 \addtotoks\@dc@arguments{#1}%
6398 \@iwruif {@@@ @dc@arguments: >>\the\@dc@arguments<>}%
6399 }

6401 \Store@Macro\@dc@addargum

6403 \pdef\@dc@ignorearg{%
6404 \long\def\@dc@addargum##1{%
6405 \Restore@Macro\@dc@addargum}%
6406 }

6408 \StoreMacro@nocat\gmu@passbraced

6410 \pdef\@dc@BareSpaceArg{%
6412 \let \gmu@passbraced \gmu@passbracedNotSp
6413 \def\@dc@addargum {%
6414 \Restore@Macro\@dc@addargum

%% \Restore@Macro\@dc@Loop@CareOfSpace

6416 \Restore@Macro\gmu@passbraced
6417 \@dc@addargum
6418 }%
6419 }

```

Parsing of Knuthian parameters string is easier to implement with full-feature `\DeclareCommand` so we postpone it till line 7867

```

6425 \def\grab@prefix@CASE{% just a shorthand
6426 \@xa\gmu@CASEsbany\gmu@forarg
6427 }

```

```

6429 \long\def\grab@prefix #1{%

```

(2010/07/26, v0.993:) made for-eaching to make it behave as expected, i.e. that latter settings prevail over the former

%% `\@dc@ResetStepAuxilia` % No! We want allow many subsequent prefixes, they make no harm.

```

6440 \gmu@foreach#1\gmu@foreach@delim {%
6442 \grab@prefix@CASE

```

```

6443     {\long!lL\par Pp}%
6444     {\@dc@long@true}%
6446     \grab@prefix@CASE
6447     {Ii}%
6448     {\@dc@ignore@true}%
6450     \grab@prefix@CASE
6451     {\@xa\expandafter}%
6452     {\@dc@xadefaulttrue}%
6454     \grab@prefix@CASE
6455     {\@xa \@xa@nx \@xa@xa \expandafter \@xanx \xanx \xaxa}%
6456     {\@dc@xadefault@ii@true}%
6458     \grab@prefix@CASE
6459     {\@xa \@nx@xa \@xa@xa \expandafter \@nxxa \nxxa}%
6460     {\@dc@xadefault@ii@true}%
6462     \grab@prefix@CASE
6463     {\@xaeacher \xaeacher \xaEacher \xaE }%
6464     {\@dc@xadefault@ii@true}%
6466     \grab@prefix@CASE
6467     {\GroteNegere \GrossIgnore \GroteN \GrossI}
6468     {\@dc@GroteNegere@true
6469     \@dc@ignore@true}%
6471     \grab@prefix@CASE
6472     {\GroteNegereStop\GrossIgnoreStop
6473     \GroteNStop\GrossIStop}
6474     {\@dc@GroteNegere@false
6475     \@dc@ignore@false}%
6477     \grab@prefix@CASE
6478     {\GroteLang\GrossLong\GroteL\GrossL}
6479     {\@dc@GroteLang@true
6480     \@dc@long@true}%
6482     \grab@prefix@CASE
6483     {\GroteLangStop\GrossLangStop
6484     \GroteLStop\GrossLStop}%
6485     {%
6486     \let@if@dc@GroteLang@=\if@dc@alllong@
6487     \let@if@dc@long@=\if@dc@alllong@
6488     }%
6498     \grab@prefix@CASE
6499     {\BareSpace}
6500     {\@dc@BareSpace@true}
6506     \gmu@EatCases
6507     {\IgnInfo{gmcommand}{while parsing arg specifiers}{#1}}%
6508     \gmu@ESAC
6510 }% of for each
6511 \@dc@ParseSpecifier %

```

Note that here (unlike in all other places) is `\@dc@ParseSpecifier` not `\@dc@ParseNextSpecifier`. The difference between them is in that the former doesn't reset the switches (which we've just set here).

```

6517 }% of \grab@prefix
6520 \long\def\dc@DefParseNext#1{%

```

This is to allow hashes in the defaults.

```

6522 \edefU\@dc@parse@next{#1}%
6523 }

6526 \def\@dc@Alias
6527 #1% \lowercase or \firstofone
6528 #2% left side of the assignment
6529 #3% right side of the assignment
6530 #4% Lower or empty
6531 {%
6532 \gmu@if {csname}%
6533 {ArgumentCatcher@\strip@bslash #3\endcsname}%
6534 {#1{%
6535 \@xa\let\csname\strip@bslash\ArgumentCatcher@
6536 \strip@bslash #2\@xa\endcsname}%
6537 \csname ArgumentCatcher@\strip@bslash #3\endcsname
6538 }{\PackageError{gmutils/gmcommand}{%
6539 You're trying to #4Alias the »\unexpanded{#3}« catcher^^J%
6540 but it is not defined!}}%
6541 }%
6542 }%
6543 }

```

To provide lowercase parallels of a specifier

```
6546 \def\@dc@LowerAliases #1{%
```

To provide lowercase aliases of the catchers:

```

6548 \@dc@Alias \lowercase {#1}{#1}{Lower}%
6549 \@dc@Alias \lowercase {\P#1}{\P#1}{Lower}% we provide »P« as a CS to avoid
        lowercase'ing it.
6551 }

```

```

6554 \def\@dc@AliasCatcher
6555 #1% left side of the assignment
6556 #2% right side of the assignment
6557 {%
6558 \@dc@Alias \firstofone {#1}{#2}{}%
6559 }

```

```

6561 \def\@dc@AliasPLower
6562 #1% specifier to alias
6563 {\@dc@AliasCatcher {P#1}{#1}%
6564 \@dc@LowerAliases{#1}%
6565 }

```

Parsing of a braced optional. Note the test is `\ifcat` so *any* begin-group character will turn it true.

Note that although this parser issues an error when brace contains `\par`, its default may still contain `\par`.

```

6574 \long\def\ArgumentCatcher@B
6575 #1% default value
6576 #2% tail of args catchers.
6577 \@dc@arguments % delimiter
6578 {%
6579 \@ifnextcat\bgroup

```

If we are before a begin-group token, we store the tail of parsers in an auxiliary macro and launch inner catcher of a mandatory argument.

```
6584   {\dc@DefParseNext{#2}\ArgumentCatcher@m@i }%
6585   {\dc@addargum {{#1}}#2\dc@arguments }%
6586 }
```

Analogously to the previous catcher only the mandatory catcher is long.

```
6591 \long\def\ArgumentCatcher@PB
6592 #1% default value
6593 #2% tail of args catchers.
6594 \dc@arguments % delimiter
6595 {\@ifnextcat\bgroup
6596   {\dc@DefParseNext{#2}\ArgumentCatcher@Pm@i }%
6597   {\dc@addargum {{#1}}#2\dc@arguments }%
6598 }
```

```
6600 \dc@LowerAliases B
```

```
6603 \long\def\ArgumentCatcher@T@ % parsing of a single Token continued:
6605 #1% kind of test (strings or StrX so far (2010/12/28, 7.24))
6606 #2% the list to search #4 on,
6607 #3% the default value,
6608 #4% the tail of args parsers,
6609 #5% the token we search in #1 (it's an unbraced single token as we checked in line
        6628).
6611 {%
6612   #1 #5{#2}%
6613   {\dc@addargum {{#5}}#4\dc@arguments}%
6614   {\dc@addargum {{#3}}#4\dc@arguments#5}%
6615 }
```

```
6617 \long\def\ArgumentCatcher@T % parsing of a single Token. It's always long since
        we look for a single unbraced token so there is no danger of "runaway argument".
```

This catcher uses the `StrX` test so is very subtle: it applies `\ifx` first and if the compared tokens are both CS es and `\ifx`-equal then `\ifstrings` is applied.

```
6624 #1% the list we search optional token on,
6625 #2% the default value,
6626 #3% the tail of args parser.
6627 \dc@arguments{%
6628   \@ifnextnotgroup
6630   {\ArgumentCatcher@T@ \gmu@ifStrXany {#1}{#2}{#3}}%
6631   {\dc@addargum {{#2}}#3\dc@arguments}% if we parse an opening or closing
        brace, then we are sure it's not any expected Single.
6634 }
```

```
6636 \dc@AliasPLower T
```

```
6639 \long\def\ArgumentCatcher@T #1#2#3% as in ...@T
6640 {% one-token catcher that respects space.
6641   \ArgumentCatcher@Loop
6642   {StrX}{\any}{#1}{\any}{}{1}{#2}{% empty each
6643   }{#3}%
6644 }
```

```
6646 \dc@AliasPLower T
```

This is incompatibility of this version (v0.993) with other versions.

```
%% \let\ArgumentCatcher@PS\ArgumentCatcher@S % as above: we always act
‘long’ with single tokens.
```

```
%% \let\ArgumentCatcher@T\ArgumentCatcher@S % we allow T (for Token) as
```

```
%% % an alias of S.
```

```
%% \let\ArgumentCatcher@PT\ArgumentCatcher@S
```

```
6659 \edef\ArgumentCatcher@S
```

```
6660 #1% default value
```

```
6661 {%
```

```
6662 \@nx\ArgumentCatcher@T {\@xa\@nx\all@stars}{#1}}% the s arg spec is a short-
hand for T{*}. Except the star may be of any category from {11,12,13}.
```

```
6666 \@dc@AliasPLower S
```

```
6667 \@dc@AliasCatcher * S
```

```
\if@debugacro@ 6670 \newif\if@debugacro@
```

```
\ArgumentCatcher@Loop 6674 \long\def\ArgumentCatcher@Loop % Now we introduce a general iterating catcher.
The two parsers: “while” (Q) and ”until” (U) are special cases of it.
```

Moreover, clever tripling it generalises also **GBbOoCc** specifiers (with some inner macros relaxed for the first instance).

```
6681 #1% kind of test (StrX or str or anything else for what \gmu@if{??}any/none is de-
fined)
```

The catcher has four parameters that interact with one another:

```
6686 #2% “sign” of match for iteration: \any or \none token. \any{<#3>} means that
considering {<#3>} as set of tokens  $T_3$  for each token  $x$ ,  $x$  satisfies the condition
iff  $x \in T_3$ .
```

In case of #2 and #3 that means such  $x$  will be tried to be appended to the list, while in case of #4 and #5 such  $x$  will be thrown “down the drain”.

If #3 is empty, #2==\any means that *no* token will be appended to the list while \none means that *every* one will be. (Consider phrases “any thing belonging to the empty set” and “thing not belonging to the empty set”.)

Simili modo, when #5 is empty, #4==\any means *no* token will be dropped while \none means that *every* one will be.

```
6704 #3% list of tokens \@let@token will be matched against with #1 test and sustain iter-
ation or stop catching.
```

We assure in line ?? that when we condition iteration on presence of next token on #2 list, i.e. #1 is \any, we appended the “drainers” list to #2 so we won’t stop at them.

```
6712 #4% test for “drainers” (allowed values as for #1)
```

```
6714 #5% list of “drainers”: if \@let@token satisfies the #3 test, we throw it down the drain,
otherwise we add it to the argument.
```

The “iterate or stop” matching precedes the “add or throw” matching so if we meet a stopping token, surely it will not be added to the argument.

```
6721 #6% upper bound of number of items (may be empty or negative to turn counting of
items off).
```

For a moment I was tempted to try implement *any* loop condition, but that seems not to make sense, since during argument catching tokens are not executed so hardly any tests except matching items against a list or counting them seem to be reasonable.

This catcher is always `\long` but it'd be not the only danger of iterating beyond end of file: another would be allowing all three: `#1`, `#2`, `#4` empty at the same time, but we check that in the defaults' parser (line ??).

6736 `#7%` default value (if empty sequence is parsed), `\NoValue` by default.

6738 `#8%` an “each” — stuff to be put before each token/text added to the argument. Empty by default. If `#7` is empty and used (empty sequence parsed), `#8` is not put before it.

6741 `#9%` the tail of args parser

6742 `\@dc@arguments %` delimiter

6743 `{%`

6745 `\dc@DefParseNext{#9}%`

6746 `\emptify\@dc@sequence`

6747 `\c@dc@Loop@bound=\numexpr(\gmu@ifempty{#6}{-1}{#6})*1\relax`

Since `#6` is empty by default, the counter is set to `-1` by default (l. ??)

6751 `\gmu@if {num}{\c@dc@Loop@bound<\z@}%`

We look at the sign of the bound and if it's `-` we oppose the bound and make the decrease step `= 0`

6754 `{\c@dc@Loop@bound=-\c@dc@Loop@bound`

6755 `\let\@dc@Loop@countby\z@`

6756 `}%`

Otherwise we make the decrease step `-1`.

6758 `{\let\@dc@Loop@countby\none}%`

6760 `\gmu@ifempty{#3}%`

6761 `{\gmu@if {strings}{\none #2}%`

6762 `{\@dc@Loop@iteralltrue}%` it's an additional switch for better performance in a special case: when we wish only count the items and iterate over anything

6765 `{%`

Here we'd look for presence of the next token on an empty list, what cannot be. Therefore we stop. In this case the value of switch is irrelevant.

6768 `\Store@Macro\ArgumentCatcher@Loop@defcheck`

6769 `\def\ArgumentCatcher@Loop@defcheck{%`

6770 `\Restore@Macro\ArgumentCatcher@Loop@defcheck`

6771 `\ArgumentCatcher@Loop@shipout}%`

6772 `}%` of if `#2` str-is `\any`

6773 `}%` of if `#3` empty

6774 `{\@dc@Loop@iterallfalse}%` when `#2` nonempty, we *have* to perform matching.

We prepare shortcuts for adding/rejecting all analogously:

6778 `\gmu@ifempty{#5}%`

6779 `{\gmu@if {strings} {\none #4}%`

6780 `{\@dc@Loop@addallfalse\@dc@Loop@drainalltrue}%`

6781 `{\@dc@Loop@addalltrue\@dc@Loop@drainallfalse}%`

6782 `}%` of if `#5` empty

6783 `{\@dc@Loop@addallfalse\@dc@Loop@drainallfalse}%` when `#4` nonempty, we *have* to perform matching.

6786 `\edef\ArgumentCatcher@Loop@afterpeep{%`

This is the macro that'll be executed after the main `\futurelet`, We define it here since nothing in this sequence of tokens changes during catching, only the meaning of `\@let@token`.





```

\if@dc@Loop@addall 6861 \newif\if@dc@Loop@addall
\if@dc@Loop@drainall 6862 \newif\if@dc@Loop@drainall

6865 \long\def\@dc@Loop@test
6866 #1% kind of test (StrX or str so far (2010/12/28, 7.51))
6867 #2% \any or \none
6868 #3% list of tokens to match against
6869 #4% what if OK
6870 #5% what if not OK.
6871 {% remember we are in the argument of \gmu@peep@next so we are after peep.

6874 \gmu@if {defined}{\@def@token}
6875 {\csname gmu@if#1\strip@bslash #2\@xa\endcsname
6876 \@def@token }%
6877 {\csname

```

we construct csname of the quantifier-test. Depending on #1 it will apply \ifx or \if with \noexpands.

```

6880     gmu@if%
6881     \gmu@ifdetokens {#1}{str}%
6882     {NX}%
6883     {x}%
6884     \strip@bslash #2%
6885     \endcsname
6886     \@let@token }%
6887 {#3}{#4}{#5}%
6888 }

6893 \@dc@AliasCatcher {PLoop} {Loop}

6895 \long\def\ArgumentCatcher@Loop@defcheck
6896 #1% default,
6897 #2% “eachr”
6898 {%
6899     \edefU\ArgumentCatcher@Loop@check{%
6900         \gmu@if {num}{\c@dc@Loop@bound>\z@}

```

If we haven't reached maximum allowed number of parsed items, we peep the next token, i.e. we perform \futurelet and if \@let@token is undefined or \relax then we pass it inside \@def@token so that string comparisons may look at it.

```

6907     {\advance \c@dc@Loop@bound \@dc@Loop@countby
6908         \gmu@peep@next
6909         \ArgumentCatcher@Loop@afterpeep }

```

Otherwise we finish catching and send the argument to the arguments' toks.

```

6913     {\ArgumentCatcher@Loop@shipout {#1}{#2}}% of if num bound reached
6914 }% of \ArgumentCatcher@Loop@check
6915 }% of \ArgumentCatcher@Loop@defcheck

6918 \lpdef\@dc@Loop@CareOfSpace
6919 #1% a macro with all the arguments except last
6920 {\gmu@ifpeeped x \gmu@letspace
6921     {\edef\@dc@Loop@careofspace@aas{%
6922         \unexpanded{#1{ }}%
6923     }}% edef unexpanded to allow # tokens in #1

```

```

6924     \afterassignment\@dc@Loop@careofspace@aas
6925     \let\gmu@drain= }% after = is a space.
6926     {#1}% next not space, so just #1 and let it process the next token its way.
6928 }

6931 \StoreMacro@nocat\@dc@Loop@CareOfSpace

6934 \lpdef\@dc@Loop@CareOfSpace@NoBraces
6935 #1% a macro with all the arguments except last
6936 {%
6938 \gmu@ifpeeped x \gmu@letspace
6939 {\edef\@dc@Loop@careofspace@aas{%
6940     \unexpanded{#1 }% note the space
6941     }% edef unexpanded to allow # tokens in #1
6942     \afterassignment\@dc@Loop@careofspace@aas
6943     \let\gmu@drain= }% after = is a space.
6944     {#1}% next not space, so just #1 and let it process the next token its way.
6946 }

%% \lpdef\@dc@Loop@CareOfGroup
%% #1% as for the previous macro
%% {\gmu@if x{\@let@token\bgroup}%
%%   {\gmu@passbraced{#1}}% if next bgroup
%%   {#1}% not \bgroup, so we get it normally
%% }% of \@dc@Loop@CareOfGroup No need of the above (commented out): \gmu@passbraced
does that.

6959 \lpdef\@dc@Loop@CareOfSpaceAndGroup
6960 #1% as in the previous macro
6961 {\@iwruif {Care of both: @let@token: »\meaning\@let@token«}%
6962   \@dc@Loop@CareOfSpace{%
6963     \gmu@passbraced{#1}%
6964   }%
6965 }% of “take care of both”

6968 \long\def\ArgumentCatcher@Loop@add
6969 #1% default
6970 #2% eacher
        At this level we pass both default and eacher for symmetry of the macro(s) at higher
        level.

6974 {\@dc@Loop@CareOfSpaceAndGroup
6975   {\ArgumentCatcher@Loop@add@{#2}}%
6976 }

6978 \long\def\ArgumentCatcher@Loop@add@

        But here we can limit ourselves to what we really need.

6980 #1% eacher
6981 #2% token/text to be added
6982 {\addtomacro\@dc@sequence{#1#2}%
6983   \@iwruif {##### Q-add: »\the\@dc@arguments«}%
6984   \ArgumentCatcher@Loop@check
6985 }% of \ArgumentCatcher@Loop@add@

6988 \long\def\ArgumentCatcher@Loop@drain
6989 #1% default

```

6990 #2% eacher

At this level we pass both default and eacher for symmetry of the macro(s) at higher level.

```
6994 {%
6995   \@iwruif{@@@@@ Q-drain: »\the\@dc@arguments}%
6996   \@dc@Loop@CareOfSpaceAndGroup
6997   {\@xa\ArgumentCatcher@Loop@check\@gobble}%
6998 }

7001 \long\def\ArgumentCatcher@Loop@shipout
7002 #1% default
7003 #2% eacher
```

This macro needs both of these parameters.

```
7006 {%
7007   \gmu@if x{\@dc@sequence\@empty}%
7008   {\def\@dc@sequence{#1}%
```

Thus we put the default value to the temporary macro. Now, it's nonempty, we prepend to it the "eacher":

```
7011   \gmu@if x{\@dc@sequence\@empty}%
7012   {}{\prependtomacro\@dc@sequence{#2}}%
7013 }%
7014 {}% if \@dc@sequence is nonempty, we don't modify it.
7015 \@xa\@dc@addargum\@xa{\@xa{\@dc@sequence}}%
7016 \@iwruif {@@@@@ Q-finish: »\the\@dc@arguments}%
7017 \@dc@parse@next\@dc@arguments
7018 }% of \ArgumentCatcher@Loop@shipout
```

```
7021 \long\def\ArgumentCatcher@Q
7022 #1% list
7023 #2% default (\NoValue by default, as in previous versions).
7024 {%
7025   \ArgumentCatcher@Loop
7026   {StrX}%
7027   \any
7028   { #1}% note the space before #1—settheory sum of #1 and drainers.
7029   \any
7030   { }% we ignore spaces—it's a legacy setting, maybe we'll optionise it in the future
7031   \m@ne % we allow any number of tokens
7032   {#2}% default, by default \NoValue, as defined in line ??
7033   {}% empty eacher
7034 }%
7035 }%
```

```
7037 \@dc@AliasPLower Q
```

```
7039 \@ifXeTeX {%
7040   \long\def\ArgumentCatcher@O
7041   #1% list
7042   #2% default (\NoValue by default, as in previous versions).
7043   {%
7044     \@dc@BareSpaceArg
7045     \ArgumentCatcher@Loop
7046     {str}%
```

```

7047 \any
7048 { #1}% note the space before #1—settheory sum of #1 and drainers.
7049 \any
7050 { }% we ignore spaces—it’s a legacy setting, maybe we’ll optionise it in the future
7052 \m@ne % we allow any number of tokens
7053 {#2}% default, by default \NoValue, as defined in line ??
7054 {}% empty eachr
7055 }%

7057 \@dc@AliasPLower Ô
7059 }{}% of if XeTeX

7062 \def\ArgumentCatcher@D{% decimal argument (useful for dates e.g.)
7063 \ArgumentCatcher@Q
7064 {-+0123456789}%

```

the default value will be delivered by the defaults’ catcher

```

7066 }

7068 \@dc@AliasPLower D

7070 \@ifXeTeX
7071 {%
7072 \def\ArgumentCatcher@D

```

The difference between D and D̄ catchers is that the latter stops at a blank while the former doesn’t.

```

7075 #1% default value, 0 by default
7076 {% decimal argument
7077 \ArgumentCatcher@Loop
7078 {StrX}%
7080 \any
7081 {-+0123456789}%
7082 \any
7083 {}%
7084 \m@ne
7085 {#1}%
7086 {}%
7087 }%

```

```

7089 \@dc@AliasPLower D̄
7091 }{}% of if XeTeX

7093 \def\ArgumentCatcher@U {%
7094 \ArgumentCatcher@Loop {StrX}%
7095 }

```

Yes, because *de facto* we pass it *all* the arguments of catcher Loop.

```

7099 \@dc@AliasPLower U
7102 \@dc@AliasCatcher W {U}
7104 \@dc@AliasPLower W
7107 \@dc@AliasCatcher {lostwax} U % The \lostwax catcher is a normal “until” catcher.
    It’s parsing of the specifier’s parameters that makes it different.

```

```

7111 \long\def\ArgumentCatcher@item
7112 #1% “sign” of matching
7113 #2% list to match against
7114 #3% default value
7115 #4% eacher (which in this case will be applied to the (one) item)
7117 {%
7118   \ArgumentCatcher@Loop
7119   {StrX}%
7120   #1%
7121   {#2}%
7122   \none {}% we add all that matches
7123   1 % at most one item
7124   {#3}%
7125   {#4}%
7126 }
7128 \if@XeTeX
7130   \def\ArgumentCatcher@Ü{%
7131     \@dc@BareSpaceArg
7132     \ArgumentCatcher@Loop {StrX}%
7133   }
7135   \@dc@AliasPLower Ü
7137 \fi

```

TODO: edef prefixing.

```

7141 \long\def\ArgumentCatcher@genM
7142 #1% the letter »P« (for long catcher) or nothing (for the short version)
7143 #2% number of undelimited parametrs to catch (arabic)
7144 #3% tail of parsers
7145 \@dc@arguments{%
7146   \dc@DefParseNext{#3}%
7147   \csname ArgumentCatcher@#1m@%
7148   \romannumeral#2%
7149   \endcsname
7150 }

```

They both have to be long for the tail of parser that can contain `\par`. It’s the *inner* catchers `...@(P)m@ii...` that are short or long.

```

7156 \def\ArgumentCatcher@M{\ArgumentCatcher@genM {}}
7157 \def\ArgumentCatcher@PM{\ArgumentCatcher@genM P}
7159 \@dc@AliasCatcher {m} {M}
7160 \@dc@AliasCatcher {Pm} {PM}

```

We allow specifying mandatory arguments also as `(M|m){\langle num\rangle}` and such a specifier is passed to the carrier macro.

```

7166 \@dc@AliasCatcher {ms} {M}
7167 \@dc@AliasCatcher {Pms} {PM}
7170 \@tempcnta=\@ne
7171 \@whilenum\@tempcnta<9\do{%
7173   \edef\gmu@tempa{%

```

The short catchers of mandatories:

```

7176 \def\@xanxcs{%
7177   ArgumentCatcher@m@\romannumeral\@tempcnta}%
7178 \gmu@hashes1{\numexpr\@tempcnta+1}%
7179 {\@nx\@dc@addargum{%
7180   \gmu@hashesbraced 1{\numexpr\@tempcnta+1}}}%
7181 \@nx\@dc@parse@next \@nx\@dc@arguments
7182 }% of \ArgumentCatcher@m@(rom.num).

```

And the long ones:

```

7185 \long \def\@xanxcs{%
7186   ArgumentCatcher@Pm@\romannumeral\@tempcnta}%
7187 \gmu@hashes1{\numexpr\@tempcnta+1}%
7188 {\@nx\@dc@addargum{%
7189   \gmu@hashesbraced 1{\numexpr\@tempcnta+1}}}%
7190 \@nx\@dc@parse@next \@nx\@dc@arguments
7191 }% of \ArgumentCatcher@Pm@(rom.num).

7193 }% of \edef.
7194 \gmu@tempa
7195 \advance\@tempcnta1\relax
7196 }% of the loop.

```

The loop above defines 8 pairs of macros as we see thanks to the code below:

```

7200 \if 1 1
7201 \@tempcnta\@ne
7202 \@whilenum\@tempcnta<9\do{%
7203   \typeout{\@slash ArgumentCatcher@m@\romannumeral\@tempcnta:
7204     ^J}%
7205   \Name\meaning{ArgumentCatcher@m@\romannumeral\@tempcnta}^^J}%
7206   \typeout{\@slash ArgumentCatcher@Pm@\romannumeral\@tempcnta : ^^J}%
7207   \Name\meaning
7208   {ArgumentCatcher@Pm@\romannumeral\@tempcnta}^^J^^J^^J}%
7209   }%
7210 \advance\@tempcnta\@ne
7211 }

%% \show\ArgumentCatcher@Pm@viii

7213 \fi

```

The code above produces on the terminal (without the blank between 1's of course):

```

\ArgumentCatcher@m@i:
macro:#1->\@dc@addargum {{#1}}\@dc@parse@next \@dc@arguments

\ArgumentCatcher@Pm@i:
macro:#1->\@dc@addargum {{#1}}\@dc@parse@next \@dc@arguments

\ArgumentCatcher@m@ii:
macro:#1#2->\@dc@addargum {{#1}{#2}}\@dc@parse@next \@dc@arguments

\ArgumentCatcher@Pm@ii:
macro:#1#2->\@dc@addargum {{#1}{#2}}\@dc@parse@next \@dc@arguments

\ArgumentCatcher@m@iii:
macro:#1#2#3->\@dc@addargum {{#1}{#2}{#3}}\@dc@parse@next
\@dc@arguments

```

```

\ArgumentCatcher@Pm@iii:
macro:#1#2#3->\@dc@addargum {{#1}{#2}{#3}}\@dc@parse@next
    \@dc@arguments

\ArgumentCatcher@m@iv:
macro:#1#2#3#4->\@dc@addargum {{#1}{#2}{#3}{#4}}\@dc@parse@next
    \@dc@arguments

\ArgumentCatcher@Pm@iv:
macro:#1#2#3#4->\@dc@addargum {{#1}{#2}{#3}{#4}}\@dc@parse@next
    \@dc@arguments

\ArgumentCatcher@m@v:
macro:#1#2#3#4#5->\@dc@addargum {{#1}{#2}{#3}{#4}{#5}}\@dc@parse@next
    \@dc@argu
ments

\ArgumentCatcher@Pm@v:
macro:#1#2#3#4#5->\@dc@addargum {{#1}{#2}{#3}{#4}{#5}}\@dc@parse@next
    \@dc@argu
ments

\ArgumentCatcher@m@vi:
macro:#1#2#3#4#5#6->\@dc@addargum
    {{#1}{#2}{#3}{#4}{#5}{#6}}\@dc@parse@next \@d
c@arguments

\ArgumentCatcher@Pm@vi:
macro:#1#2#3#4#5#6->\@dc@addargum
    {{#1}{#2}{#3}{#4}{#5}{#6}}\@dc@parse@next \@d
c@arguments

\ArgumentCatcher@m@vii:
macro:#1#2#3#4#5#6#7->\@dc@addargum
    {{#1}{#2}{#3}{#4}{#5}{#6}{#7}}\@dc@parse@ne
xt \@dc@arguments

\ArgumentCatcher@Pm@vii:
macro:#1#2#3#4#5#6#7->\@dc@addargum
    {{#1}{#2}{#3}{#4}{#5}{#6}{#7}}\@dc@parse@ne
xt \@dc@arguments

\ArgumentCatcher@m@viii:
macro:#1#2#3#4#5#6#7#8->\@dc@addargum
    {{#1}{#2}{#3}{#4}{#5}{#6}{#7}{#8}}\@dc@pa
rse@next \@dc@arguments

\ArgumentCatcher@Pm@viii:
macro:#1#2#3#4#5#6#7#8->\@dc@addargum
    {{#1}{#2}{#3}{#4}{#5}{#6}{#7}{#8}}\@dc@pa
rse@next \@dc@arguments

```

7292 \def\ArgumentCatcher@m@ix

And it doesn't need to be long and launch inner short macro because there's nothing more to catch.

7296 #1#2#3#4#5#6#7#8#9{%

7297 \@dc@addargum{%



```

7298     {#1}{#2}{#3}{#4}{#5}{#6}{#7}{#8}{#9}}%
7299     \the \@dc@arguments
7300 }

7302 \long\def\ArgumentCatcher@Pm@ix
7303 #1#2#3#4#5#6#7#8#9{%
7304     \@dc@addargum{%
7305         {#1}{#2}{#3}{#4}{#5}{#6}{#7}{#8}{#9}}%
7306     \the \@dc@arguments
7307 }

7310 \long\def\ArgumentCatcher@K % a Knuthian delimited or undelimited arguments
        parsed and passed to the command's body in a digest (Knuthian replacement)
7313 #1% the particular Knuthian macro
7314 #2% the tail of args parser.
7315 \@dc@arguments % tail delimiter
7316 {\dc@DefParseNext{#2}#1}

7318 \@dc@AliasPLower K

7321 \long\def\gmu@twostring#1#2{\string#1\string#2}

7324 \long\def\ArgumentCatcher@G@longorshort
7325 #1% longness prefix (\long or nothing)
7326 #2% left and right delimiters
7327 #3% default value
7328 #4% the tail of args parser.
7329 \@dc@arguments % tail delimiter
7330 {%
7331     \edef\@dc@Gname{ArgumentCatcher@G\gmu@twostring#2}%
7332     \gmu@if {curname}{\@dc@Gname \endcsname}%

    If catcher for this particular pair of delimiters is defined, we don't repeat definition (we
    assume nobody else would define such a curname).

7337     }%

    If it's undefined, we define: \ArgumentCatcher@G<stringed delimiters>

7339     {#1\Name\def{\@dc@Gname \@xa}%
7340         \@dc@Gparams#2{\@dc@addargum{##2}}##1\@dc@arguments}%
7341     }%
7342     \@xa\@ifEUnextchar\@firstoftwo#2%
7343     {\def\dc@parse@next{#4}% we hide possible \pars.
7344         \curname \@dc@Gname \endcsname\dc@parse@next}%
7345     {\@dc@addargum{#3}#4\@dc@arguments}%
7346 }

7348 \long\def\@dc@Gparams#1#2{##1#1##2#2}

    This macro expands to parameters string with #1 delimited with first argument and #2
    delimited with second. In fact it's intended to pass #1 further (it will always be braced)
    and to catch an argument put in a pair of tokens given \@ dc@Gparams as the arguments.

7356 \lpdef\ArgumentCatcher@G % short general catcher
7357 #1% left and right delimiters
7358 #2% default value
7359 {\ArgumentCatcher@G@longorshort}{#1}{#2}}

7361 \lpdef\ArgumentCatcher@PG % long general catcher

```

```

7362 #1% left and right delimiter
7363 #2% default value
7364 {\ArgumentCatcher@G@longorshort{\long}{#1}{#2}}

```

Now the “canonical” catchers as special cases of G:

```

7367 \def\@dc@DefAsGeneral
7368 #1% specifier
7369 #2% delimiters
7370 {\Name\lpdef{ArgumentCatcher@\strip@bslash{#1}}%
7371 ##1% default value
7372 {\ArgumentCatcher@G{#2}{##1}}%

```

And the long version:

```

7374 \Name\lpdef{ArgumentCatcher@P\strip@bslash{#1}}%
7375 ##1% default value
7376 {\ArgumentCatcher@PG{#2}{##1}}% »P« was missing. Fixed 2011/03/22, 16.14.
7378 \gmu@if {cat}{a\@nx#1}%
7379 {\@dc@LowerAliases #1}{}%
7380 }
7382 \@dc@DefAsGeneral A{<>}
7385 \@dc@DefAsGeneral O{[]}]

```

For legacy reasons we treat the () argument differently: the ancient **xparse** processed a parenthesised argument to a pair of braces splitting it on a comma. We leave this as a possibility with the `\DCcoordinate` declaration:

```

7392 \def\DCnocoordinate{\@dc@DefAsGeneral C{()}}
7393 \DCnocoordinate
7395 \edefU\DCcoordinate{%
7397 \long\def\ArgumentCatcher@C
7398 #1% default value
7399 #2% tail of parsers
7400 \@dc@arguments % tail's delimiter
7401 {%
7402 \@ifEUnextchar(%
7403 {\dc@DefParseNext{#2}\ArgumentCatcher@c@}%
7404 {\@dc@addargum{#1}#2\@dc@arguments}%
7405 }%
7407 \let\ArgumentCatcher@c\ArgumentCatcher@C
7408 \let\ArgumentCatcher@Pc\ArgumentCatcher@PC
7410 \long\def\ArgumentCatcher@PC#1#2\@dc@arguments{%
7411 \@ifEUnextchar(%
7412 {\dc@DefParseNext{#2}\ArgumentCatcher@Pc@}%
7413 {\@dc@addargum{#1}#2\@dc@arguments}%
7414 }%
7416 \def\ArgumentCatcher@c@(#1){%
7417 \gmu@ifxany,{#1}%
7418 {\@xa\@dc@addargum\@dc@commasep#1\@dc@commasep}%
7419 {\@dc@addargum{#1}}\@dc@parse@next\@dc@arguments
7420 }%
7422 \def\@dc@commasep#1,#2\@dc@commasep{{{#1}{#2}}}%

```

```

7424 \long\def\ArgumentCatcher@Pc@(#1){%
7425 \gmu@ifxany,{#1}%
7426 {\@xa\@dc@addargum\@dc@commasep#1\@dc@commasep}%
7427 {\@dc@addargum{#1}}\@dc@parse@next\@dc@arguments
7428 }%
7429 }

```

2010/7/21 A scope prefix argument

```

7435 \def\@dc@Scope@shipout#1{%
    The arg. is the default value (if argument not provided).
7437 \ifx\@dc@seSequence\@empty\def\@dc@seSequence{#1}\fi
7438 \@xa \gmu@ifxany\@xa \global\@xa{\@dc@seSequence}%
7439 {\@dc@addargum{\global}}}%
7440 {\@dc@addargum{\relax}}}%
7441 \@dc@parse@next\@dc@arguments
7442 }

```

```

7444 \def\ArgumentCatcher@Scope
7445 #1% the separator(s)
7446 {\let\@dc@seSequence@shipout@@\@dc@seSequence@shipout
7447 \let\@dc@seSequence@shipout\@dc@Scope@shipout
7448 \ArgumentCatcher@Q{\global\relax}{\relax}%
7449 \let\@dc@seSequence@shipout\@dc@seSequence@shipout@@
7450 \@dc@ignorearg
7451 \ArgumentCatcher@T{#1}{\NoValue}%
7452 }

```

```

7455 \@dc@AliasCatcher {PScope} \Scope
    [End of Argument Specifiers' Definitions]

```

```

7463 \long\def\gmuIfNoValueTF#1{%
7471 \@xa@ifx\@xa\NoValue\@firstofmany#1\@empty\@nil \afterfi\@firstoftwo
7472 \else\afterfi\@secondoftwo
7473 \fi}

```

```

7475 \long\def\gmuIfNoValueT#1#2{\gmuIfNoValueTF{#1}{#2}\@empty}

```

```

7477 \long\def\gmuIfNoValueF#1#2{\gmuIfNoValueTF{#1}\@empty{#2}}

```

```

7479 \long\def\gmuIfValueTF#1#2#3{\gmuIfNoValueTF{#1}{#3}{#2}}

```

```

7481 \long\def\gmuPutIfValue#1{\gmuIfValueT{#1}{#1}}

```

```
\gmuIfValueT 7484 \let\gmuIfValueT\gmuIfNoValueF
```

```
\gmuIfValueF 7487 \let\gmuIfValueF\gmuIfNoValueT
```

```

7489 \long\pdef\IfLong#1{%
    % #1 the argument to check for \par,
    % #2 what if \par found,
    % #3 what if \par not found.
7496 \edef\gmu@ifLong@resa{\detokenize{#1}}%
7497 \edef\gmu@ifLong@resa{%
7498 \IfAmong\string\par\@nx\among{\gmu@ifLong@resa}}%
7499 \gmu@ifLong@resa}

```

= (\afterassignment) pseudo-argument

```

7504 \long\@namedef{ArgumentCatcher@=} %
7505 #1% register used in the assignment
7506 #2% the tail of args parser.
7507 \@dc@arguments{%
7508   \dc@DefParseNext{%
7509     \@dc@addargum\NoValue % to make ignoring mechanism work
7510     #2\@dc@arguments}%
7511   \afterassignment\@dc@parse@next
7512 #1}% optional space and = is left at user's discretion. In fact, #1 may be empty and
       then each time our command is used the left side of the assignment has to be
       given explicitly and may even be different each time.

```

```

7517 \n@melet{ArgumentCatcher@P=} {ArgumentCatcher@=}

```

\gobblespace pseudo-argument: useful between optional and Knuthian-type arguments and after all parsing. Equivalent `>iT{\somedummymacro}`. At first I wanted mark it `\ignorespaces` but the gobbling doesn't expand macros: it doesn't use `\ignorespaces` but `\ifnextchar`'s space trick.

```

7526 \long\def\ArgumentCatcher@gobblespace
7527 {\ArgumentCatcher@T {\dc@gobblespace@dummy}{\NoValue}}
7529 \let\ArgumentCatcher@Pgobblespace\ArgumentCatcher@gobblespace
7531 \def\dc@gobblespace@dummy{\dc@gobblespace@dummy}
7533 \let\gobblespace\dc@gobblespace@dummy
       end of catchers' definitions

```

```

7538 \errorcontextlines=100

```

```

7542 \long\def\@dc@InnerName#1{%
7543   \bslash@or@ac{#1}%
7544   \bslash
7545 }%

```

```

7547 \long\def\@dc@InnerEndName #1{% version for the end-macros of environments
7548   \bslash end\strip@bslash{#1}%
7549 }%

```

Name of the CS carrying the command's specifiers sequence and arity of the inner macro.

(2010/07/26, v0.993:) This strikingly simple idea of storing the specifiers' sequence in a macro came to my head only after some three years of developing `\DeclareCommand`

```

7558 \long\edef\@dc@SpecsArName#1{%
7559   Specs \string& arity of
7560   \@nx\bslash@or@ac{#1}%
7561 }%

```

```

7565 \pdef\@dc@messages {%
7572   \gmu@notif {@dc@quiet@}{}%
7573   {%
7574     \gmu@if {DCMessages}{}%
7575     {\PackageInfo{gmcommand}{^^J%
7576       Parsing arguments for \dc@innername^^J%
7577       [specified as:
7578       \unexpanded

```

```

7579      \@xa\@xa\@xa \@xa\@xa\@xa \@xa{%
7580      \@xa\@xa\@xa\@secondofmany

```

not \@secondofthree because first use of this definition is for \DeclareCommand not having its specifiers' carrier.

```

7585      \csname \@dc@specsname\endcsname {}{}\@nil }%
7586      \space ]^^J%
7587      [in file: \@currname.\@current\space ]}% of info message
7588      }% of if DC messages
7589      {}% of if not DCMessages
7590      }% of if not @dc@quiet@
7591      {}% of if not not @dc@quiet@
7592      }% of \@dc@messages

```

(2010/07/15, v0.993:) added to modify naming of the inner macro of a \DeclareCommand ed commands: preceding the names with a backslash is OK for the letter CS'es, but it may cause a collision for the active chars. Therefore we both precede the name *and* succede it with a bslash

```

7603 \edefU\@dc@DCbody {%
7604 {%
7610 \edef\dc@innername {\@dc@InnerName #1}% we'll use it in K-type arg. catcher
      and to allow #1 == \par ( \PackageWarning is short and \typeout too!).
      And in \lostwax.

```

We define the name of the macro carrying specs and inner arity:

```

7616 \edef\@dc@specsname {\@dc@SpecsArName{#1}}%
7618 \gmu@ifCSdefined {#1}%
7619 {\gmu@if{DCMessages}}}%
7620 {\PackageWarning{gmcommand}}%
7621      ^^J%
7622      Redefining command \dc@innername\space
7623      with \string\DeclareCommand}%
7624      }%
7625      {}%
7626      }%
\gmu@if 7627 {\gmu@if {DCMessages}}}%
7628      {\PackageInfo{gmcommand}}^^J%
\dc@innername 7629 \string\DeclareCommand-ing \dc@innername\space (new)^^J}%
7630      }%
7631      {}%
7632      }%

```

First we parse the declaration options to know whether all the arguments are to be long and whether we should suppress the message "Parsing arguments for...".

```

7637 \@dc@alllong@false
7638 \@dc@quiet@false

```

Now we collect the prefixes.

```

7642 \emptify\@dc@Specs@tempa
7644 \gmu@ifsbintersect{#2}{\long!LL}}%
7645 \addtomacro\@dc@Specs@tempa{\long}%
7646 \@dc@alllong@true}%
7647 {}%

```

```

7649 \gmu@ifsbintersect{#2}{.qQ\quiet}{%
7650 \addtomacro\@dc@Specs@tempa{\quiet}%
7651 \@dc@quiet@true}%
7652 }%
7654 \gmu@ifsbintersect{#2}{iIwW\sphack}%
7655 {% if 'invisibility' is specified:
7656 \addtomacro\@dc@Specs@tempa{\sphack}%
7657 \def\@dc@bsphack@{\@nx\@bsphack}%
7658 \def\@dc@esphack@{\@nx\@esphack}%
7659 }%
7660 {\emptify\@dc@bsphack@
7661 \emptify\@dc@esphack@}% if 'invisibility' is not specified.
7663 \gmu@ifsbany\envhack{#2}%
7664 {% but if we define an environment:
7665 \addtomacro\@dc@Specs@tempa{\envhack}%
7666 \emptify\@dc@bsphack@
7667 \emptify\@dc@esphack@
7668 \edef\@dc@setThrowArgs{%
7669 \dc@ThrowArgs{\string#1}}% if our command is used as an environment, we
       throw the args to the end in a toks register (primary version defined a
       macro but that failed for #es).
7672 }%
7673 {% and if we don't define an environment:
7674 \emptify\@dc@setThrowArgs
7675 }%
7677 \relaxen\@dc@global@
7678 \relaxen\@dc@outer@
7680 \gmu@ifsbany\outer{#2}%
7681 {\addtomacro\@dc@Specs@tempa{\outer}%
7682 \let\@dc@outer@\outer}%
7683 }%
7685 \gmu@ifsbany\global{#2}{%

```

We store the information about globalness of definition for the future but in an inactive form (globalness seems to me something very local)

```

7689 \addtomacro\@dc@Specs@tempa {\IAMDeclaredGlobally}%
7690 \let\@dc@global@\global}%
7691 }%

```

Now the possible prefixes are processed.

We add the (distilled version of) them to the storage macro as its first item:

```

7696 \@xa\Name \@xa\edefU
7697 \@xa\@dc@specsname
7698 \@xa{\@xa{\@dc@Specs@tempa}}%

```

We initialise the scratch count and toks registersa and the Boolean switches before parsing of the arguments specifiers.

```

7703 \@dc@ResetParseAuxilia

```

We parse the arguments' specifiers:

```

7706 \@dc@ParseNextSpecifier #3%
7707 \@dc@buckstopshere % it's the sentinel of parsing. Since the test performed by
       \@dc@ParseNextSpecifier is a comparison of (bslash-stripped) strings, this
       CS doesn't have to be defined or have any particular meaning.

```

```

7713 \@xa \Name \@xa \addtomacro \@xa \@dc@specsname
7714 \@xa {\@xa{\the\@dc@ParsedSpecs}}%

```

We add the inner arity (braced) to the specs-carrying macro

```

7722 \@xa\Name\@xa\addtomacro\@xa\@dc@specsname
7723 \@xa{\@xa{\the \c@dc@argnum}}%

```

Now we define the basic macro:

```

7726 \@dc@outer@\@dc@global@ % possibly the prefixes,

```

Here comes the definition of the coating macro, named the same as the command:

```

7730 \protected\edef#1{% always \protected ;-)}
7731 \@dc@bsphack@ % perhaps \@bsphack

7732 \@dc@messages
7733 \@nx\@dc@\the\toks@}% in the braces appear the argument catchers.
7734 \@xanxcs{\dc@innername}%
7735 \ifx\@dc@outer@\outer
7736 \@xanxcs{\@xa\gobble\string#1 }% note the blank space after #1! If the
7737 command is to be \outer, we can't put it itself, so we put another, whose
7738 name is the same plus a space. Anyway, since #1 becomes \protected,
this case will never be executed.

7739 \else \@nx #1%
7740 \fi
7741 }% of main coating macro's \edef.

7742 \edef\gmu@DeclareCommand@resa{%
7743 \@dc@global@ % perhaps \global
7744 \long\def\@xanxcs{\dc@InnerName{#1}}% for a command \command,
7745 define \command\
7746 \@xau@\dc@innerhashes % it's #1#2#3...
7747 {% the body of the inner '\langle name \rangle' macro:
7748 \@dc@setThrowArgs
7749 \unexpanded{#4}%
7750 \@dc@esphack@}%
7751 }% of \edef.
7752 \gmu@DeclareCommand@resa
7753 }% of \DeclareCommand's body.
7754 }

7755 \def\@dc@DCprefixQlist{\global\protected\outer\long
7756 \relax % for the somewhat perverse case when \@dc@global may be relaxed (in
7757 \DeclareEnvironment). If someone might want to redefine a cs let to relax,
7758 it anyway comes as the first argument. And \relax is not a proper value for
7759 the third argument (args. spec.) so it's all right at this point. (2010/6/10)
!LL.qQiIwW\sphack\envhack}

7760 \long\def\ShowCommand #1{%
7761 \@iwruJ
7762 \@iwru{\string#1« is »\meaning#1«}%
7763 \@iwruJ
7764 \@iwru{\string\Name\string{\dc@InnerName{#1}}« is
7765 »\Name\meaning{\dc@InnerName{#1}}«}%
7766 \@iwruJ
7767 \@iwru{\string\Name\string{\dc@SpecsArName{#1}}« is

```

```

7780     >>\Name\meaning{\@dc@SpecsArName{#1}}<<}%
7781     \@iwruJ
7782     \show\show
7783 }

```

`\DeclareCommand` 7786 `\@XA{\DeclareCommand\DeclareCommand` % This is the final version by now. Note we introduce the ‘prefixes’ at the #2 position only at this point so we have yet to prefix every argument specifier separately. Note also we only here ‘teach’ `\DeclareCommand` to pass the name of its main argument in the `\dc@innername` macro.

```

7793 {
7794 #1 >Pm          % command to be defined (may be even \par if you really wish),
7796 #2 >\@xa Q{\@dc@DCprefixQlist}{} % an optional seQquence of  $\epsilon$ -TeX’s prefixes
    and/or ! or l or L for ‘all long’ and/or . or q or Q for ‘quiet’ (message
    about calling the command is suppressed) and/or i or I as ‘invisible’ or W or
    w as ‘white’ or \sphack to make the command ‘invisible’ in the leading with
    the \bsphack... \esphack. To deal with an ‘invisible’ environment, there
    is one more acceptable value of this argument: \envhack, which suppresses
    placing \bsphack—\esphack and places \@ignore@true instead in the end
    macro,
7810 #3 >Pm % arguments specification (may contain \par),
7811 #4 >Pm % the definition body; may contain nearly anything including \par and tests
    for presence/absence of arguments
    % \If[No]Value(T|F|TF) and for their longness \IfLong; you refer to the ar-
    guments with #<n>.
7815 }}\@dc@DCbody

```

Now the inner body of `\DeclareCommand` is full-featured but its specifiers’ carrier is not yet defined. Therefore we repeat its definition with the almost-fullfeatured version.

Now it’s the fixed point.

```

\DeclareCommand 7827 \@XA{\DeclareCommand \DeclareCommand \long
7828 {m >\@xa Q{\@dc@DCprefixQlist}{} mm}}%
7829 \@dc@DCbody
7838 \long\def\dc@EAName#1{% Env. arguments’ toks’ name It will be passed a stringed
    or bslashless name of a command
7840 \if\bslash#1\else#1\fi's_args}
7842 \pdef\dc@ThrowArgs#1{%
7843 \gmu@if{csname}{\dc@EAName{#1}\endcsname}
7844 }% if it’s defined, we do nothing (we assume it’s defined by us and is toks register.)
7848 {% else we define it as a new toks
7850 \@xa\newtoks\csname\dc@EAName{#1}\endcsname
7851 }%
7852 \csname\dc@EAName{#1}\endcsname=%
7853 \@xa\@xa\@xa{\@xa@gobble\the\dc@arguments}%
7854 }%

```

Now we have a convenient tool to implement parsing of a Knuthian argument(s).

Knuthian argument’s default replacement

```

7862 \def\dc@K@defaultrepl{##1}
    2009/11/22 inner pair of braces removed (as leading to additional group causing errors).

```

```

\@dc@define@K 7867 \DeclareCommand\@dc@define@K \long {mb}}%

```



```

%% \dc@xadefaultfalse First we add the particular CS to the toks register:
7870 \edef\gmu@tempa{%
7871 \unexpanded{\toks@\@xa}%
7872 {\@nx\the\toks@}%
7873 \@xanxcs{\dc@innername @K\the\c@dc@catchernum}%
7874 }%
7875 }%

```

here we add to the parsers toks list a CS named  $\langle our\ command \rangle @K \langle num.\ of\ catcher \rangle$  (the particular Knuthian catcher)

```
7878 }\gmu@tempa
```

and define this macro:

```
7880 \edef\gmu@tempa{%
```

And now we define that particular Knuthian catcher.

```

7882 \def\@xanxcs{\dc@innername @K\the\c@dc@catchernum}%
7883 \gmu@if{@dc@xadefault@ii@}{}%
7884 {\@xau{#1}}%
7885 {\unexpanded{#1}}%
7886 {%
7887 \@nx\dc@addargum{%
7888 \gmuIfValueTF{#2}{%
7889 \gmu@if{@dc@xadefault@ii@}{}%
7890 {\@xau{#2}}%
7891 {\unexpanded{#2}}%
7892 }{\@xau{\dc@K@defaultrepl}}%
7893 }}% we add the Knuthian replacement to the toks register as #n.
7895 \@nx \dc@parse@next \dc@arguments
7896 }% and continue parsing.
7897 }% of \edef. Now we execute this temporary macro, which means we \def\langle command \rangle @K \langle n \rangle
7898 \@dc@global@ % set properly before \dc@ParseNextSpecifier #3 \langle buck \rangle.
7899 \if P\dc@long@letter\relax\long\fi
7900 \gmu@tempa
7902 \dc@addtospecs {#1}%
7904 \gmuIfValueTF{#2}{%
7905 \gmu@if{@dc@xadefault@ii@}{}%
7906 {\@xa\dc@addtospecs\@xa{#2}}%
7907 {\dc@addtospecs{#2}}%
7908 }
7909 {\@xa\dc@addtospecs\@xa{\dc@K@defaultrepl}}%

```

Now we clean up and continue construction of arguments parser.

```
7912 \dc@ParseNextSpecifier
```

```
7913 }% of \dc@define@K
```

```

\dc@grab@Loop 7917 \DeclareCommand\dc@grab@Loop \long {%
7918 #1 T{\any \none Uu\U \u Ũ\Ū \ŭ Ww\W \w \lostwax} % the “sign” of match-
ing for adding; use of the letters suppresses adding them to the parsed spec-
ifiers’ list, which is crucial for \SameAs copying of specifiers: the “until” and
“while” loops don’t shift their arguments.
7924 #2 >Pm % list of tokens to match for/against iteration
7926 >i T{\drain \drop \Drain \Drop \lost \Lost}

```

```

7927 #3 T{\any \none }{\any} % the “sign” of matching for draining
7929 #4 b{} % list of drainers, empty by default
7931 >i T{\count \ubound}
7932 #5 D{\m@ne} % upper bound of iterations
7934 >i T{\default \Default}
7935 #6 b % default value (\NoValue by default)
7937 #7 T{\eachr \Eachr \defEachr} % default is \NoValue and that’s no harm
      since we test if #7 == \defEachr.
7939 #8 B{} % eachr, empty by default
7940 }{%
7942 \dc@process@matchsign {#1} {##1}%
7944 \dc@addtoParsed@{#2}%
7946 \gmu@if {strings}{\lostwax#1}%
7947 {\gmu@if {@dc@xadefault}}{%
7948   {\@xa\edefU\@xa\dc@LostWaxList\@xa{#2}}%
7949   {\edefU\dc@LostWaxList{#2}}%
7950 }%
7951 }%
7953 \@dc@process@matchsign {#3} {##3}%
7955 \dc@addtoParsed@{#4}%
7957 \@xa \@dc@addtospecs@bare \@xa{\the\numexpr #5}%
7958 \@dc@addtotoks@bare{#5}% to ignore possible expandafter.
7960 \dc@addtoParsed@ {#6}%
7962 \gmu@if {strings}{\defEachr#7}
7963 {%
7964   \Name\def{\dc@innername/defdEachr/\the\c@dc@argnum}%
7965   ##1{#8}%
7967   \Name \dc@addtoParsed@
7968   {\dc@innername/defdEachr/\the\c@dc@argnum}%
7969 }
7970 {\gmu@if {@dc@xaeacher@}}
7971 {\@dc@xadefaulttrue}{\@dc@xadefaultfalse}%
7972 \dc@addtoParsed@ {#8}%
7973 }%
7975 \gmu@if {strings}{\lostwax#1}
7976 {\grab@LostWax}
7977 {\@dc@ParseNextSpecifier}%
7978 }
\grab@LostWax 7981 \DeclareCommand\grab@LostWax {
7982 >iT{\lost \Lost}
7983 #1 b{\lostwax@NoVal}
7984 >iT {\count \ubound}
7985 #2 D{1}
7986 >iT{\endlost \EndLost}
7987 }{%
      We build an ignored “while” catcher.
7990 \def\dc@LostWaxA{>iw}%
7992 \gmu@ifdetokens {\lostwax@NoVal}{#1}
7993 {%

```

if no particular value of the “lost wax” is provided, we assume the same as the “until” delimiter(s).

```
7997     \prependtomacro\@dc@LostWaxA{>\@xa}%
7998     \addtomacro\@dc@LostWaxA{\@dc@LostWaxList}%
7999 }%
```

Otherwise first we check whether “lost wax” provided has nonempty intersection with the “until” delimiters.

```
8003  {%
8004     \@xa\gmu@ifstrintersect
8005     \@xa{\@dc@LostWaxList}{#1}
8006     {\addtomacro\@dc@LostWaxA{#1}}}%
8007     {\PackageError{gmutils/gmcommand}{^^J%
8008         You try to declare a "Lost Wax" catcher^^J%
8009         with the "Lost" list disjoint with the "Until" list.^^J%
8010         I assume sum of them}%
8011     }%
8013     \@xa \addtomacro\@xa\@dc@LostWaxA
8014     \@xa{\@xa{\@dc@LostWaxList #1}}%
8015 }%
8016 }%
8018 \addtomacro\@dc@LostWaxA{ \count #2 \relax}%
8020 \@xa\@dc@ParseNextSpecifier \@dc@LostWaxA
8021 }% of \grab@LostWax.

8024 \def\@dc@process@matchsign
8025 #1% the argument
8026 #2% its numeral in human language (for the error message)
8027 {%
8028     \gmu@CASE {strings} {{#1}\any }
8029     {\@dc@addtoParsed@bare {\any }}%
8031     \gmu@CASEsbany {#1} {Ww}
8032     {\@dc@addtotoks@bare {\any}}%
8034     \gmu@CASE {strings} {{#1}\none}
8035     {\@dc@addtoParsed@bare {\none }}%
8037     \gmu@CASEsbany {#1} {UuÜü\lostwax }
8038     {\@dc@addtotoks@bare {\none }}%
8040     \gmu@lastCASE
8041     {\PackageError{gmcommand}{%
8042         You *have* to declare »\string\any« or »\string\none« ^^J%
8043         as the #2 argument for the »\string\loop« catcher}}}%
8044 }%
8045 \gmu@ESAC
8046 }

8049 \long\pdef\UnDeclareCommand#1{%
8050     \let#1\@undefined
8051     \ifcsname\@dc@InnerName{#1}\endcsname
8052         \n@melet{\@dc@InnerName{#1}}{\@undefined}%
8053     \fi
8054     \ifcsname\@dc@SpecsArName{#1}\endcsname
8055         \n@melet{\@dc@SpecsArName{#1}}{\@undefined}%
8056     \fi
8057 }
```

## The \SameAs parsing

Implemented 2010/4/14–16. Lets you not to repeat all the complicated specifiers but just write `\SameAs\⟨another command⟩`

```
8066 \edef\gmu@tempa{%
8067   \def\@nx\gmu@ifHashless@##1%
8068   \detokenize{macro:}##2->##3\@nx\@nil}%
8069 \gmu@tempa
```

This produces `\gmu@ifHashless@#1macro:#2->#3\@nil` with the middle delimiter detokenized

```
8072 {\gmu@if {}{\gmu@ifempty{#2}{1}{0}\gmu@ifempty{#3}{2}{1}}}
```

This expands to 1 if #1 is a hashless macro (to be honest, if its meaning contains string `macro:->`)

```
8078 \long\edef\gmu@ifHashlessMacro
8079 #1% a CS, name or active char
      %% #2% (implicit) what if hashless
      %% #3% (implicit) what if hashy
8082 {%
8083   \edef\@nx\gmu@WHL@arg{\@nx\@xanxtri{#1}}%
      %% \unexpanded{\typeout{@@@>>\meaning\gmu@WHL@arg}}%
8085   \unexpanded{\@xa\@xa\@xa\gmu@ifHashless@
8086     \@xa\meaning\gmu@WHL@arg}%
8087   \relax % to ensure that #3 of \gmu@ifHashless@ is nonempty if match for the
      delimiters is found earlier
8089   \detokenize{macro:***->}\@nx\@nil % the sentinels
8090 }
```

```
8093 \long\def\gmu@ifDeclared#1{%
      % #1 a CS or csname or an active char,
      % #2 true branch,
      % #3 false branch
```

```
8099 \let\gmu@ifDe@next\@secondoftwo
```

The test is three-level: first we check whether the specifiers' carrier is defined, then whether the inner macro is defined, then we check if the outer CS is a hashless macro (it could have been that the outer CS was redefined with sth. else)

```
8106 \gmu@CASEnot {csname} {\@dc@SpecsArName{#1}\endcsname}%
8107 {}%
8109 \gmu@CASEnot {csname}{\@dc@InnerName{#1}\endcsname}%
8110 {}%
8112 \gmu@EatCases
8113 {\gmu@ifHashlessMacro{#1}%
8114   {\edef\gmu@ifDe@resa{%
8115     \@nx\gmu@ifxany\@xanxcs{\@dc@InnerName{#1}}%
8116     {\unexpanded\@xa\@xa\@xa{\gmu@WHL@arg}}%
```

Above: to get the contents of the outer macro we use the macro carrying conversion of possible name into a CS and expand it twice. Then we freeze it with `\unexpanded`.

```

8120         {\let\@nx\gmu@ifDe@next\@nx\@firstoftwo}% if the inner macro is present
            in the contents of #1, we take the first implicit argument
8123         {}% otherwise we do nothing (taking the second implicit is already set)
8125     }% of edef
8126     \gmu@ifDe@resa
8127 }% of if a hashless macro
8128 {}% of if not a hashless macro
8129 }%
8131 \gmu@ESAC
8132 \gmu@ifDe@next
8133 }% of \gmu@ifDeclared
8136 \long\def\@dc@SameAs#1{%

```

As you see, it's very simple when we took care of storing the specifiers in a special macro: we only take the middle brace of it.

```

8140     \gmu@ifDeclared {#1}%
8141     {%

```

7 `\expandafters` before `\@dc@ParseNextSpecifier` and 3 before `\@secondofthree` to expand the specifiers carrier twice, then get its middle piece of data, which is the specifiers' sequence, and then parse as a part of "our" specifiers.

```

8147     \@xa\@xa\@xa \@xa\@xa\@xa \@xa
8148     \@dc@ParseNextSpecifier
8149     \@xa\@xa\@xa \@secondofthree
8150     \csname \@dc@SpecsArName{#1}\endcsname
8151 }%
8152 {%
8153     \PackageError{gmcommand}{%
8154         Command \@nx#1 ^^J%
8155         is not declared with \DeclareCommand. ^^J%
8156         I can't repeat its parameters in current command declaration}%
8157     }%
8158 }%
8159 }% of \@dc@SameAs

```

```

\DeclareCommand 8163 \DeclareCommand\DCUse{
\DCUse          8164 #1 >Pm % the command
8165 }{%

```

TODO: handling longness of the arguments properly. (Now it assumes all the inner arguments are `\long`).

```

8170     \gmu@ifDeclared{#1}%

```

If `#1` is a Declared command, we put its inner macro to the `\@dc@arguments` toks and launch catcher of proper number of undelimited arguments.

```

8175     {%
8176         \@dc@arguments\@xa{\@xa
8177             \csname \@dc@InnerName{#1}\endcsname
8178         }%
8180         \csname ArgumentCatcher@Pm@\romannumeral
8181             \@xa \@xa \@xa \@thirdofthree
8182             \csname \@dc@SpecsArName{#1}\endcsname % of arity carrier
8183         \endcsname % of the catcher

```

```

8184 }%
8185 {\DC@EndNotDeclaredErr{#1}}%
8186 }%

8189 \pdef \DC@EndNotDeclaredErr #1{%
8190 \PackageError{gmcommand}{%
8191   Command \backslash end\strip@backslash{#1} ^^J%
8192   is not declared with \DeclareCommand. ^^J%
8193   Therefore I can't use its DC-inner macro (because of its
      nonexistence).^^J}%
8194 }%
8195 }

\DeclareCommand 8198 \DeclareCommand\DCUseEnd{
\DCUseEnd 8199 #1 >Pm % the command (without “end”)
8200 }{%
8202 \gmu@ifDeclared{end\strip@backslash{#1}}%
8204 {%
8205   \@dc@arguments\@xa{\@xa
8206     \csname \@dc@InnerEndName{#1}\endcsname
8207   }%
8209   \csname ArgumentCatcher@Pm@\romannumeral
8210     \@xa \@xa \@xa \@thirdofthree
8211     \csname \@dc@SpecsArName{#1}\endcsname % of arity carrier
8212     \endcsname % of the catcher
8213   }%
8214   {\DC@EndNotDeclaredErr{#1}}%
8215 }%

      end of \SameAs parsing

```

### Immediate uses

```

\DeclareEnvironment 8223 \DeclareCommand\DeclareEnvironment \long
8224 {
8225   #1--#4 \SameAs \DeclareCommand

      We repeat the specifiers, but the role of #4 is slightly different here: here it's the \begin
      part definition.

8229   #5 m % the end definition; it can contain any #(n)—the arguments of the environ-
      ment are passed to it, too.

8232 }{%
8234 \def\gmu@DeclareEnvironment@resa{\envhack}% we add the information we de-
      fine an environment.
8236 \gmuIfValueT{#2}%
8237 {\addtomacro\gmu@DeclareEnvironment@resa{#2}}% we pass all the ‘prefixes’ to
      % \DeclareCommand

8240 \@xa\DeclareCommand\csname#1\@xa\endcsname
8241 \gmu@DeclareEnvironment@resa{#3}{#4}%

      Now the begin definition is done. Let's get down to the end definition.
8246 \let\@dc@env@endglobal=\@dc@global@ % note this CS was for sure redefined by
      the last \DeclareCommand (and by nothing else) and that's exactly what we
      want.

```

(2010/07/15, v0.993:) we make the `\end...` command `\DeclareCommand` ed for the symmetry, for simplifying definition of `\envirlet` in particular

```
8253 \edef\gmu@DeclareEnvironment@resb{%
8254   \@nx\@xa
8255   \@xanxcs{\bslash end#1}%
8256   \@nx\the
8257   \@xanxcs{\dc@EName{#1}}%
8258 }% of \edef. It results in
```

`\@xa \end<name> \the\<name>'s_args`

which does not collide with the inner macro of the `\end...` command, since the latter is `\end...\`.

```
8263 \gmu@if{csname}{\dc@EName{#1}\endcsname}{\gmu@namelet\global{%
      \dc@EName{#1}}{\@undefined}}}%
```

We postpone the definition of the `\end...` command after of the end inner macro not to spoil the `\dc@innerhashes`.

Now the end inner macro

```
8268 \edef\gmu@DeclareEnvironment@resa{%
8269   \@dc@global@\long\def % the inner end macro is always \long and perhaps
      defined \global ly.
8271   \@xanxcs{\bslash end#1}%
8272   \@xau@\dc@innerhashes % it's #1#2#3...—the inner end macro takes the same
      number of parameters as the begin macro.
8274   {% the definition body
8275     \unexpanded{#5}}%
8276 }% of \edef...@resa.
8277 \gmu@DeclareEnvironment@resa
```

And now the postponed from above definition of `\end...`

```
8281 \edef\gmu@DeclareEnvironment@resc{%
\@xanxcs 8282 \DeclareCommand\@xanxcs{end#1}%
8283 \dc@env@endglobal
8284 }% no parameters
8285 {\@xau \gmu@DeclareEnvironment@resb
8286 \ignoretrue}%
8287 }%
8288 }%
8289 \gmu@DeclareEnvironment@resc
8292 }% of \DeclareEnvironment
```

```
\NewCommand 8295 \DeclareCommand\NewCommand
8296 {\SameAs\DeclareCommand}
8297 {%
```

```
\gmu@ifCSdefined 8298 \gmu@ifCSdefined {#1}
8299 {\DCUse\DeclareCommand#1{#2}{#3}{#4}}%
```

```
\PackageError 8300 {\PackageError{gmcommand}%
8301 {Command \@nx#1 already defined \on@line!}%
8302 }%
8303 }%
8304 }
```

```
\RenewCommand 8307 \DeclareCommand\RenewCommand
```

```

8308 {\SameAs\DeclareCommand}
8309 {%
\gmu@ifCSdefined 8310 \gmu@ifCSdefined {#1}
8311 {\DCUse\DeclareCommand#1{#2}{#3}{#4}}%
\PackageError 8312 {\PackageError{gmcommand}%
8313 {Command \@nx#1 not yet defined \on@line!}%
8314 {}}%
8315 }%
8316 }

\ProvideCommand 8318 \DeclareCommand\ProvideCommand
8319 {\SameAs\DeclareCommand}
8320 {%
\gmu@ifCSdefined 8321 \gmu@ifCSdefined {#1}
8322 {}
8323 {\DCUse\DeclareCommand#1{#2}{#3}{#4}}%
8324 }

\DeclareCommand 8327 \DeclareCommand\NewEnvironment
\NewEnvironment 8328 {\SameAs\DeclareEnvironment}
8329 {%
8330 \gmu@ifdefined{#1}
8331 {\DCUse\DeclareEnvironment{#1}{#2}{#3}{#4}{#5}}%
8332 {\PackageError{gmcommand}
8333 {Environment #1 already defined \on@line!}}%
8334 }%
8335 }

\RenewEnvironment 8337 \DeclareCommand\RenewEnvironment
8338 {\SameAs\DeclareEnvironment}
8339 {%
8340 \gmu@ifdefined {#1}
8341 {\DCUse\DeclareEnvironment{#1}{#2}{#3}{#4}{#5}}%
8342 {\PackageError{gmcommand}
8343 {Environment #1 not yet defined \on@line!}}%
8344 }%
8345 }

```

## Setting for Xe<sub>3</sub>TeX

which could not be defined earlier (in `gmbase` because of lack of `\DeclareCommand`).

```

\XeTeXthree 8352 \DeclareCommand\XeTeXthree{o}{%
8356 \@ifXeTeX{%
8357 \gmuIfValueT{#1}{\PassOptionsToPackage{#1}{fontspec}}%
8358 \@ifpackageloaded{gmverb}%
8359 {%
8360 \Store@Macro\verb
8361 \StoreEnvironment{verbatim}%
8362 \StoreEnvironment{verbatim*}%
8363 }{}%
8364 \RequirePackage{xltextra}% since v 0.4 (2008/07/29) this package redefines
\verb and verbatim*, and quite elegantly provides an option to suppress
the redefinitions, but unfortunately that option excludes also a nice defini-
tion of \xxt@visibleSPACE which I fancy.

```



```

8371 \@ifpackageloaded{gmverb}%
8372 {%
8373 \Restore@Macro\verb
8374 \RestoreEnvironment{verbatim}%
8375 \RestoreEnvironment{verbatim*}%
8376 }{}%
8378 \AtBeginDocument{%
8379 \@ifpackageloaded{gmlogos}{%
8380 \Restore@Macro\LaTeX\Restore@MacroSt{LaTeX }% my version of the
      LATEX logo has been stored just after defining, in line 10065.
8383 \Restore@Macro\eTeX}%
8384 }{}%
8385 }%

8387 \pdef\adddefaultfontfeatures##1{%
8388 \addtomacro\zf@default@options{#1,}}
8389 }% of \ifXeTeX's first argument,
8390 {}% \ifXeTeX's second argument,
8391 }% of \XeTeXthree's body.

\setspacekip 8394 \DeclareCommand\setspacekip{%
8395 A{1}% optional factor for all three components
8396 O{\fontdimen2\font}%
8397 >is
8398 O{\fontdimen3\font}%
8399 >is
8400 O{\fontdimen4\font}}
8401 {\spaceskip=\glueexpr\dimexpr#2*#1\relax plus\dimexpr #3*#1\relax
8402 minus\dimexpr#4*#1\relax\relax}

8404 \pdef\unsetspaceskip{%
8405 \spaceskip=\z@skip
8406 }

8409 \def\makestarlow{%
8410 \begingroup\lccode`~=\*\lowercase{%
* 8411 \endgroup\def~{\gmu@lowstar}}% 2009/10/19 \let changed to \def to allow
      redefinitions of \gmu@lowstar.
8413 \catcode`\*=\active
8414 \defLowStarFake
8415 }

\defLowStarFake 8417 \DeclareCommand\defLowStarFake{%
8418 Q{+-0123456789, .}{0,5}% fraction of fontchar depth of the star glyph
8419 }%
8420 {%
8421 \def\gmu@lowstarfake{%
8422 \leavevmode\vbox{\hbox{*}\kern#1\fontchardp\font`*}%
8423 }%
8424 }

8427 \def\gmu@lowstarfake{*} % useful for next command where normal star is low.

The macro given below is taken from the multicol package (where its name is
\enough@room). I put it in this package since I needed it in two totally different works.

\enoughpage 8434 \DeclareCommand\enoughpage{%

```

```

8435 #1 D{\NoValue} % (optional short version (number of \baselineskips)) Before
      2010/9/10 the default value was the D-default, i.e., 0, which lead to improper
      working of this command.
8439 #2 B{2\baselineskip}% (2) optional (formerly mandatory) long version of re-
      quired room on a page
8441 >is
8442 #3 >PB{} % (3) what if the room is enough
8443 >is
8444 #4 >PB{\newpage} % (4) what if there's to little room on a page
8445 }{%
8451 \gmu@if {num}
8452 {\numexpr
8453 \ifdim
      if the space left is less than we wish then 1, otherwise 0.
8455 \gmu@pageremain
8456 <%
8457 \dimexpr
8458 (\gmuIfValueTF{#1}{#1\baselineskip}{#2})*1%
8459 \endexpr
      (2010/06/28, v0.993:) I added |(⟨...)*1| to assure that || really delimits the dimexpr
8464 1\else 0%
8465 \fi
8466 *%
8467 \ifdim \gmu@pageremain >-1sp
8468 1% if the room left on current page is nonnegative, then we indeed are on
      current page so “enough” and “not enough” does make sense: Enough
      is enough.
8471 \else 0% otherwise we are on next page (or further) and assume there is enough
      room for our purpose.
8473 \fi
8474 *%
8475 \ifdim \pagegoal <\maxdimen
8476 1 % if there are some boxes on current page, so checking room does make
      sense.
8478 \else 0% otherwise (no boxes on current page) assume there's enough room
8480 \fi
8481 >\z@
8482 }% of condition
8483 {\typeout{@@@@ not enough page \on@line}%
8484 #4%
8485 }%
8486 {#3}%
8487 }
8491 \long\def\gmu@LC@LetInners
      Macro letting the inner (executing) and the specs. carrying macros of a Declared
      command.
8496 #1% scope prefix (\global or \relax
8497 #2% left side
8498 #3% right side of the assignment

```

```

8499 {%
8500 \edef\gmu@LC@InnerLeft{\@dc@InnerName{#2}}%
8501 \edef\gmu@LC@InnerRight{\@dc@InnerName{#3}}%
8503 \gmu@namelet{#1}{\gmu@LC@InnerLeft}{\gmu@LC@InnerRight}%

\tri@let put here originally was disastrous since it gobbled the leading bslash

8507 \edef\gmu@LC@InnerLeft{\@dc@SpecsArName{#2}}%
8508 \edef\gmu@LC@InnerRight{\@dc@SpecsArName{#3}}%
8510 \gmu@namelet{#1}{\gmu@LC@InnerLeft}{\gmu@LC@InnerRight}%
8511 }

```

```

8513 \long\def\gmu@CL@PrepareArg
8514 #1% left/right
8515 #2% \@xa or \edef, maybe sth. more in the future
8516 #3% a CS, text of a name or an active char
8517 {%
8518 \Name\let{gmu@CL@#1}\@undefined
8519 \ifx\@xa#2\Name\edef{gmu@CL@#1}{\@xa#3}\fi
8520 \ifx\edef#2\Name\edef{gmu@CL@#1}{#3}\fi
8521 \unless\ifcsname gmu@CL@#1\endcsname
8522 \Name\edef{gmu@CL@#1}{%
8523 \gmu@if{cat}{\@nx~\@nx#3}%
8524 {\@nx#3}{\strip@bslash{#3}}%
8525 }%
8526 \fi
8527 }

```

```

\CommandLet 8530 \DeclareCommand\CommandLet \long {%
8531 \Scope % (1)
8532 T{\@xa\edef} % (2) whether left side should be \expandaftered
8533 m % (3) left side of assignment
8534 >iT{=} % pro forma
8535 T{\@xa\edef} % (4) whether right side should be \expandaftered
8536 m % (5) right side of the assignment

```

Both arguments may be names. Warning! The first token of a csname will not be expanded (will be `\string ed`).

```

8541 }{%
8543 \gmu@CL@PrepareArg{left}#2{#3}%
8544 \gmu@CL@PrepareArg{right}#4{#5}%
8546 \@xa\gmu@ifDeclared\@xa{\gmu@CL@right}{%
8547 \edef\gmu@LC@resa{%
8548 \DeclareCommand %
8549 \@xa\@xanxtri\@xa{\gmu@CL@left}%
8550 #1% scope prefix
8551 {\@nx\SameAs\@xa\@xanxtri\@xa{\gmu@CL@right}}% args. spec
8552 }% the body of the command will be \let via letting inners.
8553 }%
8555 \gmu@LC@resa

```

Now we have a command #2 Declared with empty body and proper args' parsers and inner macro properly named.

```

8558 \@XA{\@xa\gmu@LC@LetInners\@xa#1%
8559 \@xa{\gmu@CL@left}}\@xa{\gmu@CL@right}%

```

No Ampulex for the outer macro is necessary.

```

8561 }% of if Declared
8562 {% if not Declared, we just \let:
8563 \if\@nx#3\typeout{@@@ »\@x\@dc@InnerName\@x{\gmu@CL@right}« is
      not
8564 Declared?: \Name\meaning{\@x\@dc@InnerName\@x{\gmu@CL@right}}%
8565 }%
8566 \show\NotDeclared
8567 \fi
8568 \@XA{\@x\@tri@let\@x#1%
8569 \@x{\gmu@CL@left}}\@x{\gmu@CL@right}%
8570 }%
8571 }
\envirlet 8574 \DeclareCommand\envirlet \long{\Scope mm}{% for \letting environments.
8575 \CommandLet #1{#2}{#3}%
8576 \CommandLet #1{end#2}{end#3}%
8578 }

```

A numeric `for` loop as in decent languages:

```

\@fornum 8582 \DeclareCommand\@fornum{%
8583 m% (1) initial (least) num. value
8584 m% (2) limit (last iterated over) num. value
8585 O{1}% (3) step of iteration
8586 m% (4) body of the loop
8587 }%
8588 {\edef\gmu@fornum@min{\the\numexpr #1}%
8589 \edef\gmu@fornum@lim{\the\numexpr #2}%
8590 \let\gmu@fornum@curr\gmu@fornum@min
8591 \@whilenum\gmu@fornum@curr<\gmu@fornum@lim
8592 \do{#4%
8593 \edef\gmu@fornum@curr{%
8594 \the\numexpr\gmu@fornum@curr+#3}%
8595 }% of \@whilenum's body.
8596 }% of \@fornum.
8599 \pdef\StoreCommand{%
8600 \begingroup
8602 \MakePrivateLetters
8603 \@StoreCommand
8604 }
8607 \long\def\gmu@SC@StorageName
8608 #1% a CS, name or active char
8609 #2% group level
8610 {%
8611 \gmu@storeprefix/%
8612 \ifnum\numexpr#2>-1 \the\numexpr(#2)*1\relax\fi
8613 \bslash@or@ac {#1}%
8614 }%

```

The storing csnames with  $\#2 \leq -1$  don't contain the number.

```

8617 \def\gmu@SC@setgrouplevel
8618 #1% +|- \NoValue

```

```

8619 #2% a decimal number
8620 {\edef\gmu@SC@grouplevel{%
8621     \the\numexpr
8622     \gmuIfValueTF{#1}%
8623     {\currentgrouplevel#1#2+1}{#2}%
8624 }% of the grouplevel-carrying macro
8625 }

```

```

\@StoreCommand 8628 \DeclareCommand\@StoreCommand
8629 {%
8630 #1 T{+-}% grouplevel shift indicator
8631 #2 d % group level or shift (the latter relative to the current g.l.)
8632 #3 >Pm% name or CS
8633 }{%
8634 \endgroup % we close the group opened in line 8600.

```

Now. We define a special csname prefixed with `\gmu@storeprefix` and containing current group level. Later, when we refer to the stored command, we'll check subsequent levels, from current upwards.

```

8639 \gmu@SC@setgrouplevel{#1}{#2}%
8641 \edef\gmu@SC@resa{%
8642     \gmu@SC@StorageName{#3}{\gmu@SC@grouplevel}}%
8644 \let\gmu@SC@scope\relax
8645 \ifnum\gmu@SC@grouplevel>\currentgrouplevel
8646     \let\gmu@SC@scope\global
8647 \fi
8648 \@xa\CommandLet\@xa\gmu@SC@scope\@xa{\gmu@SC@resa}{#3}%
8650 }

```

```

8653 \pdef\StoreEnvironment{%
8654     \begingroup
8655     \MakePrivateLetters
8656     \@StoreEnvironment
8657 }

```

```

8660 \def\@StoreEnvironment#1{%
8661     \@StoreCommand{#1}%
8662     \begingroup
8663     \@StoreCommand{end#1}%
8664 }

```

```

8667 \pdef\RestoreEnvironment{%
8668     \begingroup
8669     \MakePrivateLetters
8670     \@RestoreEnvironment
8671 }

```

```

8674 \def\@RestoreEnvironment#1{%
8675     \endgroup
8676     \RestoreCommand{#1}%
8677     \RestoreCommand{end#1}%
8678 }

```

```

8681 \pdef\UseStored{%
8682     \begingroup
8683     \MakePrivateLetters

```

```

8684 \@UseStored}
\@UseStored 8688 \DeclareCommand\@UseStored \long{%
8689 #1 T{+-}% (1) indicator of grouplevel shift (in distiction from an absolute group
level)
8691 #2 d % (2) optional grouplevel or shift (0 by default)
8692 #3 O{\@gobble}% (3) stuff to be put before the #4 CS, e.g., an assignment. The
default value \@gobble makes the default use of a stored CS to be just an
execution/expansion of its meaning. Another use of this command, perhaps
the most useful, is restoring of the previous meaning, as we'll see later.
Note that #4 after #3 will be put in braces.
8699 #4 m % (4) a CS or csname to be taken from the storage liquid
8700 }{%
8702 \endgroup
8706 \gmu@SC@setgrouplevel{#1}{#2}%
8707 \edef\gmu@SC@grouplevel{%
8708 \the\numexpr\gmu@SC@grouplevel+1}% for the first turn of loop
8710 \loop
8711 \edef\gmu@SC@grouplevel{%
we decrease the group level number
8713 \the\numexpr\gmu@SC@grouplevel-1}%
8715 \edef\gmu@SC@currname{%
define the cs' storage name for that level
8717 \gmu@SC@StorageName{#4}{\gmu@SC@grouplevel}}%
8718 \ifnum
and check the conjunction of conditions: the group level is >=-1...
8720 \unless\ifnum\gmu@SC@grouplevel<-1 1\else 0\fi
and the csname remains not defined.
8722 \unless\ifcsname \gmu@SC@currname \endcsname 1\else 0\fi
8723 =11
8724 \repeat
We repeat until we find defined csname or reach the top level.
8727 \gmu@if{csname}{\gmu@SC@currname\endcsname}%
8728 {%
8730 \@XA{#3{#4}}\csname \gmu@SC@currname\endcsname}% first
8731 {%
8733 #3{#4}\@undefined}% second
8734 }% of \@UseStored
\RestoreCommand 8737 \DeclareCommand\RestoreCommand{\Scope d}{%
8738 \UseStored #2[\CommandLet #1*}%
8739 }
\gmu@smashbox 8759 \newbox\gmu@smashbox
\set@smashbox 8761 \DeclareCommand\set@smashbox{%
8762 \Scope
8763 T{hv}{h}
8764 }{#1\setbox\gmu@smashbox =\csname #2box\endcsname }%

```

```
8767 \def\smash@box{\smash{\box\gmu@smashbox}}
```

Now, using the iterating catchers, we define a finder of max/min dimen of a given sequence of arguments.

(There's also the `\gmu@comparenum` macro defined in `gmbase` that expands to the smaller or larger numbers (numexprs) of given two.)

```
\gmu@boxA 8778 \newbox\gmu@boxA
\gmu@boxB 8779 \newbox\gmu@boxB
\gmu@dima 8780 \newdimen\gmu@dima
\gmu@dimb 8781 \newdimen\gmu@dimb
```

We call this macro `\gmu@extremebox` because in the `gmbase` package we defined `\gmu@maxdim` and `\gmu@mindim` as expandable taking a sequence of `dim(expr)s` and expanding to the biggest/smallest of them.

```
\gmu@extremebox 8788 \DeclareCommand\gmu@extremebox {
8789   #1 m % a dimen register (will be assigned the extr. value) or a CS(will be defined
      as \the extreme value)
8791   #2 T {\min\max}{\max} % kind of extreme
8792   #3 T {\wd\ht\dp\totalheight}{\wd} % dimension to compare, the last two for
      total \ht+\dp.
8793   #4 \lostwax {\global\relax} \each{er{\gmu@dimextreme@step} \lost{%
      \relax}
8794   #5 T{\global}{ } % scope of assignment
8795 }{%
8796   \gmu@if x {\max#2}
8797   {\gmu@dimb=-\maxdimen
8798     \def\gmu@dimextreme@comp{<}%
8799   }
8800   {\gmu@dimb=\maxdimen
8801     \def\gmu@dimextreme@comp{>}%
8802   }%
8804   \long \def\gmu@dimextreme@step ##1{%
8805     \setbox\gmu@boxA=\hbox{##1}%
8807     \gmu@dima= \gmu@if x {\totalheight#3}
8808     {\dimexpr \ht\gmu@boxA +\dp\gmu@boxA\relax}
8809     {#3\gmu@boxA}%
8810     \relax
8812     \gmu@if {dim}
8813     {\gmu@dimb\gmu@dimextreme@comp \gmu@dima }
8814     {\gmu@if {dim} {\gmu@dima>\z@}
8815       {\gmu@dimb=\gmu@dima }
8816     }%
8817   }
8818   }%
8820 }%
8822 #4%
8823 \IfIs #1\dimen
8824 {#5#1=\gmu@dimb}
8825 {#5\edef #1{\the\gmu@dimb }}%
8826 }%
```

## Getting height ;-)

The `\gmu@getht` macro is intended for very high vboxes, i.e., whose height may exceed `\maxdimen`. (In such case TeX's `dimens` cycle which is not what we want in `<` and `>` comparisons.) So, this macro checks whether a vbox got “too high” and returns `\maxdimen` as substitute of its height. If given box fits within `\maxdimen` or is not a vbox, then its original height is returned.

```
8836 \pdef\gmu@getht
8837 #1% box register
8838 #2% dimen (or glue) register to assign the (substitute) height to
8839 {%
8840   \setbox\gmu@boxA=\copy#1\relax
8841   \ifvbox\gmu@boxA
8842     \vbadness@M
8843     \setbox\gmu@boxB=\vsplit\gmu@boxA to \maxdimen
8844     \vbadness@Restore
```

chcemy rzeczywiście pomierzyć, a nie być może odrzucić odstęp, kerny i grzywny.

```
8848   \setbox\gmu@boxB=\vbox{\splitdiscards }%
8849   \ifdim \ht\gmu@boxB >\z@
8850     \setbox \gmu@boxA=\vbox{
8851       \unvbox\gmu@boxB
8852       \unvbox\gmu@boxA
8853     }%
```

%% \else % otherwise we leave box A intact for the Test of Śunyata.

```
8855   \fi
8856 \else
8857   \setbox\gmu@boxA=\box\voidb@x
8858 \fi
8860 \ifvoid\gmu@boxA
8861   #2=\ht#1\relax
8862 \else
8863   #2=\maxdimen
8864 \fi
8865 }% of \gmu@getht
```

TODO: if one wraps such a very high vbox in an hbox, then attempt of assignment of its height results with an error “Dimension too large”. Can we handle it? *Should* we handle it?

## Enlarging and scaling of the font size

```
\enlargesize 8874 \DeclareCommand\enlargesize{
8875   #1 m % enlargement (dim(expr)) for font size
8876   #2 b % enlargement (dim(expr)) for baselineskip (if absent, #1 is used)
8877 }{\edef\gmu@tempa{%
8878   \@nx\fontsize{\the\dimexpr\f@size pt+#1}%
8879   {\the\dimexpr 1\baselineskip+ \gmuIfValueTF{#2}{#2}{#1}}%
8880 }\gmu@tempa\selectfont
8881 }

\scalesize 8883 \DeclareCommand\scalesize {
8884   #1 m % scale for font size
```



```

8885 #2 b % scale for baselineskip (if absent, #1 is used)
8886 \gobblespace
8887 }{\edef\gmu@tempa{%
8888   \@nx\fontsize{\the\dimexpr #1\dimexpr\f@size pt\relax}%
8889   {\the\dimexpr \gmuIfValueTF{#2}{#2}{#1}\baselineskip}%
8890   }\gmu@tempa\selectfont
8891 }

8894 \let\gmu@discretionaryhyphen\-\ % for the cases when we don't redefine \ but use
      % \

```

A redefinition of \- that makes it optional-argument to allow further hyphenation

```

\bihyphen 8899 \DeclareCommand\bihyphen{
8900   O{*} % token that'll make the discretionary hyphen \- allow other break-points
8901 }{%
\gmu@discretionaryhyphen 8902 \DeclareCommand\gmu@discretionaryhyphen{%
8903   T{#1}%
8904   G{&&}%
8905   a
8906 }{%
8907   \gmuIfValueT{##2}{%
8908     \gmu@ifempty{##2}{}{%
8909       \def\gmu@bihyphen@char{##2}}%
8910   }%
8911   \gmuIfValueT{##3}{%
8912     \gmu@ifempty{##3}{}{%
8913       \def\gmu@bihyphen@corr{##3}}%
8914   }%

```

Depending on #1 we allow (if present, then we take \discre) hyphenation of the word's before- and after-parts or forbid it (if absent, then we take \discretionary).

```

8920   \gmuIfValueTF{##1}\discre\discretionary
8921   {% before break
8922     \gmuIfValueTF{##2}
8923     {%
8924       \gmu@ifempty{##2}{\gmu@bihyphen@char}{##2}%
8925     }%
8926     {%
8927       \ifnum\hyphenchar\font>\z@
8928       \char\hyphenchar\font
8929       \fi}%
8930   }% end of before break
8931   {% after break
8932     \gmuIfValueT{##3}{%
8933       \gmu@ifempty{##3}{\gmu@bihyphen@corr}{##3}%
8934     }%
8935   }%
8936   {% without break
8937   }% almost as in The TEX book: unlike The TEX book, we allow hyphenchars
      ≥ 255 as we are XqTEX.
8939 }% of \DeclareCommand\-\
8940 \gmu@storeifnotyet\-\ % original \- is a TEX's primitive, therefore we should store
      it.

```

```

8942 \CommandLet\-\gmu@discretionaryhyphen
8943 \CommandLet\@dischyph\gmu@discretionaryhyphen % to override framed.sty
8945 \pdef\gmu@flexhyphen{\gmu@discretionaryhyphen#1\relax}%
8946 }% of \bihyphen

8949 \relaxen\gmu@bihyphen@corr

8952 \@ifXeTeX{%
\gmshowlists 8953 \DeclareCommand\gmshowlists {
8954     >iT{\depth}
8955     D{1}%
8956     >iT{\breadth}
8957     D{10000000}
8958     }%
8959 }{% not in XYTEX we use legacy D arg. specifiers:
\gmshowlists 8960 \DeclareCommand\gmshowlists {
8961     >iT{\depth}
8962     D{1}%
8963     >iT{\breadth}
8964     D{10000000}
8965     }%
8966 }
8967 {\tracingonline=\@ne
8968 \showboxdepth=#1\relax % how many levels of box nesting should be shown
8970 \showboxbreadth=#2\relax % after how many elements »etc.« will be put
8971 \showlists
8972 }

\gmtracingoutput 8975 \DeclareCommand\gmtracingoutput {
8976 \SameAs\gmshowlists
8977 }
8978 {\tracingonline=\@ne
8979 \tracingoutput=\@ne
8980 \showboxdepth=#1\relax % how many levels of box nesting should be shown
8982 \showboxbreadth=#2\relax % after how many elements »etc.« will be put
8983 }

\c@FiBreakPenalty 8995 \newcount\c@FiBreakPenalty

\AllowFiBreak 8997 \DeclareCommand \AllowFiBreak {%
    The defaults are as in DEK's \filbreak.

8999     Q{1}{1}
9000     = {\c@FiBreakPenalty }
9001 }
9002 {%
9003 \penalty\@M
9004 \csname vfi#1\endcsname
9005 \penalty \c@FiBreakPenalty
9006 \csname vfi#1neg\endcsname
9007 }

\IgnorePars 9025 \DeclareCommand \IgnorePars {m Q{\par}} {#1}

    This command gobbles a sequence of epty lines and explicit \par CS es. Unlike primitive \ignorespaces, it doesn't expand macros.

```

```

\putpenalty 9029 \DeclareCommand\putpenalty {=}
9030 {%
\gmu@iflastglue 9031 \gmu@iflastglue
9032 {\@tempskipa=\lastskip
9033 \mode@skip -\@tempskipa
9034 \penalty \@tempcnta
9035 \mode@skip \@tempskipa
9036 }%
\gmu@iflastkern 9037 {\gmu@iflastkern
9038 {\@temddima=\lastkern
9039 \kern-\@temddima
9040 \penalty \@tempcnta
9041 \kern \@temddima
9042 }
9043 {%
9044 \penalty\@tempcnta
9045 }%
9046 }%
9047 }

\gmu@TrashSkip 9049 \newskip \gmu@TrashSkip
9051 \pdef \mode@skip {%
9052 \ifvmode \vskip
9053 \else
9054 \ifhmode \hskip
9055 \else
9056 \ifmmode \mskip
9057 \else
9058 \gmu@TrashSkip
9059 \fi
9060 \fi
9061 \fi
9062 }
9086 </command>

```

## Ampulex Compressa-like modifications of macros—the gmampulex package<sup>5</sup>

```

9091 <utils> \gmu@PackOptionX{ampulex}
9092 <*ampulex>

```

Ampulex Compressa is a wasp that performs brain surgery on its victim cockroach to lead it to its lair and keep alive for its larva. Well, all we do here with the internal L<sup>A</sup>T<sub>E</sub>X macros resembles Ampulex's actions but here is a tool for a replacement of part of macro's definition.

```

9101 \RequirePackage{gmcommand}
\ampulexlet 9104 \DeclareCommand\ampulexlet\long

```

<sup>5</sup> This file has version number v0.996 dated 2011/10/12.

```

9108 {Q{\outer\long\global\protected}} % (1) (optional) prefix(es); allowed is any
      sequence of them in any order, just like for the original TEX's \def.
9111 T{\def\edef\gdef\xdef\pdef}{\def} % (2) (optional) kind of definition; if not
      specified, \def will be used.
9113 m % (3) macro to be let to,
9114 m % (4) macro to provide the definition,
9115 O{} % (5) \def's parameters string; empty by default,
9116 O{} % (6) definition body's parameters to be taken in a one-step expansion of the
      redefined macro; empty by default; the undelimited parameters should be
      double-braced here.
9119 m % (7) start token(s),
9120 m % (8) end token(s)
9121 m % (9) the replacement of #7, #8 and whatever between them.
9122 }{% For the example of usage see 10288.

9130 \long\def\gmu@ampulexlet@resa
9131 ##1#7% we put #7 as a delimiter
9132 ##2#8% we put #8 as a delimiter
9133 ##3\gmu@AmpulexDelimiter{%

```

We use a special (undefined) CS\gmu@AmpulexDelimiter as the final delimiter because standard L<sup>A</sup>T<sub>E</sub>X's \@nil isn't probably a good idea since we want to ampulex deep L<sup>A</sup>T<sub>E</sub>X's macros and other \gmu@... macros too.

```

9139 \gmu@ifempty{##3}%
9140 }%

```

Now \gmu@ampulexlet@resa is redefined to produce an open \gmu@ifempty depending on whether the start and end token(s) are found in the meaning of #4.

Before we proceed, we deal with a difficulty with a special case when #6 is “#1”, which occurs because of stripping braces of a single-brace argument.

```

9148 \gmu@ifutokens{#6}{##1}%
9149 {\def\ampulex@Args{%
9150   ###1}}%
9151 }%
9152 {\edef\ampulex@Args{\@nx\unexpanded{%
9153   \unexpanded{#6}}}%
9154 \long \def\gmu@ampulexlet@resc##1{%
9155   \@xa\@xa\@xa\@xa \@xa\@xa\@xa\gmu@ampulexlet@resa
9156   \@xa\@xa\@xa#4%
9157   \ampulex@Args
9158   ##1% this parameter will be substituted with #7#8 in line 9163 and with emptiness
      in line 9214.
9160   \gmu@AmpulexDelimiter
9161 }%
9163 \gmu@ampulexlet@resc{#7#8}%

```

%% \gmu@ampulexlet@resb We've just applied the checker and it produces an open \gmu@ifempty{*some tokens*} if the delimiters are found in the meaning of #4 so, if *some tokens* are none, we issue a warning

```

9168 {%
9169 \PackageWarning{gmampulex}{%
9170 \@nx#4 doesn't contain tokens
9171 \detokenize{#7} nor \detokenize{#8}. You better check if this is

```

```

9172     what you want to redefine.^^J%
9173     \@nx#4 is^^J%
9174     \meaning#4^^J%
9175     }}%

```

and we proceed if they are really some

```

9177     {%

```

We define a temporary macro with the parameters delimited with the ‘start’ and ‘end’ parameters of `\ampulexdef`. It has to stand a double `\edef`.

```

9181     \edef\gmu@ampulexlet@resa{%
9182         \long\def\@nx\gmu@ampulexlet@resa
9183         ###1\unexpanded{#7}%
9184         ###2\unexpanded{#8}%
9185         ###3\@nx\gmu@AmpulexDelimiter{%
9186         \@nx\unexpanded{###1}%

```

we drop the part between the #7 and #8 delimiters (including delimiters)

```

9189         \unexpanded{\unexpanded{#9}}% we replace the part of the redefined macro's
          meaning with the replacement text.
9191         \@nx\unexpanded{###3}%
9192     }% of inner \gmu@ampulexlet@resa
9193 }% of outer \gmu@ampulexlet@resa
9194 \gmu@ampulexlet@resa

```

Now `\gmu@ampulexlet@resa` carries the modifier of #4’s definition.

```

9199 \unless\ifx\czat#4%
9200     \edef\gmu@ampulexlet@resb{% double definition for double hashes of expanded
          \unexpanded{#1...}
9202         #1#2%
9203         \@nx#3\unexpanded{#5}{%
9204         \gmu@ampulexlet@resc}% Here we are sure the tokens sequences #7 and #8
          are in the one-level expansion of #4 so we don't pass them as sentinels
          (which BTW would totally spoil the redefinition, what it did indeed
          2010/6/23).
9209         }% of #3's definition body
9210     }% of inner \gmu@ampulexlet@resb
9211     \gmu@ampulexlet@resb
9212 \else
9213     \gmu@ifxany#2{\gdef\xdef}{\global}}}%
9214     #1\edef#3#5{\gmu@ampulexlet@resc}}}%
9215 \fi
9216 }% of if the delimiters were found in the meaning.
9217 }% of \ampulexlet

```

```

\ampulexdef 9220 \DeclareCommand\ampulexdef\long{%
9229 #1 Q{\outer\long\global\protected}} % (1) as \ampulexlet
9230 #2 T{\def\edef\gdef\xdef\pdef}{\def} % (2) ad \ampulexlet
9231 #3 m % (3) macro to be redefined,
9232 #4 O{} % (4) as \ampulexlet's #5, i.e., \def's parameters string; empty by default,
9233 #5 O{} % (5) as \ampulexlet's #6, i.e., definition body's parameters to be taken in
          a one-step expansion of the redefined macro; empty by default; the undelimited
          parameters should be double-braced here (but not doubled).

```

```

9237 #6 m % (6) start token(s),
9238 #7 m % (7) end token(s),
9239 #8 m % (8) the replacement
9240 }{%
9241 \DCUse\ampulexlet{#1}{#2}{#3}{#3}{#4}{#5}{#6}{#7}{#8}%
9242 }

```

## A definer for expandable loops

Now, as an example of use of `\ampulexlet`, we'll build a definer for expandable numerical loops.

```

9251 \def\gmu@ENumLoop#1#2{% this is a fully expandable loop generating #2 - #1 space
          tokens (cf. The  $\epsilon$ -TeX Manual p. 9).
9252
9253 \ifnum#1<#2 %
9254 \gmu@tempa
9255 \@xa\gmu@ENumLoop
9256 \@xa{\number\numexpr#1+1\@xa}%
9257 \@xa{\number#2\@xa}%
9258 \fi}% of \gmu@hashes.
9259
9260 \long\def\defENumLoop
9261 #1% the loop macro's name
9262 #2% the replacement of \gmu@tempa
9263 {%
9264 \ampulexlet#1\gmu@ENumLoop
9265 [##1##2][{##1}{##2}]%
9266 \gmu@tempa\@xa{#2\@xa}%
9267 \ampulexdef#1%
9268 [##1##2][{##1}{##2}]%
9269 \gmu@ENumLoop\@xa{#1\@xa}%
9270 }
9271 }

```

Let `\GenericInfo` write also to the terminal when `\tracingonline>0`.

```

9272
9273 \edef\GenericInfoToTerminal{%
9274 \unexpanded{%
9275 \@XA{\ampulexlet\protected\long\GenericInfo}\csname
9276 GenericInfo \endcsname[#1#2][{##1}{##2}]%
9277 \write\m@ne % we replace the token between these with:
9278 {\write\ifnum\tracingonline>\z@ \@unused\else\m@ne\fi}%
9279 }%
9280 }
9281
9282 \ampulexdef\@starttoc[#1][#1]\makeatletter\@input{%
          \makeatletter\NamedInput}
9283
9284 \</ampulex>

```

## The gmenvir Package<sup>6</sup>

```

9296 <utils> \gmu@PackOptionX{envir}

```

<sup>6</sup> This file has version number v0.996 dated 2011/10/12.

9297 `\*envir`

The `gmenviron.sty` package provides some improvements of the L<sup>A</sup>T<sub>E</sub>X's environments machinery. It provides a starred version of `begin` with which the CS respective to the argument doesn't have to be defined. This package also improves `end` by detokenising environment's name (which is equivalent to comparing the names of CS'es not strings of tokens). The package also provides some tests such as `\@ifenvir`.

For details just read the code part.

9310 `\RequirePackage{gmbase, gmampulex}` % the low-level macros

## Environments redefined

### Almost an environment or redefinition of `\begin`

We'll extend the functionality of `\begin`: the non-starred instances shall act as usual and we'll add the starred version. The difference of the latter will be that it won't check whether the 'environment' has been defined so any name will be allowed.

This is intended to structure the source with named groups that don't have to be especially defined and probably don't take any particular action except the scoping.

(If the `\begin*`'s argument is a (defined) environment's name, `\begin*` will act just like `\begin`.)

Original L<sup>A</sup>T<sub>E</sub>X's `\begin`:

```
\def\begin#1{%
  \@ifundefined{#1}%
    {\def\reserved@a{\@latex@error{Environment #1 undefined}\@eha}}%
    {\def\reserved@a{\def\@currentenv{#1}%
      \edef\@currentvline{\on@line}%
      \csname #1\endcsname}}%
  \@ignorefalse
  \begingroup\@endpfalse\reserved@a}
```

We provide a stack of environments consisting of triads `{\langle env. name \rangle}{\langle group level \rangle}{\langle beg. line \rangle}` (2010/6/9)

9344 `\emptify\@envirstack`

9346 `\def\@pushenvir{%`

`%% \edef\@currentenv{\@currentenv}% is already expanded.`

9348 `\xdef\@envirstack{%`

9349 `{\@xa\detokenize\@xa{\@currentenv}}%`

9350 `{\the\currentgrouplevel}%`

9351 `{\@currentvline}%`

9352 `\@envirstack`

9353 `}%`

9354 `}`

9356 `\def\@popenvir #1#2#3{%`

9357 `\@XA{\@popenvir@ #1#2#3}\@envirstack\@nil`

9358 `}`

9360 `\def\@popenvir@ #1#2#3#4#5#6#7\@nil{%`

9361 `\gdef #1{#4}% #1 carries last envir name`

9362 `\gdef #2{#5}% #2 carries last envir level`

9363 `\gdef #3{#6}% #3 carries last envir beginig line`

```

9364 \gdef\@envirstack{#7}% and we update the stack
9365 }
\@begnamedgroup 9369 \long\def\@begnamedgroup#1{%
9370 \edef\@prevgrouplevel{\the\currentgrouplevel}% added 2009/03/24 to han-
    dle special pseudo-environments that don't increase \currentgrouplevel(such
    as document). Note it's \edefed outside the environment's group.
9374 \@ignorefalse% not to ignore blanks after group
9375 \begingroup\@endpfalse
9376 \edef\@prevenvir{\@currenvir}% Note we \edef it inside the group (for obvious
    reason), unlike the 'previous' grouplevel.
9378 \edef\@currenvir{#1}% We could do recatcoding through \string or
    % \detokenize but all the name 'other' and 10 could affect a thousand pack-
    ages so we don't do that and we'll recatcode in a testing macro, see line 9426.
9383 \edef\@currenvline{\on@line}%
9384 \@pushenvir % we put current envir to \@envirstack.
9385 \csname #1\endcsname}% if the argument is a command's name (an environment's
    e.g.), this command will now be executed. (If the corresponding control se-
    quence hasn't been known to TEX, this line will act as \relax.)

```

Let us make it the starred version of `\begin`.

```

\begin* 9394 \def\begin{\gmu@ifstar{\@begnamedgroup}}{%
\begin 9395     \@begnamedgroup@ifcs}}
9398 \def\@begnamedgroup@ifcs#1{%
9399     \ifcsname#1\endcsname\afterfi{\@begnamedgroup{#1}}%
9400     \else\afterfi{\@latex@error{Environment #1 undefined}\@eha}%
9401     \fi}%

```

### `\@ifenvir` and improvement of `\end`

It's very clever and useful that `\end` checks whether its argument is `\ifx`-equivalent `\@currenvir`. However, in standard L<sup>A</sup>T<sub>E</sub>X it works not quite as I would expect: Since the idea of environment is to open a group and launch the CS named in the `\begin`'s argument. That last thing is done with `\csname... \endcsname` so the catcodes of chars are irrelevant (until they are `\active`, `_`, `2` etc.). Thus should be also in the `\end`'s test and therefore we ensure the compared texts are both expanded and made all 'other'.

`\gmu@ifedetokens` and `\@ifenvir` are defined in `gmbase`.

```

9423 \long\def\@fourthofmany#1#2#3#4#5\@nil{#4}%
9426 \lpdef\@ifprevenvir#1{%
    % #1 enquired environment name which will be confronted with \@prevenvir
    % #2 what if true (if the names are equivalent7)
    % #3 what if false

```

(2010/09/23, v0.993:) to be precise, it's not a change but rather a staus quo action: `\gmu@ifedetokens` suddenly turned to be expandable and un`\protected` so we make *this* macro `\protected`

```

9444 \gmu@ifedetokens
9445 {\@xa\@fourthofmany\@envirstack\relax\relax\relax\relax\@nil}%
9446 {#1}%
9447 }

```

<sup>7</sup> The names are checked whether they produce the same `\csname`. They don't have to have the same catcodes.



Note that `\@ifjobname` and `\@ifenvir` are expandable and in an `\edef` they expand to

```
\gmu@ifedetokens{<current jobname>}{<arg.1>}
```

and

```
\gmu@ifedetokens{<current envir>}{<arg.1>}
```

resp. which may be useful for some  $\TeX$ vert.

```
9457 \def\@checkend#1{%
9458   \@ifenvir{#1}%
9459   }%
9460   {\@badend{#1}}%
9461 }
```

Thanks to it you may write `\begin{macrocode*}` with  $\star_{12}$  and end it with `\end{macrocode*}` with  $\star_{11}$  (that was the problem that led me to this solution). The error messages looked really funny:

```
! LaTeX Error: \begin{macrocode*} on input line 1844 ended by
\end{macrocode*}.
```

You might also write also `\end{macrocode\star}` where `\star` is defined as ‘other’ star or letter star.

```
9475 \ampulexdef\end[#1][#1]\endcsname\@checkend{%
9476 \endcsname
9477 \@xa\gmu@ifempty\@xa{\@envirstack}%
9478 {%
9479   \PackageError {gmutils/base}%
9480   {There's no environment to pop!}{Oy vey, gefeelte fish!}%
9481 }%
9482 {\@popenvir\gmu@drain\gmu@drain\gmu@drain }%
9483 \@checkend
9484 }

9486 \pdef\@endif#1{\@ifenvir{#1}{\end{#1}}{}}
9488 \pdef\@endifprev#1{\@ifprevenvir{#1}{\end{#1}}{}}

\c@EnvirInterruption 9491 \newcount\c@EnvirInterruption

9493 \lpdef\gmu@InterruptEnvir
9494 #1% the contents of interruption.
9495 {%
9496   \global \advance\c@EnvirInterruption\@ne
9497   \Name \@popenvir
9498   {gmu@InterruptCurrenv \the\c@EnvirInterruption}\gmu@drain\gmu@drain
9499   \endgroup
9500   #1%
9501   \begingroup
9502   \@XA {\let\@currenvir}%
9503   \csname gmu@InterruptCurrenv \the\c@EnvirInterruption \endcsname
9504   \@pushenvir
9505   \global \advance\c@EnvirInterruption\m@ne
9506 }

9509 </envir>
```

## From relsize

```
9516 <utils>      \gmu@PackOptionX{relsize}
9517 <*relsize>
```

As file **relsize.sty**, v3.1 dated July 4, 2003 states, L<sup>A</sup>T<sub>E</sub>X 2<sub>ε</sub> version of these macros was written by Donald Arseneau [asnd@triumf.ca](mailto:asnd@triumf.ca) and Matt Swift [swift@bu.edu](mailto:swift@bu.edu) after the L<sup>A</sup>T<sub>E</sub>X 2.09 **smaller.sty** style file written by Bernie Cosell [cosell@WILMA.BBN.COM](mailto:cosell@WILMA.BBN.COM).

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

```
\relsize      You declare the font size with \relsize{<n>} where <n> gives the number of steps
               ("mag-step" = factor of 1.2) to change the size by. E.g., n = 3 changes from \nor|
\smaller     malsize to \LARGE size. Negative n selects smaller fonts. \smaller == \relsize{-1};
\larger      \larger == \relsize{1}. \smallerr(my addition) == \relsize{-2}; \largerr guess
\smallerr    yourself.
\largerr     (Since \DeclareRobustCommand doesn't issue an error if its argument has been defined
               and it only informs about redefining, loading relsize remains allowed.)
```

```
\relsize 9542 \protected\def\relsize#1{%
          9543   \ifmmode \@nomath\relsize\else
          9544     \begingroup
          9545       \@tempcnta % assign number representing current font size
          9546         \ifx\@currsize\normalsize 4\else % funny order is to have most ...
          9547         \ifx\@currsize\small 3\else % ...likely sizes checked first
          9548         \ifx\@currsize\footnotesize 2\else
          9549         \ifx\@currsize\large 5\else
          9550         \ifx\@currsize\Large 6\else
          9551         \ifx\@currsize\LARGE 7\else
          9552         \ifx\@currsize\scriptsize 1\else
          9553         \ifx\@currsize\tiny 0\else
          9554         \ifx\@currsize\huge 8\else
          9555         \ifx\@currsize\Huge 9\else
          9556         4\rs@unknown@warning % unknown state: \normalsize as start-
                                   ing point
          9557       \fi\fi\fi\fi\fi\fi\fi\fi\fi\fi
```

Change the number by the given increment:

```
9559     \advance\@tempcnta#1\relax
```

watch out for size underflow:

```
9561     \ifnum\@tempcnta<\z@ \rs@size@warning{small}{%
          \string\tiny}\@tempcnta\z@ \fi
9562     \@xa\endgroup
9563     \ifcase\@tempcnta % set new size based on altered number
9564     \tiny \or \scriptsize \or \footnotesize \or \small \or
          \normalsize \or
9565     \large \or \Large \or \LARGE \or \huge \or \Huge \else
9566     \rs@size@warning{large}{\string\Huge}\Huge
9567 \fi\fi}% end of \relsize.
```

```
\rs@size@warning 9570 \providecommand*\rs@size@warning[2]{\PackageWarning{gmutils
                  (relsize)}{%
9571     Size requested is too #1.\MessageBreak Using #2 instead}%
```

```

9572 }%
\rs@unknown@warning 9575 \providecommand*\rs@unknown@warning{\PackageWarning{gmutils
      (relsize)}{Current font size
9576 is unknown! (Why?!?)\MessageBreak Assuming \string\normalsize}}
      And a handful of shorthands:

```

```

\larger 9580 \DeclareRobustCommand*\larger[1][\@ne]{\relsize{+#1}}
\smaller 9581 \DeclareRobustCommand*\smaller[1][\@ne]{\relsize{-#1}}
\textlarger 9582 \DeclareRobustCommand*\textlarger[2][\@ne]{\relsize{+#1}#2}}
\textsmaller 9583 \DeclareRobustCommand*\textsmaller[2][\@ne]{\relsize{-#1}#2}}
9584 \protected\def\largerr{\relsize{+2}}
9585 \protected\def\smallerr{\relsize{-2}}

```

We could implement continuous growth: `\larger[2.567]` means predefined fontsize 2 steps up plus .567 of the difference between step 2 and step 3.

```
9592 </relsize>
```

## The gmmeta package for meta-symbols

```

9599 <utils> \gmu@PackOptionX{meta} % provides \bihyphen, \discre, \discret
9600 <*meta>
9602 \RequirePackage{gmcommand}

```

I fancy also another Knuthian trick for typesetting *<meta-symbols>* in *The T<sub>E</sub>X book*. So I repeat it here. The inner `\meta` macro is copied verbatim from `doc`'s v2.1b documentation dated 2004/02/09 because it's so beautifully crafted I couldn't resist. I only don't make it `\long`.

The new implementation fixes this problem by defining `\meta` in a radically different way: we prevent hyphenation by defining a `\language` which has no patterns associated with it and use this to typeset the words within the angle brackets.

```
9618 \pdef\meta#1{%
```

Since the old implementation of `\meta` could be used in math we better ensure that this is possible with the new one as well. So we use `\ensuremath` around `\langle` and `\rangle`. However this is not enough: if `\meta@font@select` below expands to `\itshape` it will fail if used in math mode. For this reason we hide the whole thing inside an `\nfss@text` box in that case.

```

9626 {\meta@fontsetting\ensuremath\langle}%
9627 \ifmmode\@xa\nfss@text\fi
9628 {% this has to be a begin-group because \nfss@text becomes \hbox in math mode.
9630 \gmu@activespaceblank
9631 \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.

```

9634 #1\%
9635 }%
9636 {\meta@fontsetting\ensuremath\rangle}%
9637 }% of \meta.

```

```

9639 \pdef\gmu@activespaceblank{%
9640   \@xa\def\gmu@activespace{\space\ignorespaces}% note the subtle perversity
      of this definition: if we meet more than one subsequent active spaces, then the
      first of them will typeset \space and its \ignorespaces will gobble \space
      of the second and will stop at \ignorespaces and this \ignorespaces will
      gobble the next \space and so on.

```

```

9647 }
9649 \def\meta@fontsetting{\color{red!85!black}}
9651 \def\meta@font@select{\meta@fontsetting\it}

```

But I define `\meta@font@select` as the brutal and explicit `\it` instead of the original `\itshape` to make it usable e.g. in the `gmdoc`'s `\cs` macro's argument.

The below `\meta`'s drag<sup>8</sup> is a version of *The T<sub>E</sub>X book*'s one.

```

\<.> 9663 \def\<#1>{\meta{#1}}
9665 \pdef\metachar#1{\begingroup\metacharfont #1\endgroup}
9666 \def\metacharfont{\meta@fontsetting\rm}

```

## Macros for printing macros and filenames

The `\discre` macro is defined in `gmbase`. It works like `\discretionary` but allows hyphenation before and after itself.

```

9673 \pdef\vs{\discre{\visiblepace}{}{\visiblepace}}

```

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).

```

9679 \def\printspaces#1{{\let~=\vs \let\ =\vs \gmu@pswords#1 \@nil}}
9681 \def\gmu@pswords#1 #2\@nil{%
9682   \ifx\relax#1\relax\else#1\fi
9683

```

```

          \ifx\relax#2\relax\else\vs\penalty\hyphenpenalty\gmu@pswords#2\@nil\fi}% no
          that in the recursive call of \gm@pswords the argument string is not
          extended with a sentinel space: it has been already by \printspaces.

```

```

9688 \pdef\sfname#1{\textsf{\printspaces{#1}}}
9690 \def\gmu@discretionaryslash{\discre{/}{\hbox{}}{/}}% the second
      pseudo-argument nonempty to get \hyphenpenalty not \exhyphenpenalty.
9695 \pdef\file#1{\gmu@printslashes#1/\gmu@printslashes}
9697 \def\gmu@printslashes#1/#2\gmu@printslashes{%
9698   \sfname{#1}%
9699   \ifx\gmu@printslashes#2\gmu@printslashes
9700   \else
9701   \textsf{\gmu@discretionaryslash}%
9702   \afterfi{\gmu@printslashes#2\gmu@printslashes}\fi}

```

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

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

---

<sup>8</sup> Think of the drags that transform a very nice but rather standard 'auntie' ('Tante' in Deutsch) into a most adorable Queen ;-).

```
9709 \pdef\pk#1{\textsf{#1}}
```

Some (if not all) of the below macros are copied from **doc** and/or **ltxdoc**.

A macro for printing control sequences in arguments of a macro. Robust to avoid writing an explicit `\` into a file. It calls `\ttfamily` not `\tt` to be usable in headings which are boldface sometimes.

```
\cs 9723 \DeclareCommand\cs{O{\type@bslash\penalty\M\hskip\z@skip}}{%
      % [#1] O{\bslash} the control sequence's prefix, by default it's \ allowing hy-
      % #2 m the control sequence or anything to be typeset in typewriter font.
9734 \begingroup
9735 \ifdefined\verbatim@specials\verbatim@specials\fi
9736 \edef\-\{\discretionary{%
9737   \ifdefined\gmv@hyphen\gmv@hyphen
9738   \else\unexpanded{\normalfont-}}%
9739   \fi}{}}}%
9740 \def\{\{\type@lbrace\yeshy\def\}\{\char`\}}}%
9741 \narrativett
9742 \edef\narrativett@storedhyphenchar{\the\hyphenchar\font}%
9743 \hyphenchar\font=%
9744 \ifdefined\gmv@hyphenchar\gmv@hyphenchar
9745 \else "A6
9746 \fi
9747 \cs@inner{#1}%
9748 }% of \cs

9751 \let\type@bslash\bslash

9754 \pdef\cs@inner#1#2{%
9755   #1#2%
```

`% \hyphenchar\font=\narrativett@storedhyphenchar\relax` % we don't re-store the value of `\hyphenchar` since this restores it back for the entire paragraph.

```
9760 \endgroup}
```

```
9762 \def\narrativett{\ttfamily}% such name because I introduce it to distinguish the
      narrative verbatims from the code in gmdoc.
```

```
9765 \long\pdef\env{\cs[]}
```

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

```
9769 \foone{\@makeother\^}
9770 {\pdef\hathat{\cs[^^]}}
```

```
9772 \AtBeginDocument{%
9773   \@ifpackageloaded{gmdoc}{\def\hash{\cs[#]}}{}}
```

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

```
9776 \def\possfil{\hfil\penalty1000\hfilneg}

9778 \def\possvfil{\vfil
9779   \penalty\numexpr
9780   \gmu@minnum{\clubpenalty+\widowpenalty}{9999}%
9781   \relax % eaten by \gmu@maxnum
9782   \relax % eaten by \numexpr
9783   \vfilneg}
```

## Typesetting arguments and commands

`\arg` We define a conditional and iterating command `\arg` that in math mode does what it used to do was in math and outside math it typesets mandatory, optional and picture (parenthesed) and angled arguments and optional stars. You can write

```
\arg[gefüllte]*<fisch>(mit){baigele}
```

to get

```
[<gefüllte>][*]{<fisz>}(<mit>){<baigele>}
```

or even

```
\verb+\MoltoAdagio</arg>*<Dankgesang><an>[die Gottheit]+
```

(where `</` is the escape char in verbatims) to get

```
\MoltoAdagio[*]{<Dankgesang>}<an>[<die Gottheit>]
```

(in der lydischen Tonart).

For more complicated arguments configurations consider using `gmdoc`'s environment `enumargs`.

The five macros below are taken from the `ltxdoc.dtx`.

`\cmd{\foo}` Prints `\foo` verbatim. It may be used inside moving arguments. `\cs{foo}` also prints `\foo`, for those who prefer that syntax. (This second form may even be used when `\foo` is `\outer`).

```
9811 \long\def\cmd#1{\@xa\cs\@xa{\@xa\cmd@to@cs\string#1}\spifletter}% it has
    to be un \protected! It has so many \expandafter s to allow \cmd\par and
    still keep the \cs command 'short'.
```

```
9815 \def\cmd@to@cs#1#2{\char\number`#2\relax}
```

It can be short since it never gets actual control sequence as an argument only a string of 'other' tokens (and maybe spaces).

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

```
\marg 9821 \pdef\marg#1{\narrativett\type@lbrace}\arg@wrap{#1}{%
    \narrativett\char`\\}}
```

`\oarg{text}` prints `[<text>]`, 'optional argument'. Also `\oarg{text}` does that.

```
\oarg 9827 \pdef\oarg{\@ifnextchar[\@oargsq\@oarg}
9829 \pdef\@oarg#1{\narrativett[]\arg@wrap{#1}{\narrativett}}
9830 \pdef\@oargsq[#1]{\@oarg{#1}}
```

`\parg{te,xt}` prints `(<te,xt>)`, 'picture mode argument'.

```
\parg 9834 \pdef\parg{\@ifnextchar(\@pargp\@parg}
9836 \def\@parg#1{\narrativett()\arg@wrap{#1}{\narrativett}}
9837 \def\@pargp(#1){\@parg{#1}}
```

```
9839 \pdef\aaarg{\@ifnextchar<\@aarga\@aarg}
9840 \def\@aarg#1{\narrativett<\arg@wrap{#1}{\narrativett>}}
9841 \def\@aarga<#1>{\@aarg{#1}}
```

```
9843 \def\@verbaarga#1#2>{\@aarg{#2}\arg@dc}
```

```
9845 \foone{\catcode`>\active}{%
```

```
9846 \def\@verbaargact#1#2>{\@aarg{#2}\arg@dc}%
```

```
9847 }
```

```

9849 \foone{\@makeother\{\@makeother\}}%
9850 \catcode`[=\@ne\catcode`] =\tw@}
9851 [%
9852 \def\@verbmargm#1#2}{% for an argument in curly braces in a verbatim, where the
      braces are not groupers and not necessary ‘other’. We’ll know by \@ifnextif
      that the future token is an opening brace. Note this macro has 2nd parameter
      delimited with ‘other’ closing brace (so may not act correctly when braces are
      nested (then hide them with special verbatim groupers)).
9858 \marg[#2]%
9859 \arg@dc
9860 ]%
9861 ]% of \foone

```

Now provide the default `\arg@wrap`, for meta-arguments that is.

```

9865 \def\arg@wrap{\meta}
\arg@dc 9869 \DeclareCommand\arg@dc!{%
9872 s % (1)
9873 o % (2)
9874 c % (3)
9875 b % (4)
9876 a % (5)
9877 T{\arg} % (6) just gobbled (for backwards compatibility)
9878 }{% This command iterates while it has arguments and typesets them in brackets,
      parentheses or curly braces. Note it gobbles subsequent \args and just iterates.
9881 \def\next{0}%
9882 \gmuIfValueT{#1}%
9883 {\metachar[\scanverb{*}\metachar]\def\next{1}}%
9884 \gmuIfValueT{#2}{\@oarg{#2}\def\next{1}}%
9885 \gmuIfValueT{#3}{\@parg{#3}\def\next{1}}%
9886 \gmuIfValueT{#4}{\marg{#4}\def\next{1}}%
9887 \gmuIfValueT{#5}{\aarg{#5}\def\next{1}}%
9888 \@ifnextchar\egroup{\endgroup}{%
9889 \if1\next\@xa\arg@dc
9890 \else % it’s crucial that we look for verbatim braces after we checked there were
      no #4, otherwise there would be an error.
9893 \def\next{%
9894 \@ifnextif\xiilbrace{\@verbmargm}%
9895 {% not active or other lbrace
9896 \@ifnextif<{% then we look for angles
9897 \ifnum\catcode`>=\active
9898 \@xa\@verbaargact
9899 \else\@xa\@verbaarga
9900 \fi}%
9901 {% and if not angles neither verbatim braces, then
9902 \endgroup % if we have no more arguments to typeset, we close the
      group opened in lines 9922 and 9939.
9904 \spifletter
9905 }%
9906 }%
9907 }%
9908 \@xa \next
9909 \fi

```

9910 }% of not egroup  
 9911 }% of \arg@dc

Now define the front-end macro of the \arg command:

```
9915 \foone{\obeylines\@makeother\^^C}{%
9916   \AtBeginDocument{%
9917     \let\math@arg\arg %
9918     \pdef\arg{% This is \arg for meta-arguments.
9919       \ifmmode\math@arg %
9920       \else\afterfi{%
9921         \begingroup %
9922         \ifdefined\@ifQueerEOL\@ifQueerEOL{%
9923           \def^^M{\unskip\space}% in the 'queer' EOLs scope we keep line end
9924             active in case we have \arg {<arg.>} ending a line: the next char
9925             peeper touches line end or, if the line end was 5, gobbles the space
9926             it turns into so the comment layer would 'leak' to the code layer.
9927           }{} \fi %
9928           \arg@dc}%
9929         \fi}% of \arg,
9930   }% of \AtBeginDocument,
```

And this is arg-typesetting command for verbatim arguments.

```
9938 \pdef\argv{%
9939   \begingroup %
9940   \@makeother\^^C%
9941   \pdef\arg@wrap##1{%
9942     \narrativett{##1}%
9943   }%
9944   \ifdefined\@ifQueerEOL\@ifQueerEOL{%
9945     \def^^M{\unskip\space}% in the 'queer' EOLs scope we keep line end ac-
9946       tive... as above
9947   }{}%
9948   \fi % of \ifdefined
9949   \arg@dc}%
9950 }% of \foone.
```

Now you can write

\arg{mand.\ arg} [opt.\ arg] (pict.\ arg)  
 to get {<mand. arg>} [<opt. arg>] (<pict. arg>). (Yes, with only one \arg!)  
 And  $\$ \arg(1+i) = \pi/4$  for  $\arg(1+i) = \pi/4$ .

```
\cat 9963 \DeclareCommand\cat{Q{"0123456789ABCDEF}{0}}{%
9964   $}_{\the\numexpr#1}\m@th$\spifletter
9965 }
```

A shorthand for \CS:

```
9968 \pdef\CS{%
9969   \acro{CS}%
9970   \@ifnextcat a{ }{}% we put a space if the next token is 11. It's the next best
9971     thing to checking whether the CS consisting of letters is followed by a space.
9972 }
9973 \pdef\CSs{\CS{}es\@ifnextcat a{ }{}% for pluralis.
9974 \pdef\CSes{\CS{}es\@ifnextcat a{ }{}% for pluralis.
9975 }
9976 \meta
```



## The gmlogos package—a couple of T<sub>E</sub>X-related logos

```
9986 <utils>      \gmu@PackOptionX{logos}
9987 <*logos>
```

```
9989 \RequirePackage{gmbase}
```

We'll modify The L<sup>A</sup>T<sub>E</sub>X logo now to make it fit better to various fonts.

```
9995 \let\oldLaTeX\LaTeX
9996 \let\oldLaTeXe\LaTeXe
```

```
9998 \pdef\TeX{T\kern-.1667em\lower.5ex\hbox{E}\kern-.125emX\@}
```

```
9999 \Store@Macro\TeX
```

```
10000 \AtBeginDocument{\Restore@Macro\TeX}
```

```
10002 \lpdef\EofTeX
```

```
10003 #1#2% whether put kerns before and after »E« respectively.
```

```
10004 #3% we do nothing with it (maybe unbrace) but we want to gobble possible space.
```

```
10006 {\if1#1\relax \kern-.1667em \fi
```

```
10007 \lower.5ex\hbox{E}%
```

```
10008 \if1#2\relax \kern-.125em \fi
```

```
10009 #3
```

```
10010 }
```

```
\DeclareLogo 10012 \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.
```

```
10020 \ifx\relax#1\def\gmu@DeclareLogo@resa{\@xa@gobble\string#2}%
```

```
10021 \else
```

```
10022 \def\gmu@DeclareLogo@resa{#1}%
```

```
10023 \fi
```

```
10024 \edef\gmu@DeclareLogo@resa{%
```

```
    \@nx\DeclareTextCommand\@nx#2{PD1}{\gmu@DeclareLogo@resa}}
```

```
10026 \gmu@DeclareLogo@resa
```

```
10027 \DeclareTextCommandDefault#2{#3}%
```

```
\pdef 10028 \pdef#2{#3}% added for XYLATEX.
```

```
10029 }
```

```
10032 \DeclareLogo\LaTeX{%
```

```
10033 {%
```

```
10034 L%
```

```
10035 \setbox\z@\hbox{\check@mathfonts
```

```
10036 \fontsize\sf@size\z@
```

```
10037 \math@fontsfalse\selectfont
```

```
10038 A}%
```

```
10039 \kern-.57\wd\z@
```

```
10040 \sbox\tw@ T%
```

```
10041 \vbox to\ht\tw@{\copy\z@ \vss}%
```

```
10042 \kern-.2\wd\z@% originally –, 15 em for T.
```

```
10043 }%
```

```
10044 {%
```

```
10045 \ifdim\fontdimen1\font=\z@
```

```

10046 \else
10047 \count\z@=\fontdimen5\font
10048 \multiply\count\z@ by 64\relax
10049 \divide\count\z@ by\p@
10050 \count\tw@=\fontdimen1\font
10051 \multiply\count\tw@ by\count\z@
10052 \divide\count\tw@ by 64\relax
10053 \divide\count\tw@ by\tw@
10054 \kern-\the\count\tw@ sp\relax
10055 \fi}%
10056 \gmlogos@hyphen
10057 \TeX
10058 }

\LaTeXe 10060 \DeclareLogo\LaTeXe{\mbox{\m@th \if
10061 b\expandafter\@car\@f@series\@nil\boldmath\fi
10062 \LaTeX\kern.15em2$_{\textstyle\varepsilon}$}}

10064 \Store@Macro\LaTeX
10065 \Store@MacroSt{LaTeX }

```

‘(L)TEX’ in my opinion better describes what I work with/in than just ‘L<sup>A</sup>TEX’.

```

\LaTeXpar 10071 \DeclareLogo[(La)TeX]{\LaTeXpar}{%
10072 {%
10073 \setbox\z@\hbox{(% )}
10074 \leavevmode %
10075 \copy\z@
10076 \kern-.2\wd\z@ L%
10077 \setbox\z@\hbox{\check@mathfonts
10078 \fontsize\sf@size\z@
10079 \math@fontsfalse\selectfont
10080 A}%
10081 \kern-.57\wd\z@
10082 \sbox\tw@ T%
10083 \vbox to\ht\tw@{\box\z@%
10084 \vss}%
10085 }%
10086 \kern-.07em% originally -,15 em for T.
10087 {% (
10088 \sbox\z@)%
10089 \kern-.2\wd\z@\copy\z@
10090 \kern-.2\wd\z@}\gmlogos@hyphen\TeX
10091 }
10092 }

```

Here are a few definitions which can usefully be employed when documenting package files: now we can readily refer to  $\mathcal{A}\mathcal{M}\mathcal{S}$ -TEX, BIBTEX and SL<sup>A</sup>TEX, as well as the usual TEX and L<sup>A</sup>TEX. There’s even a PLAIN TEX and a WEB.

```

10099 \gmu@ifundefined{AmSTeX}
10100 {\def\AmSTeX{\leavevmode\hbox{\mathcal A\kern-.2em\lower.376ex%
10101 \hbox{\mathcal M$}\kern-.2em\mathcal S$-\TeX}}{}}

\BibTeX 10103 \DeclareLogo\BibTeX{\rmfamily B\kern-.05em%
10104 \textsc{i{\kern-.025em}b}\kern-.08em% the kern is wrapped in braces for
my \fakescaps’ sake.

```

```

10106     \TeX}}
\SlitTeX 10109 \DeclareLogo\SlitTeX{\rmfamily
           S\kern-.06emL\kern-.18em\raise.32ex\hbox
10110           {\scshape i}\kern -.03em\TeX}}
\PlainTeX 10112 \DeclareLogo\PlainTeX{\textsc{Plain}\kern2pt\TeX}
\Web 10114 \DeclareLogo\Web{\textsc{Web}}

```

There's also the (L<sup>A</sup>)T<sub>E</sub>X logo got with the `\LaTeXpar` macro provided by `gmutils`. And here *The T<sub>E</sub>X book*'s logo:

```

\TeXbook 10117 \DeclareLogo[The TeX book]\TeXbook{\textsl{The \TeX\space book}}
10118 \let\TB\TeXbook% TUG Boat uses this.

\eTeX 10120 \DeclareLogo[e-TeX]\eTeX{%
10121   \iffontchar\font"03B5{\itshape ε}\else
10122   \ensuremath{\varepsilon}\fi-\kern-.125em\TeX}% definition sent by Karl Berry
           from TUG Boat itself.

10125 \Store@Macro\eTeX

\pdfTeX 10127 \DeclareLogo[pdfe-TeX]\pdfTeX{pdf\gmlogos@hyphen\eTeX}
\pdfTeX 10129 \DeclareLogo\pdfTeX{pdf\gmlogos@hyphen\TeX}
\pdfLaTeX 10130 \DeclareLogo\pdfLaTeX{pdf\gmlogos@hyphen\LaTeX}

```

Before version v0.996 the below was in a conditional but the

```

10135 \gmu@ifundefined{XeTeX}{%
\XeTeX 10136   \DeclareLogo\XeTeX{X\kern-.125em\relax
10137     \gmu@ifundefined{reflectbox}{%
10138       \lower.5ex\hbox{E}\kern-.1667em\relax}{%
10139       \lower.5ex\hbox{\reflectbox{E}}\kern-.1667em\relax}%
10140     \TeX
10141   }%
10142 }{}

10148 \gmu@ifundefined{XeLaTeX}{%
\XeLaTeX 10149   \DeclareLogo\XeLaTeX{X\kern-.125em\relax
10150     \gmu@ifundefined{reflectbox}{%
10151       \lower.5ex\hbox{E}\kern-.1667em\relax}{%
10152       \lower.5ex\hbox{\reflectbox{E}}\kern-.1667em\relax}%
10153     \LaTeX}}{}

```

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

```

\XeTeXpar 10161 \DeclareLogo\XeTeXpar{%
10162   \setbox\z@\hbox{()% }
10163   \leavevmode
10164   \copy\z@
10165   \kern-.2\wd\z@
10166   \smash{% the "Xe" part is copied from xltextra
10167     X\lower0.5ex
10168     \hbox{\kern-0.15em
10169       \gmu@ifundefined{XeTeXversion}%

```

```

10170     {\setbox0=\hbox{E}\dimen0=\ht0\advance\dimen0by\dp0%
10171      \raise\dimen0\hbox{\rotatebox{180}{\box0}}}%
10172   }% of if not in XƒTEX, then in XƒTEX:
10173   {\ifnum\XeTeXfonttype\font>0
10174     \ifnum\XeTeXcharglyph"018E>0
10175       \char"018E\relax
10176     \else
10177       \ifdim\fontdimen1\font=0pt
10178         \reflectbox{E}%
10179       \else
10180         \XeTeXuseglyphmetrics=1%
10181         \setbox0=\hbox{E}\dimen0=\ht0\advance\dimen0by\dp0%
10182         \raise\dimen0\hbox{\rotatebox{180}{\box0}}}%
10183     \fi
10184   \fi
10185   \else
10186     \setbox0=\hbox{E}\dimen0=\ht0\advance\dimen0by\dp0%
10187     \raise\dimen0\hbox{\rotatebox{180}{\box0}}}%
10188   \fi}% of reversed E when in XƒTEX
10189 }% of hbox
10190 }% of smash
10191 \setbox\z@\hbox{)}%
10192 \kern-.2\wd\z@
10193 \copy\z@
10194 \kern-0.15em
10195 \TeX}%

\LuaTeX 10198 \DeclareLogo [LuaTeX]\LuaTeX{\textsc{Lua}\gmlogos@hyphen\TeX}
\ConTeXt 10201 \DeclareLogo [ConTeXt]\ConTeXt{Con\gmlogos@hyphen\TeX t}

10204 \emptify\gmlogos@hyphen

10206 \def\HyphenateLogo#1{%
10207   {\let\gmlogos@hyphen\-%
10208     #1}%
10209 }

10211 </logos>

```

## The gmnotonlypream—modification of the ‘only preamble’ clause

```

10218 <utils> \gmu@PackOptionX{notonlypream}
10219 <*notonlypream>

10222 \RequirePackage{gmampulex}

```

## Not only preamble!

Let’s remove some commands from the list to erase at begin document! Primarily that list was intended to save memory not to forbid anything. Nowadays, when memory is cheap, the list of only-preamble commands should be rethought IMHO.

```

\not@onlypreamble 10237 \newcommand\not@onlypreamble[1]{{%
10238   \def\do##1{\ifx##1\else\nx\do\nx##1\fi}%
10239   \xdef\@preamblecmds{\@preamblecmds}}
10241 \not@onlypreamble\@preamblecmds
10242 \not@onlypreamble\@ifpackageloaded
10243 \not@onlypreamble\@ifclassloaded
10244 \not@onlypreamble\@ifl@aded
10245 \not@onlypreamble\@pkgextension

```

We use the two below e.g. in `\NamedInput` which we surely want to allow also within document.

```

10249 \not@onlypreamble\@pushfilename
10250 \not@onlypreamble\@popfilename
10252 \not@onlypreamble\@currnamestack

```

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

```

10258 \def\gmu@notprerr{ can be used only in preamble (\on@line)}
10260 \AtBeginDocument{%
10261   \def\do#1{\@nx\do\nx#1}%
10262   \edef\@preamblecmds{%
10263     \def\@nx\do##1{%
10264       \def##1{\@nx\gmu@NotprerrMessage##1}\@nx\@eha}}%
10265   \@preamblecmds}
10267 \def\gmu@NotprerrMessage#1{%
10268   \PackageError{gmutils/LaTeX}%
10269   {\@nx\string#1 \@nx\gmu@notprerr}}}%
10270 }

```

A subtle error raises: the L<sup>A</sup>T<sub>E</sub>X standard `\@onlypreamble` and what `\document` does with `\@preamblecmds` makes any two of 'only preamble' CS's `\ifx`-identical inside document. And my change makes any two CS's `\ifx`-different. The first it causes a problem with is standard L<sup>A</sup>T<sub>E</sub>X's `\nocite` that checks `\ifx\@onlypreamble\document`. So hoping this is a rare problem, we circumvent it. 2008/08/29 a bug is reported by Edd Barrett that with `natbib` an 'extra }' error occurs so we wrap the fix in a conditional.

```

10288 \def\gmu@nocite@ampulex{% we wrap the stuff in a macro to hide an open \if. And
      not to make the begin-input hook too large. the first optional argument is the pa-
      rameters string and the second the argument for one-level expansion of \nocite.
      Both hash strings are doubled to pass the first \def.
10294   \ampulexdef\nocite[##1][##1]
10295   \ifx
10296   {\@onlypreamble\document}%
10297   \iftrue}
10300 \AtBeginDocument{\gmu@nocite@ampulex}%
10301 </notonlypream>

```

## Improvements to mwcls sectioning commands

```

10308 <utils>   \gmu@PackOptionX{mw}

```

```
10309 <*mw>
10310 \RequirePackage{gmcommand}
```

That is, ‘Expe-ri-mente’<sup>9</sup> mit MW’s 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. pdf<sub>TEX</sub> cried of multiply defined `\labels`,
2. e.g. in a table of contents the hyperlink `<rozdzia\1\ Kwiaty polskie>` linked not to the chapter’s heading but to the last-before-it change of `\ref`.

```
10326 \AtBeginDocument{% because we don't know when exactly hyperref is loaded and
      maybe after this package.
NoNumSecs 10328 \@ifpackageloaded{hyperref}{\newcounter{NoNumSecs}%
10329 \setcounter{NoNumSecs}{617}% to make \refing to an unnumbered section vis-
      ible (and funny?).
10331 \def\gmu@hyperrefstepcounter{\refstepcounter{NoNumSecs}}%
10332 \pdef\gmu@targetheading#1{%
10333 \hypertarget{#1}{#1}}% end of then
10334 {\def\gmu@hyperrefstepcounter{}%
10335 \def\gmu@targetheading#1{#1}}% end of else
10336 }% of \AtBeginDocument
```

Auxiliary macros for the kernel sectioning macro:

```
10339 \def\gmu@dontnumbersectionsoutofmainmatter{%
10340 \if@mainmatter\else \HeadingNumberedfalse \fi}
10341 \def\gmu@clearpagesduetoopenright{%
10342 \if@openright\cleardoublepage\else \clearpage\fi}
```

To avoid `\def`ing of `\mw@sectionxx` if it’s undefined, we redefine `\def` to gobble the definition and restore the original meaning of itself.

Why shouldn’t we change the ontological status of `\mw@sectionxx` (not define if undefined)? Because some macros (in `gmdocc` e.g.) check it to learn whether they are in an `mwcls` or not.

But let’s make a shorthand for this test since we’ll use it three times in this package and maybe also somewhere else.

```
\@ifnotmw 10355 \long\def\@ifnotmw#1#2{\gmu@ifundefined{mw@sectionxx}{#1}{#2}}
```

The kernel of MW’s sectioning commands:

```
10360 \@ifnotmw{}{%
10361 \def\mw@sectionxx#1#2[#3]#4{%
10362 \edef\mw@HeadingLevel{\csname #1@level\endcsname
10363 \space}% space delimits level number!
10364 \ifHeadingNumbered
10365 \ifnum \mw@HeadingLevel>\c@secnumdepth \HeadingNumberedfalse \fi
line below is in \gmu@ifundefined to make it work in classes other than mwbk
10368 \gmu@ifundefined{if@mainmatter}{%
      \gmu@dontnumbersectionsoutofmainmatter}
10369 \fi
      % \ifHeadingNumbered
      % \refstepcounter{#1}%
```

---

<sup>9</sup> A. Berg, *Wozzeck*.

```

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

10378 \def\HeadingRHeadText{#2}%
10379 \def\HeadingTOCText{#3}%
10380 \def\HeadingText{#4}%
10381 \def\mw@HeadingType{#1}%
10382 \if\mw@HeadingBreakBefore
10383 \if@specialpage\else\thispagestyle{closing}\fi
10384 \gmu@ifundefined{if@openright}{}{\gmu@clearpagesduetoopenright}%
10385 \if\mw@HeadingBreakAfter
10386 \thispagestyle{blank}\else
10387 \thispagestyle{opening}\fi
10388 \global\@topnum\z@
10389 \fi% of \if\mw@HeadingBreakBefore

```

placement of \refstep suggested by me (GM):

```

10392 \ifHeadingNumbered
10393 \refstepcounter{#1}%
10394 \protected@edef\HeadingNumber{\csname the#1\endcsname\relax}%
10395 \else
10396 \let\HeadingNumber\@empty
10397 \gmu@hyperrefstepcounter % we step an auxiliary counter to make a hyperref's
label/target.
10399 \fi% of \ifHeadingNumbered

10401 \if\mw@HeadingRunIn
10402 \mw@runinheading
10403 \else
10404 \if\mw@HeadingWholeWidth
10405 \if@twocolumn
10406 \if\mw@HeadingBreakAfter
10407 \onecolumn
10408 \mw@normalheading
10409 \pagebreak\relax
10410 \if@twoside
10411 \null
10412 \thispagestyle{blank}%
10413 \newpage
10414 \fi% of \if@twoside
10415 \twocolumn
10416 \else
10417 \@topnewpage[\mw@normalheading]%
10418 \fi% of \if\mw@HeadingBreakAfter
10419 \else
10420 \mw@normalheading
10421 \if\mw@HeadingBreakAfter\pagebreak\relax\fi
10422 \fi% of \if@twocolumn
10423 \else
10424 \mw@normalheading
10425 \if\mw@HeadingBreakAfter\pagebreak\relax\fi
10426 \fi% of \if\mw@HeadingWholeWidth

```

```

10427 \fi% of \if\mw@HeadingRunIn
10428 }

```

### An improvement of MW's \SetSectionFormatting

A version of MW's \SetSectionFormatting that lets to leave some settings unchanged by leaving the respective argument empty ({} or []).

Notice: If we adjust this command for new version of MWCLS, we should name it \SetSectionFormatting and add issuing errors if the inner macros are undefined.

```

[#1] the flags, e.g. breakbefore, breakafter;
#2 the sectioning name, e.g. chapter, part;
#3 preskip;
#4 heading type;
#5 postskip

```

```

10451 \relaxen\SetSectionFormatting
\SetSectionFormatting 10452 \newcommand*\SetSectionFormatting[5][\empty]{%
10453 \ifx\empty#1\relax\else% empty (not \empty!) #1 also launches \else.
10454 \def\mw@HeadingRunIn{10}\def\mw@HeadingBreakBefore{10}%
10455 \def\mw@HeadingBreakAfter{10}\def\mw@HeadingWholeWidth{10}%
10456 \gmu@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.

10459 \fi
10460 \gmu@ifundefined{#2}{\@namedef{#2}{\mw@section{#2}}}{}%
10461 \mw@secdef{#2}{@preskip}{#3}{2 oblig.}%
10462 \mw@secdef{#2}{@head}{#4}{3 oblig.}%
10463 \mw@secdef{#2}{@postskip}{#5}{4 oblig.}%
10464 \ifx\empty#1\relax
10465 \mw@secundef{#2@flags}{1 (optional)}%
10466 \else\mw@setflags{#2}%
10467 \fi}

\mw@secdef 10469 \def\mw@secdef#1#2#3#4{%
% #1 the heading name,
% #2 the command distincter,
% #3 the meaning,
% #4 the number of argument to error message.

10476 \gmu@ifempty{#3}
10477 {\mw@secundef{#1#2}{#4}}
10478 {\@namedef{#1#2}{#3}}}

\mw@secundef 10480 \def\mw@secundef#1#2{%
10481 \gmu@ifundefined{#1}{%
10482 \ClassError{mwcls/gm}{%
10483 \command \bslash#1 undefined \MessageBreak
10484 after \bslash SetSectionFormatting!!!\MessageBreak}{%
10485 Provide the #2 argument of \bslash SetSectionFormatting.}}}}

```

First argument is a sectioning command (wo. the backslash) and second the stuff to be added at the beginning of the heading declarations.

```

\addtoheading 10490 \def\addtoheading#1#2{%
10491 \n@melet{gmu@addtoheading@resa}{#1@head}%

```



```

10492 \edef\gmu@addtoheading@resa{\unexpanded{#2}\@xa\unexpanded{%
      \gmu@addtoheading@resa}}%
10493 \n@melet{#1@head}{\gmu@addtoheading@resa}%
10494 }
10496 }% of \@ifnotmw's else.

```

### Negative \addvspace

When two sectioning commands appear one after another (we may assume that this occurs only when a lower section appears immediately after higher), we prefer to put the *smaller* vertical space not the larger, that is, the preskip of the lower sectioning not the postskip of the higher.

For that purpose we modify the very inner macros of MWCLS to introduce a check whether the previous vertical space equals the postskip of the section one level higher.

```

10508 \@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 10513 \def\@afterheading{%
10514   \@nobreaktrue
10515   \xdef\gmu@prevsec{\mw@HeadingType}% added now
10516   \everypar{%
10517     \grelaxen\gmu@prevsec% added now. All the rest is original LATEX.
10518     \if@nobreak
10519     \@nobreakfalse
10520     \clubpenalty \@M
10521     \if@afterindent \else
10522     {\setbox\z@\lastbox}%
10523     \fi
10524     \else
10525     \clubpenalty \@clubpenalty
10526     \everypar{}%
10527     \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 10534 \def\gmu@checkaftersec{%
10535   \gmu@ifundefined{gmu@prevsec}{}{%
10536     \ifgmu@postsec% an additional switch that is true by default but may be turned
      into an \ifdim in special cases, see line 10572.
10539     {\@xa\mw@getflags\@xa{\gmu@prevsec}%
10540     \glet\gmu@checkaftersec@resa\mw@HeadingBreakAfter}%
10541     \if\mw@HeadingBreakBefore\def\gmu@checkaftersec@resa{11}\fi% if the cur-
      rent heading inserts page break before itself, all the play with vskips is
      irrelevant.
10544     \if\gmu@checkaftersec@resa\else
10545     \penalty10000\relax
10546     \skip\z@=\csname\gmu@prevsec @postskip\endcsname\relax
10547     \skip\tw@=\csname\mw@HeadingType @preskip\endcsname\relax

```

```

10548 \gmu@ifundefined{\mw@HeadingType @twoheadskip}{%
10549 \ifdim\skip\z@>\skip\tw@
10550 \vskip-\skip\z@% we strip off the post-skip of previous header if it's bigger
        than current pre-skip
10552 \else
10553 \vskip-\skip\tw@% we strip off the current pre-skip otherwise
10554 \fi}{% But if the two-header-skip is defined, we put it
10556 \penalty10000
10557 \vskip-\skip\z@
10558 \penalty10000
10559 \vskip-\skip\tw@
10560 \penalty10000
10561 \vskip\csname\mw@HeadingType @twoheadskip\endcsname
10562 \relax}%
10563 \penalty10000
10564 \hrule height\z@\relax% to hide the last (un)skip before
        subsequent \addvspaces.
10566 \penalty10000
10567 \fi
10568 \fi
10569 }% of \gmu@ifundefined{gmu@prevsec} 'else'.
10570 }% of \def\gmu@checkaftersec.

10572 \def\ParanoidPostsec{% this version of \ifgmu@postsec is intended for the special
        case of sections may contain no normal text, as while gmdocing.
10575 \def\ifgmu@postsec{% note this macro expands to an open \if.
10576 \skip\z@=\csname\gmu@prevsec @postskip\endcsname\relax
10577 \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 gm-
        docing).
10581 }}

10583 \let\ifgmu@postsec\iftrue

10585 \def\gmu@getaddvs#1\addvspace#2\gmu@getaddvs{%
10586 \toks\z@={#1}%
10587 \toks\tw@={#2}}

        And the modification of the inner macros at last:

10590 \def\gmu@setheading#1{%
10591 \@xa\gmu@getaddvs#1\gmu@getaddvs
10592 \edef#1{%
10593 \the\toks\z@\@nx\gmu@checkaftersec
10594 \@nx\addvspace\the\toks\tw@}}

10596 \gmu@setheading\mw@normalheading
10597 \gmu@setheading\mw@runinheading

10599 \def\SetTwoheadSkip#1#2{\@namedef{#1@twoheadskip}{#2}}
10601 }% of \@ifnotmw's else.

```

### My setup of headings for mwcls

The setup of headings' skips was tested in 'real' typesetting, for money that is. The skips are designed for 11/13pt 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 of that time, “Wiedza Powszechna” Editions.

```

10613 \@ifnotmw}{% We define this declaration only when in mwcls.
\WPheadings 10614 \DeclareCommand\WPheadings{T{\chapter}}{%
10615   \SetSectionFormatting[breakbefore,wholewidth]
10616     {part}{\z@\@plus1fill}{\z@\@plus3fill}%
10618   \gmuIfValueF{#1}{%
10619     \gmu@ifundefined{chapter}{}{%
10620       \SetSectionFormatting[breakbefore,wholewidth]
10621         {chapter}
10622         {66\p@}{67\p@} for Adventor/Schola 0,95.
10623         {\FormatHangHeading{\LARGE}}
10624         {27\p@\@plus0,2\p@\@minus1\p@}%
10625     }%
10626 }% of unless #1

10628   \SetTwoheadSkip{section}{27\p@\@plus0,5\p@}%
10629   \SetSectionFormatting{section}
10630     {24\p@\@plus0,5\p@\@minus5\p@}%
10631     {\FormatHangHeading {\Large}}
10632     {10\p@\@plus0,5\p@}% ed. Krajewska of “Wiedza Powszechna”, as we under-
      stand her, wants the skip between a heading and text to be rigid.

10636   \SetTwoheadSkip{subsection}{11\p@\@plus0,5\p@\@minus1\p@}%
10637   \SetSectionFormatting{subsection}
10638     {19\p@\@plus0,4\p@\@minus6\p@}
10639     {\FormatHangHeading {\large}}% 12/14 pt
10640     {6\p@\@plus0,3\p@}% after-skip 6 pt due to p.12, not to squeeze the before-
      skip too much.

10643   \SetTwoheadSkip{subsubsection}{10\p@\@plus1,75\p@\@minus1\p@}%
10644   \SetSectionFormatting{subsubsection}
10645     {10\p@\@plus0,2\p@\@minus1\p@}
10646     {\FormatHangHeading {\normalsize}}
10647     {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.

10651   \SetSectionFormatting[runin]{paragraph}
10652     {7\p@\@plus0,15\p@\@minus1\p@}
10653     {\FormatRunInHeading{\normalsize}}
10654     {2\p@}%
10656   \SetSectionFormatting[runin]{subparagraph}
10657     {4\p@\@plus1\p@\@minus0,5\p@}
10658     {\FormatRunInHeading{\normalsize}}
10659     {\z@}%
10660 }% of \WPheadings
10661 }% 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.)

```

10702 \@ifnotmw{% we are not in mwcls and want to handle mwcls-like sectionings i.e., those
        written with two optionals.
10705   \def\gmu@secini{gm@la}%
10706   \Store@Macros{%
10707     \partmark \chaptermark \sectionmark \subsectionmark
10708     \subsubsectionmark \paragraphmark}%
\gmu@secxx 10710   \def\gmu@secxx#1#2[#3]#4{%
10711     \ifx\gmu@secstar\@empty
        a tiny little trick to allow a special version of the heading just to the running head.
10714     \@namedef{#1mark}##1{% we redefine \(\sec)mark to gobble its argument and
        to launch the stored true marking command on the appropriate argument.
10717     \storedcsname{#1mark}{#2}%
10718     \Restore@MacroSt{#1mark}% after we've done what we wanted we restore
        original \#1mark.
10720     }%
10721     \def\gmu@secstar{[#3]}% if \gmu@secstar is empty, which means the sec-
        tioning command was written starless, we pass the 'true' sectioning com-
        mand #3 as the optional argument. Otherwise the sectioning command
        was written with star so the 'true' s.c. takes no optional.
10726     \fi
10727     \@xa\@xa\csname\gmu@secini#1\endcsname
10728     \gmu@secstar{#4}}%
10730 }{% 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.
10733   \def\gmu@secini{gm@mw}%
10735   \let\gmu@secmarkh\@gobble% in mwcls there's no need to make tricks for special
        version to running headings.
\gmu@secxx 10738   \def\gmu@secxx#1#2[#3]#4{%
10739     \@xa\@xa\csname\gmu@secini#1\endcsname
10740     \gmu@secstar[#2][#3]{#4}}%
10741 }
10743 \def\gmu@sec#1{\@dblarg{\gmu@secx{#1}}}
```

```

10744 \def\gmu@secx#1[#2]{%
10745   \@ifnextchar[{\gmu@secxx{#1}{#2}}{\gmu@secxx{#1}{#2}[#2]}}% if there's only
      one optional, we double it not the mandatory argument.

10749 \def\gmu@straightensec#1{% the parameter is for the command's name.
10750   \gmu@ifundefined{#1}{}{% we don't change the ontological status of the command
      because someone may test it.
10752   \n@melet{\gmu@secini#1}{#1}%
10753   \@namedef{#1}{%
10754     \gmu@ifstar{\def\gmu@secstar{*}\gmu@sec{#1}}{%
10755       \def\gmu@secstar{\gmu@sec{#1}}}}%
10756 }%

10758 \let\do\gmu@straightensec
10759 \do{part}\do{chapter}\do{section}\do{subsection}\do{subsubsection}
10760 \@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?

10765 </mw>
      enumerate* and itemize* moved to gmbase.

```

## The gmtypos package—some bits of typography (in a non-systematic way)

Mostly according to Polish 20th Century typesetting standards.

```

10776 <utils>   \gmu@PackOptionX{typos}
10777 <*typos>
10778 \RequirePackage{gmcommand, gmnotonlypream}

10780 \unless\ifcsname ifgmu@quiet\endcsname
10781   \@xa\newif\csname ifgmu@quiet\endcsname
10782 \fi

quiet 10785 \DeclareOption{quiet}{\gmu@quiettrue}
10787 \ProcessOptions

```

## Brave New World of X<sub>3</sub>T<sub>E</sub>X

\@ifXeTeX moved to **gmbase** (2010/04/10)

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

```

10797 \AtBeginDocument{%
10798   \@ifpackageloaded{fontspec}{%
10799     \pdef\udigits{%
10800       \addfontfeature{Numbers=Uppercase}}%
10801   }{%
10802     \emptify\udigits}%
10803 }

```

## Fractions

```
10808 \def\Xedekfracc{\gmu@ifstar\gmu@xedekfraccstar\gmu@xedekfraccplain}
```

(plain) The starless version turns the font feature `frac` on.

(\*) But nor my modification of Minion Pro neither T<sub>E</sub>X 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.

```
10823 \def\gmu@xedekfraccstar{%
10824   \def\gmu@xfraccdef##1##2{%
10825     \iffontchar\font ##2
10826     \@namedef{gmu@xfracc##1}{\char##2 }%
10827     \else
10828     \nmelet{gmu@xfracc##1}{relax}%
10829     \fi}%
10831   \def\gmu@dekfracc##1/##2{%
10832     {\addfontfeature{%
10833       VerticalPosition=Numerator}##1}\gmu@numeratorkern
10834     \char"2044 \gmu@denominatorkern
10835     {\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.

```
10838   \gmu@xfraccdef{1/4}{ "BC}%
10839   \gmu@xfraccdef{1/2}{ "BD}%
10840   \gmu@xfraccdef{3/4}{ "BE}%
10841   \gmu@xfraccdef{1/3}{ "2153}%
10842   \gmu@xfraccdef{2/3}{ "2154}%
10843   \gmu@xfraccdef{1/5}{ "2155}%
10844   \gmu@xfraccdef{2/5}{ "2156}%
10845   \gmu@xfraccdef{3/5}{ "2157}%
10846   \gmu@xfraccdef{4/5}{ "2158}%
10847   \gmu@xfraccdef{1/6}{ "2159}%
10848   \gmu@xfraccdef{5/6}{ "215A}%
10849   \gmu@xfraccdef{1/8}{ "215B}%
10850   \gmu@xfraccdef{3/8}{ "215C}%
10851   \gmu@xfraccdef{5/8}{ "215D}%
10852   \gmu@xfraccdef{7/8}{ "215E}%
10853   \pdef\dekfracc@args##1/##2{%
10854     \gmu@ifundefined{gmu@xfracc\detokenize{##1/##2}}{%
10855       \gmu@dekfracc{##1}/{##2}}{%
10856       \csname gmu@xfracc\detokenize{##1/##2}\endcsname}%
10857     \if@gmu@mmhbox\egroup\fi
10858   }% of \dekfracc@args.
10859   \gmu@ifstar{\let\gmu@dekfracc\gmu@dekfraccsimple}{}%
10860 }

10862 \def\gmu@xedekfraccplain{% 'else' of the main \gmu@ifstar
10863   \pdef\dekfracc@args##1/##2{%
10864     \ifmmode\hbox\fi}%

```

```

10865     \addfontfeature{Fractions=On}%
10866     ##1/##2}%
10867     \if@gmu@mmhbox\egroup\fi
10868   }% of \dekfracc@args
10869   }

\if@gmu@mmhbox 10871 \newif\if@gmu@mmhbox% we'll use this switch for \dekfracc and also for \thous
                (hacky thousand separator).

10874 \pdef\dekfracc{%
10876   \ifmmode\hbox\bgroup\@gmu@mmhboxtrue\fi
10877   \dekfracc@args}

10880 \def\gmu@numeratorkern{\kern-.055em\relax}
10881 \def\gmu@denominatorkern{\kern-.05em\relax}

```

What have we just done? We defined two versions of the `\Xefrac` declaration. The starred version is intended to make use only of the built-in fractions such as  $1/2$  or  $7/8$ . To achieve that, a handful of macros is defined that expand to the Unicodes of built-in fractions and `\dekfracc` command is defined to use them.

The unstarred version makes use of the Fraction font feature and therefore is much simpler.

Note that in the first argument of `\gmu@ifstar` we wrote 8 (eight) #s to get the correct definition and in the second argument ‘only’ 4. (The  $\text{\LaTeX} 2_{\epsilon}$  Source claims that that is changed in the ‘new implementation’ of `\gmu@ifstar` so maybe it’s subject to change.)

A simpler version of `\dekfracc` is provided in line 12147.

```

10902 \pdef\textsuperscript@@#1{%
10903   \@textsuperscript{\selectfont#1}}%
10904 \def\@textsuperscript#1{%
10905   {\m@th\ensuremath{\mbox{\fontsize\sf@size\z@#1}}}}
10907 \let\textsuperscript@@\textsuperscript

10909 \def\GMtextsuperscript{%
10910   \@ifXeTeX{%
\textsuperscript 10911   \DeclareCommand\textsuperscript{sm}{%
10912     \gmuIfValueTF{##1}{\textsuperscript@@{##2}}%
10913     {%
10914       \begingroup
10915         \addfontfeature{VerticalPosition=Numerator}##2%
10916         \endgroup}}%
10917   }{\truetextsuperscript}}

10919 \def\truetextsuperscript{%
10920   \let\textsuperscript\textsuperscript@@
10921 }

```

### Settings for mathematics in main font

`\gmath` I used these terrible macros while typesetting E. Szarzyński’s *Letters* in 2008. The `\gmath` declaration introduces math-active digits and binary operators and redefines Greek letters and parentheses, the `\garamath` declaration redefines the quantifiers and is more Garamond Premier Pro-specific.

`\gmath` So, when you set default fonts (in the preamble), put `\gmath` to set what possible from them to math. This sets the `normal` math version. If you want to use another set of fonts

elsewhere in your document and have according math for them, set them ‘for a while’ in the preamble and in their scope declare `\gmath[<version>]`. Then put `\mathversion{normal}` right after `\begin{document}` and `\gmath[<version>]` where you want those other fonts.

`\gmath` takes second optional argument, in parentheses, which should be (sth. that expands to) a `\fontspec` font selecting command. A font selected by this command will be declared and used when some char in basic font is missing and only else the default math font will be left for such a char.

So,

```
\gmath[<version>] (<rescue font (fontspec-ification)>)
```

Note that `\gmath` without first optional argument has always to come first because otherwise it overwrite settings of your math version.

```
10952 \def\gmu@getfontstring{%
10953   \xdef\gmu@fontstring{%
10954     \gmu@fontstring@}}
10956 \def\gmu@fontstring@{%
10957   \@xa\@xa\@xa\gmu@fontstring@@\@xa\meaning\the\font\@nil}
10960 \def\gmu@fontstring@@#1"#2"#3\@nil{"#2"}
10962 \def\gmu@getfontscale#1Scale#2=#3,{%
10963   \ifx\gmu@getfontscale#3\else
10964     \def\gmu@tempa{MatchLowercase}%
10965     \def\gmu@tempb{#3}%
10966     \ifx\gmu@tempa\gmu@tempb
10967       \gmu@calc@scale{5}%
10968       \@xa\@firstoftwo
10969     \else
10970       \def\gmu@tempa{MatchUppercase}%
10971       \ifx\gmu@tempa\gmu@tempb
10972         \gmu@calc@scale{8}%
10973         \afterfifi\@firstoftwo
10974       \else\afterfifi\@secondoftwo
10975     \fi
10976   \fi
10977   {\xdef\gmu@fontscalebr{[\gmu@fontscale] }}%
10978   {\xdef\gmu@fontscalebr{[#3] }}%
10979   \afterfi\gmu@getfontscale
10980 \fi
10981 }
10984 \def\gmu@getfontdata#1{%
10985   \global\emptify\gmu@fontscalebr
10986   \begingroup
10987   #1%
10988   \@xa\@xa\@xa\gmu@getfontscale
10989   \csname zf@family@options\f@family\endcsname
10990   ,Scale=\gmu@getfontscale,%
10991   \gmu@getfontstring
10992   \xdef\gmu@theskewchar{\the\skewchar\font}%
10993   \endgroup}
10996 \def\gmu@stripchar#1{""}
\gmath@getfamnum 10998 \DeclareCommand\gmath@getfamnum{C{\gmath@fam}}{%
```



```

10999 \edef\gmath@famnum{\@xa\gmu@stripchar\meaning#1}%
11000 }

\XeTeXmathcode<char slot> [=]<type> <family> <char slot>

\gmathFams 11004 \DeclareCommand\gmathFams{o % the name of math version
11005 C{\NoValue} % 'rescue' font. Will be accessible via \symgmathRoman<version>
(math family) and \gmath@fontt<version> (font).
11008 }{%
11009 \gmuIfValueT{#1}{%
11010 \DeclareMathVersion{#1}% this sets the defaults so no need to define them ex-
plicitly
11012 }% of if #1 given

11014 \gmu@getfontdata{\rmfamily\itshape}%
11016 \edef\gmu@tempa{%
11017 \gmuIfValueTF{#1}{\@nx\SetSymbolFont{letters}{#1}}%
11018 {\@nx\DeclareSymbolFont{letters}}%
11019 {\encodingdefault}{gmathit\gmuPutIfValue{#1}}{m}{it}%
11020 \gmuIfValueT{#1}{%
11021 \@nx\DeclareSymbolFont{letters#1}%
11022 {\encodingdefault}{gmathit\gmuPutIfValue{#1}}{m}{it}%
11023 }%
11024 \@nx\DeclareFontFamily{\encodingdefault}{gmathit\gmuPutIfValue{%
#1}}{%
11025 \skewchar\font\gmu@theskewchar\space}%
11026 \@nx\DeclareFontShape{\encodingdefault}{gmathit\gmuPutIfValue{%
#1}}{m}{it}{%
11027 <-> \gmu@fontscalebr \gmu@fontstring}}{%
11028 \gmuIfValueT{#1}{%
11029 \@nx\SetMathAlphabet\@nx\mathit{#1}{\encodingdefault}{%
gmathit#1}{m}{it}}%
11030 }\gmu@tempa
11032 \gmuIfValueT{#2}{%
11033 \gmu@getfontdata{#2\gmu@ifstored{\upshape}{\storedcsname{
upshape}}}{\upshape}}%
11034 \edef\gmu@tempa{%
11035 \@nx\DeclareSymbolFont{gmathRoman\gmuPutIfValue{#1}}%
11036 {\encodingdefault}{gmathRm\gmuPutIfValue{#1}}{m}{n}%
11037 \@nx\DeclareFontFamily{\encodingdefault}{gmathRm\gmuPutIfValue{%
#1}}{%
11038 \skewchar\font\gmu@theskewchar\space}%
11039 \@nx\DeclareFontShape{\encodingdefault}{gmathRm\gmuPutIfValue{%
#1}}{m}{n}{%
11040 <-> \gmu@fontscalebr \gmu@fontstring}}{%
11041 }\gmu@tempa
11042 \@xa\font\csname gmath@fontt\gmuPutIfValue{#1}\endcsname
11043 =\gmu@fontstring\relax
11045 \gmu@getfontdata{#2\gmu@ifstored{\itshape}{\storedcsname{
itshape}}}{\itshape}}%
11046 \edef\gmu@tempa{%
11047 \@nx\DeclareSymbolFont{gmathItalic\gmuPutIfValue{#1}}%
11048 {\encodingdefault}{gmathIt\gmuPutIfValue{#1}}{m}{n}%

```

```

11049     \@nx\DeclareFontFamily{\encodingdefault}{gmathIt\gmuPutIfValue{%
          #1}}{%
11050         \skewchar\font\gmu@theskewchar\space}%
11051     \@nx\DeclareFontShape{\encodingdefault}{gmathIt\gmuPutIfValue{%
          #1}}{m}{n}{%
11052         <-> \gmu@fontscalebr \gmu@fontstring}{}%
11053     }\gmu@tempa
11054 }% of if #2 given

11056 \gmu@getfontdata{\rmfamily\upshape}%
11057 \edef\gmu@tempa{%
11058     \gmuIfValueTF{#1}{\@nx\SetSymbolFont{gmathroman}{#1}}%
11059     {\@nx\DeclareSymbolFont{gmathroman}}%
11060     {\encodingdefault}{gmathrm\gmuPutIfValue{#1}}{m}{n}%
11061     \@nx\DeclareFontFamily{\encodingdefault}{gmathrm\gmuPutIfValue{%
          #1}}{%
11062         \skewchar\font\gmu@theskewchar\space}%
11063     \@nx\DeclareFontShape{\encodingdefault}{gmathrm\gmuPutIfValue{%
          #1}}{m}{n}{%
11064         <-> \gmu@fontscalebr \gmu@fontstring}{}%
11065     \gmuIfValueT{#1}{%
11066         \@nx\SetMathAlphabet\@nx\mathrm{#1}{\encodingdefault}{%
          gmathrm#1}{m}{n}}%
11067     }\gmu@tempa
11068     \@xa\font\csname gmath@font\gmuPutIfValue{#1}\endcsname
11069     =\gmu@fontstring\relax
11070     \gmathfamshook
11071 }

11073 \emptify\gmathfamshook

\gmathbase 11075 \DeclareCommand\gmathbase{oC{\NoValue}}{%
11076     \gmathFams[#1](#2)%
\gmath@do 11077 \DeclareCommand\gmath@do{%
11078     m % (1) the character or CS to be declared,
11079     o % (2) the Unicode to be assigned,
11080     m % (3) math type (CS like \mathord etc.)
11081     C{\gmath@fam} % font family
11082 }{%
11084     \gmath@getfamnum(##4)%
11085     \gmuIfValueTF{##2}{%
11086         \edef\gmu@tempa{%
11087             = \mathchar@type##3\space
11088             \gmath@famnum\space
11089             "##2\relax}%
11090         \if\relax\@nx##1%
11091             \gmu@ifstored{##1}{\Store@Macro##1}%
11092             \edef\gmu@tempa{%
11093                 \XeTeXmathchardef \@nx##1\gmu@tempa}%
11094         \else
11095             \edef\gmu@tempa{%
11096                 \XeTeXmathcode `##1 \gmu@tempa}
11097         \fi%
11098     }%

```

```

11099     {% no value of ##2
11100     \edef\gmu@tempa{%
11101     \XeTeXmathcode `##1 =
11102     \mathchar@type##3\space
11103     \gmath@famnum\space
11104     `##1\relax}%
11105     }%
11106     \gmu@tempa
11107 }% of \gmath@do

\gmath@doif 11109 \DeclareCommand\gmath@doif{%
11110     m      % (1) the Unicode hex of char enquired,
11111     m      % (2) the char or CS to be declared,
11112     m      % (3) math type CS(\mathord etc.),
11113     o      % (4) second-choice Unicode (taken if first-choice is absent),
11114     o      % (5) third-choice Unicode (as above if second-choice is absent from
           font).
11115     B{\gmath@fam} % (6) never used in this package. Why?
11116     C{\NoValue}
11117 }{%
11118     \gmu@storeifnotyet{##2}%
11119     \@xa\let\@xa\gmath@ft\csname
11120     gmath@font%
11121     \ifx\gmath@fam##6\else t\fi
11122     \gmath@version
11123     \endcsname
11124     \iffontchar\gmath@ft"##1 \gmath@do##2[##1]##3(##6)%
11125     \else
11126     \gmuIfValueTF{##4}{%
11127     \iffontchar\gmath@ft"##4 \gmath@do##2[##4]##3(##6)%
11128     \else
11129     \gmuIfValueTF{##5}{%
11130     \iffontchar\gmath@ft"##5 \gmath@do##2[##5]##3(##6)%
11131     \else
11132     \gmath@restore{##1}{##2}{##3}{##4}{##5}{##7}%
11133     \fi}%
11134     {\gmath@restore{##1}{##2}{##3}{##4}{##5}{##7}}%
11135     \fi}%
11136     {\gmath@restore{##1}{##2}{##3}{##4}{##5}{##7}}%
11137     \fi
11138 }% of \gmath@doif In the command above we try to define math char or a CS
           in the family given as ##6. If there're no respective chars, we try the same
           with the family given (as a word) in ##7.
11142 \def\gmath@restore##1##2##3##4##5##6{%
11143     \gmuIfValueT{##6}%
11144     {\ifcsname ##6\endcsname
11145     \edef\gmu@tempa{%
11146     \unexpanded{\gmath@doif{##1}{##2}{##3}[##4][##5]}%
11147     {\@xanxcs{##6}}\relax
11148     }\gmu@tempa
11149     \@xa\@gobbletwo % if family ##6 is defined, we gobble the other branch
11151     \fi
11152     }%

```

```

11153     \firstofone
11154     {\if\relax\@nx##2%
11155         \Restore@Macro##2%
11156     \fi
11157     }%
11158 }%
11160 \iffalse % doesn't work in a non-math font.
\gmath@delc 11161 \DeclareCommand\gmath@delc{mo}{%
                % #1 the char or CS to be declared,
                % [#2] the Unicode (if not the same as the char).
11167 \gmath@getfamnum
11168 \gmuIfValueTF{##2}{%
11169     \edef\gmu@tempa{%
11170         = \gmath@famnum\space "##2\relax}%
11171     \edef\gmu@tempa{%
11172         \XeTeXdelcode `##1 \gmu@tempa}
11173     }%
11174     {%
11175     \edef\gmu@tempa{%
11176         \XeTeXdelcode `##1 =
11177         \gmath@famnum\space
11178         `##1\relax}%
11179     }%
11180     \gmu@tempa
11181 }% of \gmath@delc
11183 \def\gmath@delcif##1##2{%
                % #1 the Unicode enquired,
                % #2 the char to be delcode-declared
11189     \iffontchar\gmath@font"##1 \gmath@delc##2[##1]\fi}
11190 \fi% of iffalse
11192 \def\gmath@delimif##1##2##3{%
                % #1 the Unicode enquired,
                % #2 the CS defined as \XeTeXdelimiter,
                % #3 the math type CS (probably \mathopen or \mathclose).
11199     \iffontchar\gmath@font"##1
11200     \gmath@getfamnum
11201     \protected\edef##2{\@nx\ensuremath{%
11202         \XeTeXdelimiter \mathchar@type##3\space
11203         \gmath@famnum\space "##1\relax}}%
11204     \fi}% of \gmath@delimif.
11206 \pdef\rmopname##1##2##3{%
11207     \mathop {##1\kern \z@ \mathrm{##3}}\csname n##2limits@\endcsname
11208 }%
\gmu@dogmathbase 11210 \DeclareCommand\gmu@dogmathbase{oC{\NoValue}}{%
11211     \Restore@Macro\mathchar@type%
11213     \gmuIfValueT{##1}{\mathversion{##1}}%
11215     \edef\gmath@version{\gmuPutIfValue{##1}}%
11217     \@xa\let\@xa\gmath@fam\csname symgmathroman%
11218     \endcsname
11219     \edef\gmath@famm{symgmathRoman\gmath@version}% as you see, this is not
                a font family (number) but a macro containing the name: of the secondary

```

(‘rescue’) family.

```
11223 \typeout{@@@ gmutils.sty: taking some math chars from the font^^J
      \gmu@fontstring@}%
11224 \gmath@do+\mathbin
11225 \gmath@doif{2212}-\mathbin[2013] (\gmath@famm) % minus sign if present or
      else en dash
11226 \gmath@do=\mathrel
11227 \gmath@do0\mathord
11228 \gmath@do1\mathord
11229 \gmath@do2\mathord
11230 \gmath@do3\mathord
11231 \gmath@do4\mathord
11232 \gmath@do5\mathord
11233 \gmath@do6\mathord
11234 \gmath@do7\mathord
11235 \gmath@do8\mathord
11236 \gmath@do9\mathord
11238 \gmath@doif{2264}\le\mathrel (\gmath@famm) %
11239 \let\leq\le
11240 \let\leeng\le
11241 \gmath@doif{2265}\ge\mathrel (\gmath@famm) %
11242 \let\geq\ge
11243 \let\geeng\ge
11244 \gmath@doif{2A7D}\xleq\mathrel (\gmath@famm) %
11245 \gmath@doif{2A7E}\xgeq\mathrel (\gmath@famm) %
11246 \@ifpackageloaded{polski}{%
11247   \ifdefined\xleq
11248   \gmu@storeifnotyet\leq
11249   \let\leq=\xleq
11250   \let\le=\leq
11251   \fi
11252   \ifdefined\xgeq
11253   \gmu@storeifnotyet\geq
11254   \let\geq=\xgeq
11255   \let\ge=\geq
11256   \fi}{}%
11258 \gmath@do.\mathpunct
11259 \gmath@do,\mathpunct
11260 \gmath@do;\mathpunct
11261 \gmath@do...\mathpunct
11262 \gmath@do(\mathopen
11263 \gmath@do)\mathclose
11264 \gmath@do[\mathopen
11265 \gmath@do]\mathclose
11267 \gmath@doif{00D7}\times\mathbin (\gmath@famm) %
11268 \gmath@do:\mathrel
11269 \gmath@doif{00B7}\cdot\mathbin (\gmath@famm) %
11270 \gmath@doif{22C6}\*\mathbin (\gmath@famm) % low star
11271 \gmath@doif{2300}\varnothing\mathord (\gmath@famm) %
11272 \gmath@doif{221E}\infty\mathord (\gmath@famm) %
11273 \gmath@doif{2248}\approx\mathrel (\gmath@famm) %
11274 \gmath@doif{2260}\neq\mathrel (\gmath@famm) %
```

```

11275 \let\ne\neg
11276 \gmath@doif{00AC}\neg\mathbin(\gmath@famm)%
11277 \gmath@doif{00AC}\nego\mathord(\gmath@famm)%
11278 \gmath@do/\mathop
11279 \gmath@do<\mathrel
11280 \gmath@do>\mathrel
11281 \gmath@doif{2329}\langle\mathopen(\gmath@famm)%
11282 \gmath@doif{232A}\rangle\mathclose(\gmath@famm)%
11283 \gmath@doif{2202}\partial\mathord(\gmath@famm)%
11284 \gmath@doif{00B1}\pm\mathbin(\gmath@famm)%
11285 \gmath@doif{007E}\sim\mathrel(\gmath@famm)%
11286 \gmath@doif{2190}\leftarrow\mathrel(\gmath@famm)%
11287 \gmath@doif{2192}\rightarrow\mathrel(\gmath@famm)%
11288 \gmath@doif{2194}\leftrightarrow\mathrel(\gmath@famm)% if not present,
% \gmathfurther will take care of it if left and right arrows are present.
11291 \gmath@doif{2191}\uparrow\mathrel(\gmath@famm)% it should be a delimiter
(declared with \gmath@delimif) but in a non-math font the delimiters
don't work (2008/11/19) and I don't think I'll ever need up- and down-
arrows as delimiters.
11295 \gmath@doif{2193}\downarrow\mathrel(\gmath@famm)%
11297 \gmath@doif{2208}\in\mathrel[03F5][0454](\gmath@famm)%

```

As a fan of modal logics I allow redefinition of `\lozenge` and `\square` iff both are in the font. I don't accept the 'ballot box' U+2610.

```

11301 \if\iffontchar\gmath@font"25CA 0\else 1\fi
11302 \iffontchar\gmath@font"25FB 0\else\iffontchar\gmath@font"25A1
% \else 2\fi\fi
11303 \gmath@do\lozenge[25CA]\mathord
11304 \gmath@doif{25FB}\square\mathord[25A1](\gmath@famm)% 'medium white
square (modal operator)' of just 'white square'.
11306 \fi
11307 \gmath@doif{EB08}\bigcircle\mathbin(\gmath@famm)%
11308 \gmath@doif{2227}\wedge\mathbin(\gmath@famm)%
11309 \gmath@doif{2228}\vee\mathbin(\gmath@famm)%
11311 \gmath@doif{0393}\Gamma\mathalpha(\gmath@famm)%
11312 \gmath@doif{0394}\Delta\mathalpha(\gmath@famm)%
11313 \gmath@doif{0398}\Theta\mathalpha(\gmath@famm)%
11314 \gmath@doif{039B}\Lambda\mathalpha(\gmath@famm)%
11315 \gmath@doif{039E}\Xi\mathalpha(\gmath@famm)%
11316 \gmath@doif{03A3}\Sigma\mathalpha(\gmath@famm)%
11317 \gmath@doif{03A5}\Upsilon\mathalpha(\gmath@famm)%
11318 \gmath@doif{03A6}\Phi\mathalpha(\gmath@famm)%
11319 \gmath@doif{03A8}\Psi\mathalpha(\gmath@famm)%
11320 \gmath@doif{03A9}\Omega\mathalpha(\gmath@famm)%
11322 \@xa\let\@xa\gmath@fam\csname symletters\gmath@version%
11323 \endcsname
11324 \edef\gmath@famm{symgmathItalic\gmath@version}%
11326 \gmath@doif{03B1}\alpha\mathalpha(\gmath@famm)%
11327 \gmath@doif{03B2}\beta\mathalpha(\gmath@famm)%
11328 \gmath@doif{03B3}\gamma\mathalpha(\gmath@famm)%
11329 \gmath@doif{03B4}\delta\mathalpha(\gmath@famm)%
11330 \gmath@doif{03F5}\epsilon\mathalpha(\gmath@famm)%
11331 \gmath@doif{03B5}\varepsilon\mathalpha(\gmath@famm)%

```

```

11332 \gmath@doif{03B6}\zeta\mathalpha(\gmath@famm)%
11333 \gmath@doif{03B7}\eta\mathalpha(\gmath@famm)%
11334 \gmath@doif{03B8}\theta\mathalpha(\gmath@famm)%
11335 \gmath@doif{03D1}\vartheta\mathalpha(\gmath@famm)%
11336 \gmath@doif{03B9}\iota\mathalpha(\gmath@famm)%
11337 \gmath@doif{03BA}\kappa\mathalpha(\gmath@famm)%
11338 \gmath@doif{03BB}\lambda\mathalpha(\gmath@famm)%
11339 \gmath@doif{03BC}\mu\mathalpha(\gmath@famm)%
11340 \gmath@doif{03BD}\nu\mathalpha(\gmath@famm)%
11341 \gmath@doif{03BE}\xi\mathalpha(\gmath@famm)%
11342 \gmath@doif{03C0}\pi\mathalpha(\gmath@famm)%
11343 \gmath@doif{03A0}\Pi\mathalpha(\gmath@famm)%
11344 \gmath@doif{03C1}\rho\mathalpha(\gmath@famm)%
11345 \gmath@doif{03C3}\sigma\mathalpha(\gmath@famm)%
11346 \gmath@doif{03DA}\varsigma\mathalpha(\gmath@famm)% 03C2?
11347 \gmath@doif{03C4}\tau\mathalpha(\gmath@famm)%
11348 \gmath@doif{03C5}\upsilon\mathalpha(\gmath@famm)%
11349 \gmath@doif{03D5}\phi\mathalpha(\gmath@famm)%
11350 \gmath@doif{03C7}\chi\mathalpha(\gmath@famm)%
11351 \gmath@doif{03C8}\psi\mathalpha(\gmath@famm)%
11352 \gmath@doif{03C9}\omega\mathalpha(\gmath@famm)%
11354 \if 1 1%
11355 \iffontchar\gmath@font"221A
11356 \fontdimen61\gmath@font=1pt
11357 \edef\sqrtsign{%
11358 \XeTeXradical \@xa\gmu@stripchar\meaning\symgmathroman\space
"221A\relax}%
11359 \fi
11360 \fi% of if 1 1.
11361 \def\max{\rmopname \relax m{max}}%
11362 \def\min{\rmopname \relax m{min}}%
11363 \def\lim{\rmopname \relax m{lim}}%
11364 \def\sin{\rmopname \relax o{sin}}%
11365 \def\cos{\rmopname \relax o{cos}}%
11366 \def\tg{\rmopname \relax o{tg}}%
11367 \def\ctg{\rmopname \relax o{ctg}}%
11368 \def\tan{\rmopname \relax o{tan}}%
11369 \def\ctan{\rmopname \relax o{ctan}}%
11370 }% of \gmu@dogmathbase
11371 \AtBeginDocument{\gmu@dogmathbase[#1](#2)%
11372 \let\gmathbase\gmu@dogmathbase
11373 }% of atbd
11374 \not@onlypreamble\gmathbase
11375 }% of \gmathbase

```

The `\gmatbase` declaration defines a couple of gmath defining commands and then launches them for the default font at begin document and becomes only that launching.

```
11381 \@onlypreamble\gmathbase
```

It works as follows: if `\gmathbase` occurs first time in a document inside `document` then an error error is raised. But if `\gmathbase` occurs first time in the preamble, then it removes itself from the only-preamble list and redefines itself to be only the inner macro of the former itself.

```

11388 \pdef\gmathfurther{%
11395   \def\do##1##2##3{\gmu@storeifnotyet##1%
11396     \def##1{%
11397       \mathop{\mathchoice{\hbox{%
11398         \rm
11399         \edef\gma@tempa{\the\fontdimen8\font}%
11400         \larger[3]%
11401         \lower\dimexpr(\fontdimen8\font-\gma@tempa)/2 %
11402         \hbox{##2}}}{\hbox{%
11403         \rm
11404         \edef\gma@tempa{\the\fontdimen8\font}%
11405         \larger[2]%
11406         \lower\dimexpr(\fontdimen8\font-\gma@tempa)/2 %
11407         \hbox{##2}}}}%
11408       {\mathrm{##2}}{\mathrm{##2}}##3}}%
11409   \iffontchar\gmath@font"2211   \do\sum{\char"2211}}{\fi%
11410   \do\forall{\gma@quantifierhook \rotatebox[origin=c]{180}{A}%
11411     \gmu@forallkerning
11412   }{\nolimits}%
11413   \def\gmu@forallkerning{\setbox0=\hbox{A}\setbox2=\hbox{%
11414     \scriptsize x}%
11415     \kern\dimexpr\ht2/3*2 -\wd0/2\relax}% to be able to redefine it when the
11416     big quantifier is Bauhaus-like.
11417   \do\exists{\rotatebox[origin=c]{180}{\gma@quantifierhook
11418     E}}{\nolimits%
11419   \def\do##1##2##3{\gmu@storeifnotyet##1%
11420     \def##1{##3{%
11421       \mathchoice{\hbox{\rm##2}}{\hbox{\rm##2}}%
11422       {\hbox{\rm\scriptsize##2}}{\hbox{\rm\tiny##2}}}}}%
11423   \unless\iffontchar\gmath@font"2227
11424     \do\vee{\rotatebox[origin=c]{90}{<}}\mathbin%
11425   \fi
11426   \unless\iffontchar\gmath@font"2228
11427     \do\wedge{\rotatebox[origin=c]{-90}{<}}\mathbin
11428   \fi
11429   \unless\iffontchar\gmath@font"2194
11430     \if\iffontchar\gmath@font"2190 0\else1\fi
11431     \iffontchar\gmath@font"2192 0\else2\fi
11432     \do\leftrightharpoon{\char"2190\kern-0,1em \char"2192}\mathrel
11433   \fi\fi
11434   \def\do##1##2##3{\gmu@storeifnotyet##1%
11435     \def##1####1{##2{\hbox{%
11436       \rm
11437       \setbox0=\hbox{####1}%
11438       \edef\gma@tempa{\the\ht0}%
11439       \edef\gma@tempb{\the\dp0}%
11440       ##3%
11441       \setbox0=\hbox{####1}%
11442       \lower\dimexpr(\ht0 + \dp0)/2-\dp0
11443       -((\gma@tempa+\gma@tempb)/2-\gma@tempb) %
11444       \box0}}}}%
11445   \do\bigl\mathopen\larger

```



```

11447 \do\bigr\mathclose\larger
11448 \do\Bigl\mathopen\largerr
11449 \do\Bigr\mathclose\largerr
11450 \do\biggl\mathopen{\larger[3]}%
11451 \do\biggr\mathclose{\larger[3]}%
11452 \do\Biggl\mathopen{\larger[4]}%
11453 \do\Biggr\mathclose{\larger[4]}%
11456 \addtotoks\everymath{%
11459 \def\do##1##2{\gmu@storeifnotyet##1%
11460 \def##1{\ifmmode##2{\mathchoice
11461 {\hbox{\rm\char`##1}}{\hbox{\rm\char`##1}}}%
11462 {\hbox{\rm\scriptsize\char`##1}}{\hbox{%
\rm\tiny\char`##1}}}%
11463 \else\char`##1\fi}}%
11464 \do\{\mathopen
11465 \do\}\mathclose
11467 \def\={\mathbin{=}}%
11468 \def\neqb{\mathbin{\neq}}%
11469 \let\neb\neqb
11470 \def\do##1{\gmu@storeifnotyet##1%
11471 \edef\gma@tempa{%
11472 \def\@xanxcs{\@xa\gobble\string##1r}{%
11473 \@nx\mathrel{\@nx##1}}}%
11474 \gma@tempa}%
11475 \do\vee \do\wedge \do\neg
11476 \def\fakern{\mkern-3mu}%
11477 \thickmuskip=8mu plus 4mu\relax
11479 \gma@gmathhook
11480 }% of \everymath.
11481 \everydisplay\everymath
11482 \ifdefined\Url
11483 \ampulexdef\Url{\let\do}\@makeoother
11484 {\everymath}\let\do\@makeoother}% I don't know why but the url package's
% \url typesets the argument inside a math which caused digits not to be
typewriter but Roman and lowercase.
11488 \fi% of \ifdefined\Url.
11489 }% of \def\gmathfurther.
11491 \Store@Macro\mathchar@type
\gmath 11493 \DeclareCommand\gmath{oC{\NoValue}}{%
11494 \gmathbase[#1](#2)%
11495 \gmathfurther
11496 \gmuIfValueT{#1}{\csname gmathhook#1\endcsname}% this allows adding version-
specific stuff (I first used this for Fell fonts rescued with Garamond Premier)
11499 }
11501 \pdef\gmathscripts{%
11502 \addtotoks\everymath{\catcode`\^=7\relax \catcode`\_ =8\relax }%
11503 \everydisplay\everymath}
11505 \pdef\gmathcats{%
11506 \addtotoks\everymath{\gmu@septify}%
11507 \everydisplay\everymath}
11509 \emptify\gma@quantifierhook

```

```

\quantifierhook 11510 \def\quantifierhook#1{%
\gma@quantifierhook 11511   \def\gma@quantifierhook{#1}}

11513   \emptify\gma@gmathhook
\gmathhook 11514   \def\gmathhook#1{\addtomacro\gma@gmathhook{#1}}

\gma@dollar 11517 \def\gma@dollar$#1${{\gmath$#1$}}%
\gma@bare 11518 \def\gma@bare#1{\gma@dollar$#1$}%
\gma@checkbracket 11519 \def\gma@checkbracket{\@ifnextchar\[%
11520   \gma@bracket\gma@bare}
\gma@bracket 11521 \def\gma@bracket\[#1\>{{\gmath\[#1\}}\@ifnextchar\par{{\noindent}}
\gma 11522 \def\gma{\@ifnextchar$%
11523   \gma@dollar\gma@checkbracket}

\garamath 11528 \DeclareCommand\garamath{%
11529   O{\rm}% the font command
11530 }{%

Before 2009/10/19 all the stuff was added to \everymath which didn't work.

11533   \quantifierhook{\addfontfeature{OpticalSize=800}}%
\gma@arrowdash 11535 \def\gma@arrowdash{%
11536   \setbox0=\hbox{\char"2192}\copy0\kern-0,6\wd0
11537   \bgcolor\rule[-\dp0]{0,6\wd0}{%
\dimexpr1,07\ht0+\dp0}\kern-0,6\wd0}}%
\gma@gmathhook 11539 \def\gma@gmathhook{%
11540   \def\do####1####2####3{\gmu@storeifnotyet####1%
11541   \def####1{####3}%
\mathchoice 11542   \mathchoice{\hbox{#1####2}}{\hbox{#1####2}}%
11543   {\hbox{#1\scriptsize####2}}{\hbox{#1\tiny####2}}}}%
11544   \do\mapsto{\rule[0,4ex]{0,1ex}{0,4ex}\kern-0,05em%
11545   \gma@arrowdash\kern-0,05em\char"2192}\mathrel
11546   \do\cup{\scshape u}\mathbin
11547   \do\varnothing{\setbox0=\hbox{\gma@quantifierhook\addfontfeature{%
Scale=1.272727}0}}%
11548   \setbox2=\hbox{\char"2044}%
11549   \copy0 \kern-0,5\wd0 \kern-0,5\wd2 \lower0,125\wd0 \copy2
11550   \kern0,5\wd0\kern-0,5\wd2}}% of \varnothing
11551   \do\leftarrow{\char"2190\kern-0,05em\gma@arrowdash}\mathrel
11552   \do\shortleftarrow{\char"2190}\mathrel
11553   \do\rightarrow{\gma@arrowdash\kern-0,05em\char"2192}\mathrel
11554   \do\shortrightarrow{\char"2192\relax}\mathrel
11555   \do\in{\gma@quantifierhook\char"0454}\mathbin
11556   \do\prec{\gma@quantifierhook
11557   \rotatebox[origin=c]{-90}}%
11558   \glyphname{u03A5.a}}\mathrel % added 2009/9/11
11559   }% of \gma@gmathhook
11560 }% of \garamath.

```

### Minion and Garamond Premier kerning and ligature fixes

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

```

\gmu@tempa 11569 \def\gmu@tempa{\kern-0,08em\penalty10000\hskip0sp\relax
11570   s\penalty10000\hskip0sp\relax}
11572 \protected\edef\Vs{V\gmu@tempa}

```

```

11574 \protected\edef\Ws{W\gmu@tempa}
11576 \pdef\Wz{W\kern-0,05em\penalty10000\hskip0sp\relax z}

```

### A left-slanted font

Or rather a left Italic *and* left slanted font. In both cases we sample the skewness of the `itshape` font of the current family, we reverse it and apply to `\itshape` in `\litshape` and `\textlit` and to `\sl` in `\lsl`. Note a slight asymmetry: `\litshape` and `\textlit` take the current family while `\lsl` and `\textlsl` the basic Roman family and basic (serif) Italic font. Therefore we introduce the `\lit` declaration for symmetry, that declaration left-slants `\it`.

I introduced them first while typesetting E. Szarzyński's *Letters* to follow his (elaborate) hand-writing and now I copy them here when need left Italic for his *Albert Camus' The Plague* to avoid using bold font.

Of course it's rather esoteric so I wrap all that in a declaration.

```

11597 \pdef\leftslanting@{%
\litdimen 11598 \def\litdimen{\strip@pt\fontdimen1\font ex}%
\litcorrection 11599 \def\litcorrection{%
11600 \ifhmode\null\nobreak\hskip\litdimen\relax\fi}%
\litkern 11601 \def\litkern{% note it's to be used inside the left slanted font, unlike \litcor|
11604 \leavevmode\null
11605 \kern-\litdimen\relax}%
\dilitkern 11606 \def\dilitkern{\kern\litdimen\litkern}%

11608 \pdef\litshape{%
11610 \litcorrection
11611 \itshape
11612 \@tempdima=-2\fontdimen1\font
11613 \advance\leftskip by\strip@pt\fontdimen1\font ex % to assure at least
the lowercase letters not to overshoot to the (left) margin. Note this has
any effect only if there is a \par in the scope.
11617 \litcorrection
11618 \edef\gmu@tempa{%
11619 \@nx\addfontfeature{FakeSlant=\strip@pt\@tempdima}}% when
not \edefed, it caused an error, which is perfectly understandable.
11622 \gmu@tempa}%
11625 \pdef\textlit##1{%
11626 {\litshape##1}}%
11628 \pdef\lit{\rm\litshape}%
11631 \pdef\lsl{%
11632 \litcorrection
11633 \it
11636 \@tempdima=-\fontdimen1\font
11637 \litcorrection
11638 \xdef\gmu@tempa{%
11639 \@nx\addfontfeature{RawFeature={%
slant=\strip@pt\@tempdima}}}%
11640 \rm % Note in this declaration we left-slant the basic Roman font not the itshape
of the current family.
11642 \gmu@tempa}%

```

Now we can redefine `\em` and `\emph` to use left Italic for nested emphasis. In Polish typesetting there is bold in nested emphasis as I have heard but we don't like bold since it perturbs homogeneous greyness of a page. So we introduce a three-cycle instead of two: Italic, left Italic, upright.

```

11650 \pdef\em{%
11651   \ifdim\fontdimen1\font=\z@ \itshape
11652   \else
11653     \ifdim\fontdimen1\font>\z@ \litshape
11654     \else \upshape
11655     \fi
11656   \fi}%
11659 \pdef\emph##1{%
11660   {\em##1}}%
11661 }% of \leftslanting@.

11663 \pdef\leftslanting{\AtBeginDocument\leftslanting@}
11665 \AtBeginDocument{\let\leftslanting\leftslanting@}

```

### Fake Old-style Numbers

While preparing documentation of this package I faced an aesthetic problem of lack of old-style numbers in a font I fancy. The font is for the sans serif and the digits occur only in the date in title so it would be a pity not too use a nice font when only one or two numbers are needed.

```

\romorzero 11676 \def\romorzero#1{%
11677   \ifnum#1=0 zero\else\romannumeral#1 \fi}

\fakeonum 11679 \DeclareCommand\fakeonum{%
11680   o % fake bold for the digit »2« (for which emboldening improves look),
11682   >Pm % the text to fake old-style numbers in.
11683 }{% I tried to use this command as a declaration but active digits are very uncomfort-
      able, e.g. you can't define macros with arguments.

11687 \gmu@if@onum{#2}{%
11688   \begingroup
11689   \edef\gmu@tempa{#2}%
11690   \makeatletter%
11691   \gmuIfValueT{#1}{%
11692     \prependtomacro\fake@onum@ii{%
11693       \begingroup\addfontfeature{FakeBold=#1}}%
11694     \addtomacro\fake@onum@ii\endgroup
11695   }%
11696   \endlinechar\m@ne % to suppress the line end added by \scantokens, especially
      in active ^^M's scopes.

11698   \gmu@dofakeonum
11699   \@xa\scantokens\@xa{\gmu@tempa}%
11700   \endgroup
11701 }% of \gmu@ifonum 'else'.
11702 }% of \fakeonum.

\gmu@dofakeonum 11704 \def\gmu@dofakeonum{%
11705   \def\do##1{%
11706     \catcode`##1\active
11707     \scantokens{%

```

```

11708     \@xa\let\@xa##1%
11709     \csname fake@onum@\@xa\romorzero\string##1\endcsname\empty}}%
11710     \do0\do1\do2\do3\do4\do5\do6\do7\do8\do9%
11711 }

11713 \def\do#1#2{%
11714     \@namedef{fake@onum@\romorzero#1}{#2}}

\gmu@tempa 11716 \def\gmu@tempa#1{%
11717     \do#1{\leavevmode
11718         \gmu@calculateslant{#1}% uses \gmu@tempa and \gmu@tempb, therefore goes
            first. And defines \gmu@tempd.
11720         \gmu@measurewd{#1}% the width of char #1 is in \gmu@tempa without kerning
            and in \gmu@tempb with kerning.

11723         \edef\gmu@tempc{\the\fontcharht\font`#1}%
11724         \hbox to \gmu@tempb {%
11725             \hss\resizebox{\gmu@tempa}%
11726             {\dimexpr\fontdimen5\font+\gmu@tempc-\fontdimen8\font}%
11727             {\gmu@tempd#1}\hss}}

11730 \gmu@tempa0 % \fake@onum@zero
11731 \gmu@tempa1 % \fake@onum@i
11732 \gmu@tempa2 % \fake@onum@ii

\gmu@tempa 11734 \def\gmu@tempa#1{%
11735     \do#1{\leavevmode
11736         \gmu@measurewd{#1}%
11737         \lower
11738         \dimexpr\fontdimen8\font-\fontdimen5\font\relax
11739         \hbox to \gmu@tempb {\hss#1\hss}}%
11740 }

11742 \gmu@tempa3 % \fake@onum@iii
11743 \gmu@tempa4 % \fake@onum@iv
11744 \gmu@tempa5 % \fake@onum@v
11745 \gmu@tempa7 % \fake@onum@vii
11746 \gmu@tempa9 % \fake@onum@ix

\gmu@tempa 11748 \def\gmu@tempa#1{% to preserve pseudo-kerning in digits sequences.
11749     \do#1{\leavevmode
11750         \gmu@measurewd#1%
11751         \hbox to \gmu@tempb {\hss#1\hss}}}

11753 \gmu@tempa6 % \fake@onum@vi
11754 \gmu@tempa8 % \fake@onum@viii

\gmu@if@onum 11757 \protected\def\gmu@if@onum{%
11758     \edef\gmu@tempa{\@xa\meaning\the\font}%
11759     \@xa\@ifinmeaning\detokenize{+onum}\of\gmu@tempa
11760 }

```

Thus `\gmu@if@onum` becomes a two-argument command that executes its #1 if there is `+onum` in current font specification or its #2 if `+onum` is absent.

One could easily generalise `\gmu@if@onum` to `\@if@fontfeature`, i.e. to a test for an arbitrary font feature, probably with employing that very nice feature specification of `fontspec`, so that you could write `\IfFontFeature{Numbers=OldStyle}{}{fake old-style digits}`.

```

11771 \pdef\gmu@getslant{% we define \gmu@tempa to the (fake) slant of current font.
11772 \edef\gmu@tempa{\@xa\meaning\the\font\detokenize{slant=0,}}%
11773 \edef\gmu@tempb{%
11774 \def\@nx\gmu@tempb###1%
11775 \detokenize{slant=}%
11776 ###2,###3}%
11777 \gmu@tempb\@nil{##2}%
11778 \edef\gmu@tempa{\@xa\gmu@tempb\gmu@tempa\@nil pt}%
11779 }

\gmu@calculateslant 11781 \def\gmu@calculateslant#1{%
11782 \gmu@getslant
11783 \edef\gmu@tempa{\the\numexpr\dimexpr\fontdimen1\font + \gmu@tempa}%
% \gmu@tempa bears the number of scaled points of total slant
% (\fontdimen1\font+ slant=... if present) per 1pt of #1.
11787 \edef\gmu@tempa{\the\numexpr \gmu@tempa *
11788 \numexpr\fontdimen5\font\relax/\numexpr\fontcharht\font`#1
11789 \relax}% we scale the total slant of #1 by the ratio of original and scaled height
of #1.
11792 \edef\gmu@tempd{%
11793 \the\dimexpr \gmu@tempa sp - \fontdimen1\font}% and we subtract slant-
fontdimen from the scaled total slant.
11795 \ifdim\gmu@tempd=\z@ \emptify\gmu@tempd
11796 \else\edef\gmu@tempd{%
11797 \@nx\addfontfeature{FakeSlant=\strip@pt\dimexpr\gmu@tempd}}%
11798 \fi}

\gmu@cepstnof 11800 \DeclareCommand\gmu@cepstnof{O{\gmu@tempa}% a cs to be \xdefed the font spec-
ification,
11802 s% not used really,
11803 m% \fontspec token or name of feature font ( Italic, Bold, SmallCaps, BoldItalic
),
11805 O{, Scale=MatchLowercase}% fontspec font features (key=val)
11806 }
11807 {% \gmu@cepstnof's body
\gmu@cepstnof@resc 11808 \def\gmu@cepstnof@resc##1:##2:\@nil{%
11809 \ifx:##2:\else RawFeature={\gmu@maybestripcomma##2,,\@nil} \fi}%

\gmu@cepstnof@resd 11811 \def\gmu@cepstnof@resd##1:##2:\@nil{% to check whether font name contains
/ (which may not be true!)
11813 \ifx&##2&\@xa\@firstoftwo\else\@xa\@secondoftwo\fi}%

\gmu@cepstnof@rese 11815 \def\gmu@cepstnof@rese##1:##2:\@nil{% to check whether the font name con-
tains : when it doesn't contain / .
11817 \ifx&##2&\@xa\@firstoftwo\else\@xa\@secondoftwo\fi}%
11819 \edef\gmu@cepstnof@resa{%

now, reserved B parses the font name and features. It uses an auxiliary reserved C
because after / may be or may not be features specification.

11823 \@xa\@xa\@xa\gmu@cepstnof@resd\@xa\meaning\the\font//\@nil
11824 {% font name doesn't contain a slash
11825 \@xa\@xa\@xa\gmu@cepstnof@rese\@xa\meaning\the\font::\@nil
11826 {% nor does it contain a colon
11827 \def\@nx\gmu@cepstnof@resb\detokenize{select font

```

```

11828         }####1\detokenize{"}####2\@nx\@nil{%
11829         \ifx\fontspec#3%
11830         \@nx\@nx\@nx\fontspec[\@gobble#4\@empty]{####1}% gobble a comma
11831         \else
11832         #3Font={####1}, #3Features={\@gobble #4\@empty}%
11833         \fi
11834     }%
11835 }%
11836 {% no slash but there is a colon
11837 \def\@nx\gmu@cepstnof@resb\detokenize{select font
11838 " }####1:####2\detokenize{"}####3\@nx\@nil{%
11839 \ifx\fontspec#3%
11840
11841         \@nx\@nx\@nx\fontspec[\@nx\gmu@cepstnof@resc####2::\@nx\@nil#4]{
11842         ####1}%
11843     \else
11844         #3Font={####1}, #3Features={%
11845         \@nx\gmu@cepstnof@resc####2::\@nx\@nil#4}%
11846     \fi
11847 }%
11848 }% of ‘no slash’ case
11849 {% font name contains a slash
11850 \def\@nx\gmu@cepstnof@resb\detokenize{select font
11851 " }####1/####2\detokenize{"}####3\@nx\@nil{%
11852 \ifx\fontspec#3%
11853
11854         \@nx\@nx\@nx\fontspec[\@nx\gmu@cepstnof@resc####2::\@nx\@nil#4]{%
11855         ####1}%
11856     \else
11857         #3Font={####1}, #3Features={%
11858         \@nx\gmu@cepstnof@resc####2::\@nx\@nil#4}%
11859     \fi
11860 }% of \gmu@cepstnof@resb
11861 }% of ‘slash present’ case
11862 } \gmu@cepstnof@resa
11863 \xdef#1{\@xa\@xa\@xa\gmu@cepstnof@resb\@xa\meaning\the\font\@nil}%
11864 }%

\gmu@stripcomma 11862 \def\gmu@stripcomma#1,{#1}

\gmu@maybestripcomma 11864 \def\gmu@maybestripcomma#1,,#2\@nil{#1}

11866 \pdef\gmu@setbasefont{\@xa\let\@xa\gmu@basefont\the\font}

11868 \let\setbasefont\gmu@setbasefont

\gmu@calc@scale 11870 \DeclareCommand\gmu@calc@scale{%
11871 O{1}% a factor,
11872 m% number of the fontdimen
11873 }
11874 {\beginingroup

We ‘descale’ the current font:
11876 \gmu@cepstnof\fontspec[, Scale=1]\gmu@tempa
11877 \@xa\let\@xa\gmu@currfont@descaled\the\font

```

```

11878 \gmu@basefont
11879 \gmu@cepstnof\fontspec[, Scale=1]\gmu@tempa
now also the base font is descaled.

11881 \xdef\gmu@fontscale{%
11882 \strip@pt
11883 \dimexpr 1pt *
11884 \numexpr\dimexpr#1\fontdimen#2\font\relax\relax /
11885 \numexpr\fontdimen#2\gmu@currfont@descaled\relax
11886 \relax}%
11887 \endgroup}

```

## Varia

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

```

\gmu@tilde 11895 \def\gmu@tilde{%
11896 \leavevmode\lower.8ex\hbox{$\,\widetilde{\mbox{ }}\,$}

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

11901 \AtBeginDocument{% to bypass redefinition of \~ as a text command with various
encodings
11903 \pdef\texttilde{%
11910 \@ifnextchar/{\gmu@tilde\kern-0,1667em\relax}\gmu@tilde}}

```

We prepare the proper kerning for “~/”.

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.

```

11919 \foone{\catcode\^^M\active}% the comment signs here are crucial.
\defobeylines 11920 {\def\defobeylines{\catcode\^^M=13 \def^^M{\par}}}
```

Another thing I dislike in L<sup>A</sup>T<sub>E</sub>X yet is doing special things for `\...skip`’s, ’cause I like the Knuthian simplicity. So I sort of restore Knuthian meanings:

```

\deksmallskip 11929 \def\deksmallskip{\vskip\smallskipamount}
\undeksmallskip 11930 \def\undeksmallskip{\vskip-\smallskipamount}
\dekmedbigskip 11931 \def\dekmedbigskip{\vskip\glueexpr \medskipamount+\smallskipamount}
\dekmedskip 11932 \def\dekmedskip{\vskip\medskipamount}
\dekbigskip 11933 \def\dekbigskip{\vskip\bigskipamount}

\hfillneg 11936 \def\hfillneg{\hskip 0pt plus -1fill\relax}

```

A mark for the **TO-DO!**s:

```

\TODO 11941 \newcommand*\TODO[1][ ] [ ] {%
11942 \sffamily\bfseries\huge TO-DO!\if\relax#1\relax\else\space\fi#1}}

```

I like two-column tables of contents. First I tried to provide them by writing `\begin{% multicols}{2}` and `\end{multicols}` out to the `.toc` file but it worked wrong in some cases. So I redefine the internal L<sup>A</sup>T<sub>E</sub>X macro instead.

```

\twocoltoc 11950 \newcommand*\twocoltoc{%
11951 \RequirePackage{multicol}%
\@starttoc 11952 \def\@starttoc##1{%
11953 \begin{multicols}{2}\makeatletter

```



```

11954 \NamedInput % changed from \@input 2010/8/13.
11955 {\jobname .##1}%
11956 \if@filesw \@xa \newwrite \csname tf@##1\endcsname
11957 \immediate \openout \csname tf@##1\endcsname \jobname
        .##1\relax
11958 \fi
11959 \@nobreakfalse\end{multicols}%
11960 }% of \starttoc
11961 }

```

```
11963 \@onlypreamble\twocoltoc
```

An equality sign properly spaced:

```
11968 \pdef\equals{\hunskip${}={}\$ \ignorespaces}
```

And for the L<sup>A</sup>T<sub>E</sub>X's pseudo-code statements:

```
11970 \pdef\eequals{\hunskip${}=={}\$ \ignorespaces}
```

```
11972 \pdef\cdot{\hunskip${}\cdot{}\$ \ignorespaces}
```

While typesetting a UTF-8 ls-R result I found a difficulty that follows: UTF-8 encoding is handled by the `inputenc` package. It's O.K. so far. The UTF-8 sequences are managed using active chars. That's O.K. so far. While writing such sequences to a file, the active chars expand. You feel the blues? When the result of expansion is read again, it sometimes is again an active char, but now it doesn't star a correct UTF-8 sequence.

Because of that I wanted to 'freeze' the active chars so that they would be `\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 11992 \def\freeze@actives{%
11993 \count\z@\z@
11995 \@whilenum\count\z@<\ccclvi\do{%
11996 \ifnum\catcode\count\z@=\active
11997 \uccode`~=\count\z@
11998 \uppercase{\let~\relax}%
11999 \fi
12000 \advance\count\z@\@ne}}

```

A macro that typesets all 256 chars of given font. It makes use of `\@whilenum`.

```

\ShowFont 12006 \newcommand*\ShowFont[1][6]{%
12007 \begin{multicols}{#1}[The current font (the \f@encoding\ encoding):]
12008 \parindent\z@
12009 \count\z@\m@ne
12010 \@whilenum\count\z@<\ccclv\do{
12011 \advance\count\z@\@ne
12012 \ \the\count\z@:\sim\char\count\z@\par}
12013 \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 12021 \newcommand*\liturgiques[1][red]{% Requires the color package.
12022 \gmu@RPfor{xcolor}\color%
\czerwo 12023 \newcommand*\czerwo{\small\color{#1}}% environment
\czer 12024 \newcommand{\czer}[1]{\leavevmode{\czerwo##1}}% we leave vmode because if

```

we don't, then `verse's \everypar` would be executed in a group and thus its effect lost.

```

12027 \Store@Macro\*%
12028 \def\*\{\czer{\storedcsname{*}}}%
\+ 12029 \def\+\{\czer{+}}%
\nieczer 12030 \newcommand*\nieczer[1]{\textcolor{black}{##1}}%
12031 }

```

After the next definition you can write `\gmu@RP[<options>]{<package>}{<CS>}` to get the package #2 loaded with options #1 if the CS#3 is undefined.

```

\gmu@RPfor 12036 \newcommand*\gmu@RPfor[3][[]]{%
12038 \ifx\relax#1\relax\emptify\gmu@resa
\gmu@resa 12039 \else \def\gmu@resa{[#1]}%
12040 \fi
12041 \@xa\RequirePackage\gmu@resa{#2}}

```

Since inside `document` we cannot load a package, we'll redefine `\gmu@RPfor` to issue a request before the error issued by undefined CS.

```

12046 \AtBeginDocument{%
\gmu@RPfor 12047 \renewcommand*\gmu@RPfor[3][[]]{%
12048 \unless\ifdefined#3%
12049 \@ifpackageloaded{#2}{}%
12050 \typeout{^^J! Package `#2' not loaded!!! (\on@line)^^J}}%
12051 \fi}}

```

It's very strange to me but it seems that `c` is not defined in the basic math packages. It is missing at least in the *Symbols* book.

```

12057 \pprovide\continuum{%
12058 \gmu@RPfor{eufрак}\mathfrac\ensuremath{\mathfrac{c}}{}}

```

And this macro I saw in the `ltugproc` document class and I liked it.

```

\iteracro 12062 \def\iteracro{%
12063 \pdef\acro##1{%
12064 \begingroup
12065 \acroprsetting
12066 \gmu@acrospace##1 \gmu@acrospace
12067 \endgroup
12068 }%
12069 }

12071 \emptify\acroprsetting

12073 \iteracro

\gmu@acrospace 12075 \def\gmu@acrospace#1 #2\gmu@acrospace{%
12076 \gmu@acroiter#1\gmu@acroiter
12077 \ifx\relax#2\relax\else
12078 \space
12079 \afterfi{\gmu@acrospace#2\gmu@acrospace}% when #2 is nonempty, it is
ended with a space. Adding one more space in this line resulted in an
infinite loop, of course.
12083 \fi
12084 }

\gmu@acroiter 12087 \def\gmu@acroiter#1{%

```

```

12088 \gmu@notif {x\@xa\@xa} {\@firstofmany#1\@undefined\@nil
      \gmu@acroiter}
12089 {\gmu@acrokernel{#1}%

```

and iterate

```

12091 \gmu@acroiter
12092 }% of if sentinel or not
12093 {}%
12094 }

```

```

\gmu@acrokernel 12097 \def\gmu@acrokernel #1{%
12098 \gmu@if {cat} {a\@xanx\@firstofmany#1\@undefined\@nil}%
12099 {\gmu@if {num} {\`#1=\ucode`#1\space }% a space to delimit numer (without
      it further macros were expanded which in this case are \expandafter%
      \@firstoftwo\else\@secondoftwo and this made the test didn't work.)
12103 {{\acrocore{#1}}}%
12104 {{#1}}% tu było \smallerr
12105 }%
12106 {#1}%
12107 }

```

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.

```

12112 \pdef\acrocore{\smaller % was: \scshape\lowercase
12113 }

```

Since the fonts I am currently using do not support required font feature, I skip the following definition.

```

\IMHO 12118 \def\IMHO{\acro{IMHO}\spifletter}
\AKA 12119 \def\AKA{\acro{AKA}\spifletter}
12121 \pdef\usc#1{{\addfontfeature{Letters=UppercaseSmallCaps}#1}}

```

```

\uscacro 12123 \def\uscacro{\let\acrocore\usc}

```

\SecondClass moved to **gmbase**.

Cf. *The T<sub>E</sub>X book* ex. 11.6.

A line from L<sup>A</sup>T<sub>E</sub>X:

```
% \check@mathfonts\fontsize\sf@size\z@\math@fontsfalse\selectfont
```

didn't work as I would wish: in a \footnotesize's scope it still was \scriptsize, so too large.

```

\gmu@dekfraccsimple 12135 \def\gmu@dekfraccsimple#1/#2{\leavevmode\kern.1em
12136 \raise.7ex\hbox{%
12137 \gmu@fracfontsetup#1}\gmu@numeratorkern
12138 \dekfraccslash\gmu@denominatorkern
12139 {%
12140 \gmu@fracfontsetup#2}%
12141 \if@gmu@mmhbox\egroup\fi}

```

```

\gmu@fracfontsetup 12143 \def\gmu@fracfontsetup{%
12144 \smaller[3]\addfontfeature{FakeBold=1}}

```

```

\dekfraccsimple 12147 \def\dekfraccsimple{%
12148 \let\dekfracc@args\gmu@dekfraccsimple

```

```

12149 }
\dekfracslash 12150 \ifXeTeX{\def\dekfracslash{\char"2044 }}
\dekfracslash 12151 {\def\dekfracslash{/}} % You can define it as the fraction slash, \char"2044
12153 \dekfracssimple

```

A macro that acts like \, (thin and unbreakable space) except it allows hyphenation afterwards:

```
\ikern 12161 \newcommand*\ikern{\, \penalty\@M\hskip\z@skip\relax}
```

### Faked small caps

```

\gmu@scapLetters 12168 \def\gmu@scapLetters#1{%
12169   \ifx#1\relax\relax\else% two \relaxes to cover the case of empty #1.
12170   \ifcat a\@nx#1\relax
12171   \ifnum\the\lccode`#1=`#1\relax
12172   {\fakescapscore\MakeUppercase{#1}}% not Plain \uppercase because
      that works bad with inputenc.
12174   \else#1%
12175   \fi
12176   \else#1%
12177   \fi%
12178   \@xa\gmu@scapLetters
12179   \fi}%
\gmu@scapSpaces 12181 \def\gmu@scapSpaces#1 #2\@nil{%
12182   \ifx#1\relax\relax
12183   \else\gmu@scapLetters#1\relax
12184   \fi
12185   \ifx#2\relax\relax
12186   \else\afterfi{\ \gmu@scapSpaces#2\@nil}%
12187   \fi}
\gmu@scapss 12189 \def\gmu@scapss#1\@nil{{\def~{{\nobreakspace}}}%
\nobreakspace 12190   \gmu@scapSpaces#1 \@nil}}% % \def\\{{\newline}}\relax adding redefini-
      tion of \\ caused stack overflow. Note it disallows hyphenation except at
      \-.

```

```
12194 \pdef\fakescaps#1{{\gmu@scapss#1\@nil}}
```

```
12196 \let\fakescapscore\gmu@scalematchX
```

Experimete z akcentami patrz no3.tex.

```
\tinycae 12199 \def\tinycae{{\tiny\AE}}% to use in \fakescaps[\tiny]{...}
```

```
12201 \RequirePackage{calc}
```

wg \zf@calc@scale pakietu fontspec.

```
12205 \@ifpackageloaded{fontspec}{%
```

```
\gmu@scalar 12206   \def\gmu@scalar{1.0}%
```

```
\zf@scale 12207   \def\zf@scale{}%
```

```
\gmu@scalematchX 12208   \def\gmu@scalematchX{%
```

```
12209     \begingroup
```

```
\gmu@scalar 12210     \ifx\zf@scale\empty\def\gmu@scalar{1.0}%
```

```
12211     \else\let\gmu@scalar\zf@scale\fi
```

```
12212     \setlength\@tempdima{\fontdimen5\font}% 5—ex height
```

```
12213     \setlength\@tempdimb{\fontdimen8\font}% 8—XqTeX synthesised upper-
      case height.

```

```

12215     \divide\@tempdimb by1000\relax
12216     \divide\@tempdima by\@tempdimb
12217     \setlength{\@tempdima}{\@tempdima*\real{\gmu@scalar}}%
12218     \gmu@ifundefined{fakesc@extrascale}{}{%
12219         \setlength{\@tempdima}{\@tempdima*\real{%
            \fakesc@extrascale}}}%
12220     \@tempcnta=\@tempdima
12221     \divide\@tempcnta by 1000\relax
12222     \@tempcntb=-1000\relax
12223     \multiply\@tempcntb by\@tempcnta
12224     \advance\@tempcntb by\@tempdima
12225     \xdef\gmu@scscale{\the\@tempcnta.%
12226         \ifnum\@tempcntb<100 0\fi
12227         \ifnum\@tempcntb<10 0\fi
12228         \the\@tempcntb}%
12229     \endgroup
12230     \addfontfeature{Scale=\gmu@scscale}%
12231 }{\let\gmu@scalematchX\smallerr}

```

```

\fakescextrascale 12233 \def\fakescextrascale#1{\def\fakesc@extrascale{#1}}
\fakesc@extrascale

```

### See above/see below

To generate a phrase as in the header depending of whether the respective label is before of after.

```

\wyzejnizej 12239 \newcommand*\wyzejnizej[1]{%
12240     \edef\gmu@tempa{\gmu@ifundefined{r@#1}{\arabic{page}}{%
12241         \@xa\@xa\@xa\@secondoftwo\csname r@#1\endcsname}}%
12242     \ifnum\gmu@tempa<\arabic{page}\relax wy\.\zej\fi
12243     \ifnum\gmu@tempa>\arabic{page}\relax ni\.\zej\fi
12244     \ifnum\gmu@tempa=\arabic{page}\relax \@xa\ignorespaces\fi
12245 }

```

### luzniej and napapierki—environments for fine-tuning of paragraphs vs. page breaks

The name of first of them comes from Polish typesetters’ phrase “rozbijać [skład] na papierki”—‘to broaden [leading] with paper scratches’. The English term is “feathering” or “carding”.

**WARNING.** According to the guidelines for good typographers’ practice, e.g., Filip Trzaska’s manual of 1960’s, the practice mentioned above and facilitated by the environment defined below is

**WRONG / UNETHICAL / SINFUL / AN ABOMINATION / BAD KARMA\*** at least in Poland (although probably not illegal in most countries, including Poland).

Please think twice before committing it and don’t blame me when judged by your Buddha(s) / God(s) / Conscience(s) / Atman(s) / Karman(s) / Brahman(s)\*.

\* depending on your Ethics/Morality Provider.

Please note also that using the **geometry** package option **lines=***(number)* or **heightrounded** is pretty sufficient in typical cases to get leading without over- and underfulls.

FYI, the proper practice seem to be, translating into T<sub>E</sub>X terms, to set **\baselineskip** rigid (as the standard classes do and probably all the others), set all the standard vskips as (integer) multiples of the normal-size **\baselineskip** and then, when a part of text is set with smaller or larger font sizes, complete it with fractional vskips that with its own height add up to an integer multiple of the normal **\baselineskip**.

If all this is done correctly, all the normal size lines are placed evenly on all pages, which is called “register” in Polish and “grid” in English.

Now, having said that, let’s provide you with a tool of temptation and fall.

```
\napapierkistretch 12290 \def\napapierkistretch{0,3pt}% It's quite much for 11/13pt leading.
\napapierkicore 12292 \def\napapierkicore{\advance\baselineskip%
12293 by 0ptplus\napapierkistretch\relax}
12296 \DeclareEnvironment{napapierki}{s}{%
12298 \par\gmuIfValueT{#1}{\global}%
12300 \napapierkicore}
12301 {%
12302 \par
12303 \gmuIfValueT{#1}{\global\baselineskip=1\baselineskip\relax
12304 }%
12305 }% so that you can use \napapierki*...\endnapapierki* in interlacing environments.
```

```
\gmu@luzniej 12318 \newcount\gmu@luzniej
\nuzniejcore 12320 \newcommand*\luzniejcore[1][1]{%
12321 \advance\gmu@luzniej\@ne% We use this count to check whether we open the en-
environment or just set \looseness inside it again.
12323 \ifnum\gmu@luzniej=\@ne \multiply\tolerance by 2 \fi
12324 \looseness=#1\relax}
```

After `\begin{luzniej}` we may put the optional argument of `\luzniejcore`

```
luzniej 12328 \newenvironment*{luzniej}{\par\luzniejcore}{\par}
```

The starred version sets `\looseness` in `\everypar`, which has its advantages and disadvantages.

```
luzniej* 12333 \newenvironment*{luzniej*}[1][1]{%
12334 \multiply\tolerance by 2\relax
12335 \everypar{\looseness=#1\relax}}{\par}
```

```
\nawj 12337 \newcommand*\nawj{\kern0,1em\relax}% a kern to be put between parentheses and
letters with descendants such as j or y in certain fonts.
```

The original `\pauza` of `polski` has the skips rigid (one is even a kern). We make the skips flexible. Moreover, our `\pauza` begins with `\ifhmode` to be usable also at the beginning of a line where it marks a part of a dialogue.

```
\pauza@skipcore 12346 \def\pauza@skipcore{\hskip0.2em plus0.1em\relax
12347 \pauzacore
12348 \@ifnextanyIS {, ; :}%
12349 {% 2009/11/22 added a special case of a comma following pauza, 2011/02/22, 13.33
extended to colon and semicolon (well, the latter should not happen in such a
position but a colon did (in Polish)).
12353 }{\hskip.2em plus0.1em\relax\ignorespaces}}%
\ppauza@skipcore 12355 \def\ppauza@skipcore{\unskip\penalty10000\hskip0.2em plus0.1em\relax
12356 \ppauza@dash\hskip.2em plus0.1em\ignorespaces}
12359 \AtBeginDocument{%
12360 \pdef\pauza{%
12361 \ifhmode
12362 \unskip\penalty10000
12363 \hskip0.2em plus0.1em\relax
```

```

12364     \pauzacore\hskip.2em plus0.1em\relax\ignorespaces%
12365     \else
12366     \pauzadial
12367     \fi
12368 }%

```

According to *Instrukcja technologiczna. Skład ręczny i maszynowy* the dialogue dash (in Polish) should be followed by a rigid hskip of 1/2 em.

```

12373 \pdef\pauzadial{%
12374     \ifhmode\unskip\quad\else\leavevmode\fi
12375     \pauzacore\penalty10000\hskip0,5em\ignorespaces
12376 }

```

And a version with no space at the left, to begin a `\noindented` paragraph explaining e.g. a quotation:

```

12380 \pdef\lpauza{%
12381     \leavevmode
12382     \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_10`. 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...`

```

12391 \pdef\ppauza{%
12392     \ifvmode \PackageError{gmutils}{%
12393         command \bslash ppauza (en dash) not intended for vmode.}%
12394     Use \bslash ppauza (en dash) only in number and numeral
12395         ranges.}%
12396     \else
12397     \unskip\discretionary
12398     {\ppauza@dash}{\ppauza@dash}{\ppauza@dash}%
12399 }% of at begin document

```

```

12401 \ifdefined\XeTeXversion
12402 \AtBeginDocument{% to be independent of moment of loading of polski.
12403     \pdef\—{%
12404         \ifhmode
12405         \unskip\penalty10000
12406         \afterfi{%
12407             \@ifnextanyRS { , ; }% was comma as a special case; 2011/02/22, 13.32
12408             added colon and semicolon.

```

The above needs some explanation, I suppose. We check if next token is blank space or some punctuations, for which purpose we use `\@ifnextanyRS`, where `RS` stands for “respecting space”. So if next is a space it turns true. Then `\pauza@skipcore`'s test does not respect space (gobbles it) and typesets punctuators with no skip while a blank replaces with a skip.

```

12419     {\pauza@skipcore}%
12420     {\@ifnextMac{\pauza@skipcore}%
12421         {\pauzacore\penalty\hyphenpenalty\hskip\z@skip}%
12422     }%
12423 }% of \afterfi's argument

```

```

12424     \else
According to Instrukcja technologiczna. Skład ręczny i maszynowy the dialogue dash
should be followed by a rigid hskip of 1/2 em.
12428     \leavevmode\pauz@core\penalty10000\hskip0,5em\ignorespaces
12429     \fi
12430 }%

```

The next command's name consists of letters and therefore it eats any spaces following it, so `\@ifnextspace` would always be false, therefore we don't use it.

```

12434     \pdef\--{%
12435     \ifvmode     \PackageError{gmutils}{%
12436     command \bslash ppauza (en dash) not intended for vmode.}{%
12437     Use \bslash ppauza (en dash) only in number and numeral
        ranges.}%
12438     \else
12439     \afterfi{%
12440     \@ifnextspace{\ppauza@skipcore}{%
12441     \@ifnextMac\ppauza@skipcore
12442     {\unskip\discretionary
12443     {\ppauza@dash}{\ppauza@dash}{\ppauza@dash}}}%
12444     }%
12445     \fi
12446 }%
\emdash 12448 \def\emdash{\char`-\}
12449 }% of at begin document

```

```

\longpauza 12451 \def\longpauza{%
\pauz@core@long 12452 \def\pauz@core@long{---}%
12453 \let\pauz@core\pauz@core@long
12454 }
12455 \longpauza

```

```

\shortpauza 12457 \def\shortpauza{%
\pauz@core@short 12458 \def\pauz@core@short{\hbox{-\kern,23em\relax\llap{-}}}%
12459 \let\pauz@core\pauz@core@short
12460 }%

```

```

\ppauza@dash 12461 \def\ppauza@dash{-}%
12464 \else % not XqTeX
\longpauza 12465 \def\longpauza{\def\pauz@core{---}}
\pauz@core 12466 \longpauza
\shortpauza 12467 \def\shortpauza{%
\pauz@core 12468 \def\pauz@core{--\kern,23em\relax\llap{--}}}%
\ppauza@dash 12469 \def\ppauza@dash{--}%

```

```

12472 \fi% of if XqTeX or not.

```

```

12475 \ifdefined\XeTeXversion

```

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 `-`.

```

12481 \foone{\catcode`-\active \catcode`-\active \catcode`-\active}{%
\adashes 12482 \def\adashes{\AtBeginDocument\adashes}% because \pauza is defined at begin
document.
\adashes 12484 \AtBeginDocument{\def\adashes{%

```



```

\catcode`-\active \def-{\-}%
\catcode`-\active \def-{\-}%
\addtomacro\dospecials{\do\-\do\-\}%
\addtomacro\@sanitize{\@makeother\-\@makeother\-\}%
\addtomacro\gmu@septify{\do\-\13\do\-\13\relax}%
}}
\else
\relaxen\adashes
\fi

```

The hyphen shouldn't be active IMHO because it's used in T<sub>E</sub>X control such as `\hskip-2pt`. Therefore we provide the `\ahyphen` declaration reluctantly, because sometimes we need it and always use it with caution. Note that my active hyphen in vertical and math modes expands to `-12`.

```

\gmu@dywiz 12502 \def\gmu@dywiz{\ifmmode-\else
12503 \ifvmode-\else\afterfifi\dywiz\fi\fi}%
12505 \foone{\catcode`-\active}{% aktywnej diefis aktywny dywiz active hyphen
\ahyphen 12506 \def\ahyphen{\let-\gmu@dywiz\catcode`-\active}}

```

To get current time. Works in  $\epsilon$ -T<sub>E</sub>Xs, including X<sub>Y</sub>T<sub>E</sub>X. `\czas` typesets 15:20 and `\czas[:]` typesets 15:20.

```

\czas 12511 \newcommand*\czas[1][.]{%
12512 \the\numexpr(\time-30)/60\relax#1%
12513 \@tempcnta=\numexpr\time-(\time-30)/60*60\relax
12514 \ifnum\@tempcnta<10 0\fi\the\@tempcnta}
tytulowa 12516 \newenvironment*{tytulowa}{\newpage}{\par\thispagestyle{%
empty}\newpage}

```

To typeset peoples' names on page 4 (the editorial page):

```

\nazwired 12519 \def\nazwired{\quad\textsc}

```

### Typesetting dates in my memoirs

A date in the `YYYY-MM-DD` format we'll transform into 'DD mmmm YYYY' format or we'll just typeset next two tokens/`{...}` if the arguments' string begins with `--`. The latter option is provided to preserve compatibility with already used macros and to avoid a starred version of `\thedata` and the same time to be able to turn `\datef` off in some cases (for `SevSev04.tex`).

```

12534 \pdef\polskadata{%
\gmu@datefsl 12535 \DeclareCommand\gmu@datefsl{%
12536 ##1 Q{0123456789\bgroup}>iT{/} % year
12537 ##2 Q{0123456789\bgroup}>iT{/} % month
12538 ##3 Q{0123456789\bgroup} % day
12539 >is % terminator of date scanning that will be gobbled
12540 ##4 T{,}% another terminator of date, but this will be printed
12541 ##5 K{##1\gmu@datefsl} % additional stuff after comma (or instead date)
12542 }{%
12543 \gmuIfValueF{##2}{\gmuPutIfValue{##3}}%
12544 \gmuIfValueT{##2}{%
12545 \@tempcnta=0##3\relax\the\@tempcnta
12546 \ifcase##2\relax\or\ stycznia\or\ lutego%

```

```

12547      \or\ marca\or\ kwietnia\or\ maja\or\ czerwca\or\ lipca\or\
           sierpnia%
12548      \or\ września\or\ października\or\ listopada\or\ grudnia\else
12549      {}%
12550      \fi}%
12551      \gmuIfValueT{##1}{\space ##1}%
12552      \gmuPutIfValue{##4}%
12553      \gmu@ifempty{##5}{{ ##5}%
12554      }% of \gmu@datefsl.
12555      }% of \polskadata
12557      \polskadata

```

For documentation in English:

```

12560      \pdef\englishdate{%
\gmu@datefsl 12561      \DeclareCommand\gmu@datefsl{%
12562          Q{0123456789\bgroup}>iT{/ -} % (1) year
12563          Q{0123456789\bgroup}>iT{/ -} % (2) month
12564          Q{0123456789\bgroup} % (3) day
12565          >is
12566          T{,} K{##1\gmu@datefsl} % (4, 5) additional stuff after comma
12567      }{%
12568          \gmuIfValueF{##2}{\gmuPutIfValue{##3}}%
12569          \gmuIfValueT{##2}{%
12570              \ifcase##2\relax\or January\or February%
12571              \or March\or April\or May\or June\or July\or August%
12572              \or September\or October\or November\or December\else
12573              {}%
12574          \fi}%
12575          \space
12576          \@tempcnta=##3\relax\the\@tempcnta,
12577          \gmuIfValueT{##1}{ ##1}%
12578          \gmuPutIfValue{##4}%
12579          \gmu@ifempty{##5}{{ ##5}%
12580          }% of \gmu@datefsl.
12581      }%

```

Dates for memoirs to be able to typeset them also as diaries.

```

\ifdate 12586      \newif\ifdate
12588      \pdef\bidate #1{%
12589          \gmu@datefsl#1\gmu@datefsl
12590      }

\linedate 12592      \DeclareCommand\linedate{s{m}
12593      {%
12594          \par
12595          \linedate@hook #1#2\@nil%
12596          \addvspace{\dateskipamount}%
12597          \possvfil% if we put it before \addvspace, the v-space is always added.
12600          \ifdate
%%          \addvspace{\dateskipamount}%
12602          \if@CofaĆPrzedData@

```

```

12603     \penalty\@M
12604     \vskip-0,5\baselineskip
12605     \else
12606     \@CofacPrzedData@true
12607     \fi
12609     \date@line{\DateFont \bdate{#1#2}}%
12610     \nopagebreak

%%     \else% %\ifnum\arabic{dateinsection}>0\dekbigskip\fi
%%     \addvspace{\bigskipamount}\possvfil

12613     \fi
12614 }% end of \linedate.

\if@CofacPrzedData@ 12616 \newif\if@CofacPrzedData@
12617 \@CofacPrzedData@true

12619 \pdef\NoPredate {\@CofacPrzedData@false}

\DateFont 12621 \newcommand*\DateFont{\footnotesize\itshape}

\linedate@hook 12623 \def\linedate@hook #1\@nil{}

\dateskipamount 12625 \newskip\dateskipamount
12626 \dateskipamount\medskipamount

12628 \pdef\rdate{\let\date@line\rightline \linedate}

\date@left 12631 \def\date@left#1{\par{%
12632     \raggedright#1%
12633     \leftskip\z@skip
12634     \@@par}}%

12636 \pdef\ldate{%
12638     \let\date@line\date@left
12639     \linedate}

\runindate 12641 \newcommand*\runindate[1]{%
12642     \paragraph{\footnotesize\itshape \gmu@datef#1\gmu@datef}%
12643     \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.

12646 \let\thedata\ldate

12649 \pdef\zwrobcy#1{\emph{#1}} % ostinato, allegro con moto, garden party etc., także
kompliment

12652 \pdef\tytul#1{\emph{#1}}

Maszynopis w świecie justowanym zrobi delikatną chorągiewkę. (The maszynopis
environment will make a delicate ragged right if called in a justified world.)

maszynopis 12658 \newenvironment{maszynopis}[1][\ttfamily
12659     \hyphenchar\font=45\relax% this assignment is global for the font.
12660     \@tempskipa=\glueexpr\rightskip+\leftskip\relax
12661     \ifdim\gluestretch\@tempskipa=\z@
12662     \tolerance900
it worked well with tolerance = 900.
12664     \advance\rightskip by\z@ plus0,5em\relax\fi

```

```

12665 \fontdimen3\font=\z@% we forbid stretching spaces...
% \fontdimen4\font=\z@ but allow shrinking them.
12667 \hyphenpenalty0 % not to make TEX nervous: in a typewriting this marvellous
algorithm of hyphenation should be turned off and every line broken at the
last allowable point.
12670 \Store@Macro\pauzacore
\pauzacore 12671 \def\pauzacore{-\rlap{\kern-0,3em-}}%
12672 }\par}

12676 \pdef\justified{%
12677 \leftskip=1\leftskip% to preserve the natural length and discard stretch and
shrink.
12679 \rightskip=1\rightskip
12680 \parfillskip=1\parfillskip
12681 \advance\parfillskip by 0sp plus 1fil\relax
12682 \let\\\@normalcr}

```

To conform Polish recommendation for typesetting saying that a paragraph's last line leaving less than `\parindent` should be stretched to fill the text width:

```

\fullpar 12687 \DeclareCommand\fullpar{%
12688 T{+-}%
12689 Q{+-0123456789} % optional looseness (most probably negative)
12690 }{%
12691 \begingroup
12692 \gmuIfValueT{#1}{\looseness=#1\gmuIfValueTF{#2}{#2}{1}\relax
12693 \multiply\tolerance by \tw@
12694 }%
12695 \fullparcore
12696 \par
12697 \endgroup}

12699 \pdef\fullparcore{%
12700 \hunskip
12701 \parfillskip\z@skip}

```

To conform Polish recommendation for typesetting that says that the last line of a paragraph has to be `2\parindent` long at least. The idea is to set `\parfillskip` naturally rigid and long as `\textwidth-2\parindent`, but that causes non-negligible shrinking of the inter-word spaces so we provide a declaration to catch the proper glue where the `parindent` is set (e.g. in footnotes `parindent` is 0pt)

```

\twoparinit 12711 \newcommand*\twoparinit{% the name stands for 'last paragraph line's length minimum
two \parindent.
\twopar@defts 12713 \def\twopar@defts{%
12714 \hspace-\leftskip-\rightskip-\fontcharwd\font`...}%
\twopar@atleast 12715 \def\twopar@atleast{2\@parindent}%
\twopar 12716 \DeclareCommand\twopar{%
12717 T{+-}% (1) you can specify loosening the paragraph by one only by typing single
+ and tightening by one by typing single -.
12719 Q{+-0123456789}% (2)
12720 A{\twopar@atleast}% (3)
12721 >iT{\cipolagwa}}{%
12722 \begingroup
12723 \gmuIfValueT{##1}{%
12724 \looseness=##1\gmuIfValueTF{##2}{##2}{1}\relax

```

```

12725     \multiply\tolerance by2
12726     }%
12727     \twoparcore<##3>%
12728     \endgraf
12729     \endgroup
12730 }% of \twopar.

12732 \ifdefined\XeTeXversion
\twoparcore 12733     \DeclareCommand\twoparcore{%
12734         A{\twopar@atleast}
12735         \gobblespace
12736     }{%
12737         \hunskip % it's O.K. it's in a group, it'll work anyway.
12738         \edef\gmu@tempa{\the\dimexpr\twopar@defts-##1\relax}%
12739         \parfillskip=\glueexpr\gmu@tempa minus \gmu@tempa
12740         \relax% to delimit \glueexpr.
12741         \relax% to delimit the assignment.
12742     }%
12743 \else % not XeTeX—doesn't use \fontcharwd.
\twoparcore 12744     \DeclareCommand\twoparcore{%
12745         A{\twopar@default}
12746         \gobblespace
12747     }{%
12748         \hunskip % it's O.K. it's in a group, it'll work anyway.
12749         {\setbox0=\hbox{\dots}}%
12750         \xdef\gmu@tempa{\the\wd0}}%
12751         \edef\gmu@tempa{%
12752             \the\dimexpr\hsize-\leftskip-\rightskip
12753             -\gmu@tempa-2\@parindent\relax}%
12754         \parfillskip=\glueexpr\gmu@tempa minus \gmu@tempa
12755         \relax% to delimit \glueexpr.
12756         \relax% to delimit the assignment.
12757     }%
12758     \fi

12760 \AtBeginDocument{%
\restoreparindent 12761     \def\restoreparindent{\parindent\@parindent}%
12762     }% of \AtBeginDocument.
12763 }% of \twoparinit.

```

### For dati under poems

Or explanations under results of `time`.

```

\gmu@leftskipcorr 12771 \def\gmu@leftskipcorr{%
12772     \kern1\leftskip
12773     \ifcsname \@currentvir leftskip\endcsname
12774     \kern-1\csname \@currentvir leftskip\endcsname
12775     \fi
12776 }

```

```

\wherncore 12779 \DeclareCommand\wherncore{om}{%
    % [#1] optional value of \hskip of (left) indent of the parbox. If absent,
    % parbox is aligned right;
    % [#2] ot ext for the datum parbox.

```

```

12785 \gmuIfValueTF{#1}{\leftline{%
12786     \kern1\leftskip
12787     \whernfont
12788     \hskip#1\relax\parbox
12789     {\dimexpr\hsize-\leftskip-\rightskip-#1}%
12790     {#2}% of \parbox,
12791     }% of \leftline,
12792 }% of ValueT{#1}.
12793 {% ValueF{#1}:
12794     \rightline
12795     {\whernfont
12796     \whern@parbox{#2}%
12797     \kern1\rightskip
12798     }% of \rightline,
12799     \setprevdepth
12800     }% of ValueF{#1},
12801 }% of \wherncore.

\whern@parbox 12804 \DeclareCommand\whern@parbox{%
12805     T {\leftskip\rightskip}% horizontal alignment of resulting box (the side to be
                ragged)
12807     O{t} % vertical alignment of parbox
12808     >is % separator
12809     O{0,7666\hsize} % (3) width of parbox
12810     m % (4) parbox contents
12811 }{%

    % #1 S the skip of the ragged side,
    % #2 S t he \parbox's contents.

12816     \parbox[#2]{#3}{%
12817         \gmuIfValueTF{#1}{#1}{\leftskip}=0sp plus \textwidth
12818         \parfillskip0sp\relax
12819         \let\\\linebreak
12820         \disobeylines
12821         \whernfont #4\unskip\strut\endgraf
12822         \getprevdepth
12823     }% of \parbox,
12824 }% of \whern@parbox.

\whern 12826 \def\whern{%
12827     \endgraf\nopagebreak
12828     \gmu@ifstar{\wherncore}%
12829     {\vskip\whernskip\wherncore}%
12830 }

12832 \let\whernfont\footnotesize

\whernskip 12834 \newskip\whernskip
12835 \whernskip2\baselineskip minus 2\baselineskip\relax

12837 \foone{\obeylines}{%

\whernup 12838 \DeclareCommand\whernup{%
12839     #1     o % a vskip before
12840     >is % separating star (ignored)
12841     #2     o % custom width of parbox

```

```

12842 #3    >Pm%
12843 #4    o % a vskip between this box and subsqt. text
12844     >iQ{ \par
12845     }%
12846   }% of args' spec
12847 }% of obey lines
12848 {\par
12849   \gmuIfValueT{#1}{\vskip#1\relax}%

```

The `prosato` env. of `gmverse` & derivs. set some positive `parskip` and `counterkrank` its first occurrence by a `vskip` of its negative. Therefore here we put ourselves in a `\rlap` and put at the very beginning of a paragraph.

```

12855   \ifinner \noindent\nobreak\kern-1\leftskip \nobreak
12856     \rlap{%
12857       \vbox{%
12858         \fi % of if inner
12859         \leftline{%
12861           \gmuIfValueTF{#2}{\whern@parbox\rightskip[b][#2]}%
12862           {\whern@parbox\rightskip[b]}%
12863           {#3}%
12864         }%
12865       \ifinner
12866         \gmuIfValueT {#4}{\vskip\glueexpr #4\endexpr }%
12867         \phantom{Ala ma Filifjonkgyr}%
12868       }% of vbox
12869     }% of rlap
12870
12872   \penalty\@M
12873   \kern 1\leftskip
12874   \@ifenvir{quote}{\hskip\z@skip}{\hskip\parindent\relax}%
12875   \else
12876     \gmuIfValueT {#4}{\vskip \glueexpr #4\endexpr }%
12877   \fi
12878 }% of \whernup's body.

```

### Thousand separator

```

12884 \pdef\thousep#1{% a macro that'll put the thousand separator between every two
      three-digit groups.

```

First we check whether we have at least five digits.

```

12888   \gmu@thou@fiver#1\relax\relax\relax\relax\relax% we put five \relaxes
      after the parameter to ensure the string will meet \gmu@thou@fiver's
      definition.
12891   \gmu@thou@fiver{#1}{% if more than five digits:
12892     \emptify\gmu@thou@put
12893     \relaxen\gmu@thou@o\relaxen\gmu@thou@i\relaxen\gmu@thou@ii
12894     \@tempcnta\z@
12895     \gmu@thou@putter#1\gmu@thou@putter
12896     \gmu@thou@put
12897   }}

```

```

\gmu@thou@fiver 12899 \def\gmu@thou@fiver#1#2#3#4#5\gmu@thou@fiver#6#7{% this macro only checks if
      the text delimited with itself consists of at least five tokens/braces
12901   \ifx\relax#5\relax\@xa\@firstoftwo

```

```

12902 \else\@xa\@secondoftwo
12903 \fi{#6}{#7}}
\gmu@thou@putter 12906 \def\gmu@thou@putter #1#2{% we are sure to have at least five tokens before the
                    sentinel \gmu@thou@putter.
12908 \advance\@tempcnta\@ne
12909 \@tempcntb\@tempcnta
12910 \divide\@tempcntb3\relax
12911 \@tempcnta=\numexpr\@tempcnta-\@tempcntb*3
12912 \edef\gmu@thou@put{\@xa{\gmu@thou@put}\unexpanded{#1}%
12913 \ifx\gmu@thou@putter#2\else
12914 \ifcase\@tempcnta
12915 \gmu@thou@o\or\gmu@thou@i\or\gmu@thou@ii% all three CSes are
                    yet \relax so we may put them in an \edef safely.
12918 \fi
12919 \fi}% of \edef
12920 \ifx\gmu@thou@putter #2% if we are at end of the digits...
12921 \edef\gmu@tempa{%
12922 \ifcase\@tempcnta
12923 \gmu@thou@o\or\gmu@thou@i\or\gmu@thou@ii
12924 \fi}%
12925 \@xa\let\gmu@tempa\gmu@thousep% ... we set the proper CS...
12926 \else% or ...
12927 \afterfi{% iterate.
12928 \gmu@thou@putter#2}% of \afterfi
12929 \fi% of if end of digits.
12930 }% of \gmu@thou@putter.
\gmu@thousep 12933 \def\gmu@thousep{\,% in Polish the recommended thousand separator is a thin
                    space.

```

So you can type `\thousep{7123123123123}` to get 7 123 123 123 123. But what if you want to apply `\thousep` to a count register or a `\numexpr`? You should write one or two `\expandafters` and `\the`. Let's do it only once for all:

```
12941 \pdef\xathousep#1{\@xa\thousep\@xa{\the#1}}
```

Now write `\xathousep{\numexpr 10*9*8*7*6*120}` to get 3 628 800.

```

\shortthousep 12945 \def\shortthousep{%
\thous 12946 \DeclareCommand\thous{
12947 D {\NoValue} % decimal argument
12948 }{%

```

we declare it as a command with Q-type argument to allow spaces between digits.

```

12952 \ifmmode\hbox\bgroup\@gmu@mmhboxtrue\fi
12953 \gmuIfValueTF{##1}{% we are given a sequence of digits
12954 \@tempcnta=##1\relax
12955 \ifnum\@tempcnta<0 $-$$%
12956 \@tempcnta=-\@tempcnta
12957 \fi
12958 \xathousep\@tempcnta
12959 \if@gmu@mmhbox\egroup
12960 \else\@xa\spifletter
12961 \fi
12962 }%

```



```

12963     {% no bare digits given, then we assume the argument is braced.
12964     \thousep
12965     }%
12966 }% of \thous.
12967 }% of \shortthousep.

```

And now write `\thous 3628800` to get 3 628 800 even with a blank space (beware of the range of T<sub>E</sub>X's counts).

### Footnotes suggested by Andrzej Tomaszewski

```

\ATfootnotes 12976 \DeclareCommand\ATfootnotes{s}{%
    We make the footnote mark in the footnote \scriptsize not \scriptscriptsize.
12982     \gmuIfValueT{#1}% the following setting is suitable for old style numbers in foot-
        note marks, therefore I place it in the starred version of the command.
12985     {\prependtomacro\gmu@ATfootnotes{%
12986         \pdef\@makefnmark{%
12987             \mbox {\normalfont \textsuperscript {\smaller[3]\@thefnmark
                }}}%
12988     }% of prepend,
12989 }% of \gmuIfValueT.

12991     \gmu@ATfootnotes
12992     \gmu@AT@ampulex\maketitle% without hyperref
12993     \unless\if@HyOrg@maketitle@ampulex@
12994         \ifdefined\HyOrg@maketitle
12995             \afterfi{\gmu@AT@ampulex\HyOrg@maketitle}% with hyperref
12996         \fi
12997         \@HyOrg@maketitle@ampulex@true
12998     \fi
12999 }

yOrg@maketitle@ampulex@ 13001 \newif\if@HyOrg@maketitle@ampulex@

13003 \pdef\gmu@AT@ampulex #1{%
13007     \ampulexdef#1{\def\@makefnmark}%
13008     \if@twocolumn
13009     {\gmu@ATfootnotes\if@twocolumn}% Ampulex redefinition of \maketitle for the
        standard classes.
\@makefntext 13011     \ampulexdef#1{\long\def\@makefntext}%
13012     \if@twocolumn{\gmu@ATfootnotes\if@twocolumn}% Ampulex redefinition of \maketi|
        tle for mwcls.

13014 }

13016 \pdef\gmu@ATfootnotes{%
    And we make the footnote number not be in superscript but on the base line, according
    to Andrzej Tomaszewski's suggestion on BachoTEX 2008, and the same size as in the
    footnote mark.
13020     \long\pdef\@makefntext##1{%
13021         \ifdefined\@parindent \parindent \@parindent
13022         \else \parindent 1em\relax
13023         \fi
13024         \indent{\ATf@font\scriptsize%
            {\@thefnmark}}%
13025         \gmu@fnhook
13026

```

```

13027 \enspace\ignorespaces##1}%
13028 }
13030 \let\ATf@font\normalfont
13032 \emptify\gmu@fnhook

```

### Only this paragraph

```

13037 \pdef\rrthis{% 'rag right this': make only the current paragraph ragged right (e.g.
        if the paragraph consists of a long URL).
13040 \begingroup\rightskip=0sp plus \hsize \endgraf\endgroup}
13042 \pdef\centerthis{% 2009/12/15
13043 \begingroup
13044 \rightskip=1\rightskip plus \hsize
13045 \leftskip=1\leftskip plus \hsize
13046 \parfillskip=\z@skip
13047 \endgraf\endgroup}

```

### Conditional tilde

Polish typesetting standards say that for *czcionki* (font size) 12 dd and 10 dd, if leading is narrower than  $3\frac{1}{2}$  *kwadratu*,  $3\frac{1}{2}\times 48$  dd, then hanging letters are allowed, which also applies to *czcionki* 8 and 6 dd in less than 3 *kwadraty* leading. I take this recommendation not strictly but as an inspiration, i.e., I translate »dd« to »pt«.

```

\TrzaskaTilde 13061 \def\TrzaskaTilde{%
13062 \@xa\DeclareCommand\@xa\gmu@smarttilde
13063 \@xa{\@xa T\@xa{\all@stars~}}{%
13064 \gmuIfValueTF{##1}{\nobreakspace{}}{%
13065 {\ifdim\dimexpr\hsize-\leftskip-\rightskip
13066 -\ifdim\hangindent<\z@-\fi\hangindent % the last parameter is used with
        respect to the floatflt package.
13068 >%
13069 \ifdim\f@size pt>\dimexpr10pt-1sp\relax
13070 168dd
13071 \else
13072 144dd
13073 \fi
13074 \nobreakspace {}}%
13075 \else
13076 \ %
13077 \fi
13078 }% of \gmu@ifstar's else,
13079 }% of \gmu@smarttilde,
13080 \let~\gmu@smarttilde
13081 }% of \TrzaskaTilde.

```

### A really empty page

Copied from Marcin Woliński's macros.

```

\clearemptydoublepage 13088 \newcommand{\clearemptydoublepage}{%

```

```

13089         \newpage{\pagestyle{empty}\cleardoublepage}}
13092 \foone\obeylines{%
\disobeylines 13093   \def\disobeylines{% for arguments in which line end is active to simulate normal
                    behaviour
13095     \ifnum\catcode`^^M=\active%
13096     \pdef^^M{\@ifnextgroup{\ifhmode\unskip\space\fi}}%
                    \gmu@disMinner}}%
\gmu@disMinner 13097   \def\gmu@disMinner##1{%
13098     \ifx^^M##1\endgraf%
13099     \else\afterfi{\ifhmode\unskip\space\fi}\fi##1}%
13100   \fi}%
13101 }

```

The `\star` CS and `active` `*` should be defined different to make them distinguishable by tests, especially with `\gmu@ifstar` in mind.

```

13111 \DeclareCommand\star{Q{0123456789}{1}}
13118 {\gmu@flexhyphen
13119   \gmu@star@loop0{#1}\relax
13120 }%
\gmu@star@loop 13123 \def\gmu@star@loop#1#2{% this is an expandable loop as in The  $\epsilon$ -TeX Manual p. 9.
13125   \ifnum#1<\numexpr#2\relax%
13126   \gmu@lowstar
13127   \gmu@flexhyphen
13128   \@xa\gmu@star@loop
13129   \@xa{\number\numexpr#1+1\@xa}%
13130   \@xa{\number#2\@xa}%
13131   \fi}
13134 \ifdefined\XeTeXversion
13139 \foone{%
13140   \catcode`„\active
13141   \catcode`" \active
13142   \catcode`'\active
13143   \catcode`—\active
13144   \catcode`- \active
13145 }{%
\activequotes 13146 \def\activequotes{%
13147   \incsdef„{}{\@ifnextchar—%
13148     {\lv\llap{\string„}\pauzadial}{\string„}}%
13149   \incsdef"{}{\string" \@ifnextanyRS{.,}{\quotkern}}}%
13150   \incsdef'{}{\string' \@ifnextanyRS{.,}{\quotkern}}}%
13151   \catcode`„\active
13152   \catcode`" \active
13153   }%
\activepunctsQ 13154 \def\activepunctsQ {„ "' —}%
13155 }

```

(Cyrillic) iotified e. The special delimiter that will be lost.

```

13159 \catcode`€\active
defined later, here for proper catcode.

```

```

\ac 13163 \DeclareCommand\ac {b} {%

```

```

13164 \gmuIfValueTF{#1}%
13165 {%
13166   \acro{#1}%
13167 }%
13168 {\ac@u}%
13169 }

\ac@kernel 13171 \def\ac@kernel#1{{\gmu@acrokernel {#1}}}% #1 in braces because \acrocore
           valid in Dzienniczek uses \lowercase (that requires braced text).

           Delimiters of until-iterating arguments
           They don't contain space(s)!

13179 \@xa\def\@xa\@dc@basicdelims\@xa{%
13180   \two@Ms \par \relax
13181   \bgroup
13182   \egroup \begingroup \endgroup
13183   \begin \end
13184   $%
13185   \(\)\[\]% other math delims
13186   &\% tabular delims
13187 }

           Now they do:

13190 \let\@dc@basicdelimsp\@dc@basicdelims

13192 \@xa\addtomacro\@xa\@dc@basicdelimsp
13193 \@xa{\all@spaces}

13195 \@xa\def\@xa\@dc@punctanddelims
13196 \@xa{\@dc@basicdelims : , . ; ? ! " « » < > " ' ` ' }%

13198 \let \@dc@acpunctanddelims \@dc@punctanddelims

13200 \@xa\addtomacro\@xa\@dc@acpunctanddelims
13201 \@xa{\activepunctsQ}

           now, it's the list containing spaces

13204 \let \@dc@acpunctanddelimsp \@dc@punctanddelims

13206 \@xa\addtomacro\@xa\@dc@acpunctanddelimsp
13207 \@xa{\all@spaces}

13209 \addtomacro\@dc@acpunctanddelimsp{~}

           acro iterating until

\ac@u 13212 \DeclareCommand\ac@u{
13213   >\@xa U {\@dc@acpunctanddelimsp () [] <> \K} \default{}
13214   \eachcr {\ac@kernel}
13215   >iW{\K}
13216 }
13217 {%
13218   \ifmmode\@xa\text\else\@xa\firstofone\fi{#1}%
13219 }

13221 \fi % of if XYTeX of l. 13134.

```

## enumerate\* and itemize\*

We wish the starred version of `enumerate` to be just numbered paragraphs. But `hyperref` redefines `\item` so we should do it a smart way, to set the L<sup>A</sup>T<sub>E</sub>X's `list` parameters that is.

(Marcin Woliński in `mwcls` defines those environments slightly different: his item labels are indented, mine are not; his subsequent paragraphs of an item are not indented, mine are.)

```
enumerate* 13236 \@namedef{enumerate*}{%
13237   \ifnum\@enumdepth>\thr@@
13238     \@toodeep
13239   \else
13240     \advance\@enumdepth\@ne
13241     \edef\@enumctr{enum\romannumeral\the\@enumdepth}%
13242     \@xa\list\csize label\@enumctr\endcsize{%
13243       \partopsep\topsep \topsep\z@ \leftmargin\z@
13244       \itemindent\@parindent % \advance\itemindent\labelsep
13245       \labelwidth\@parindent
13246       \advance\labelwidth-\labelsep
13247       \listparindent\@parindent
13248       \usecounter \@enumctr
13249       \def\makelabel##1{##1\hfil}}%
13250   \fi}
13251 \@namedef{endenumerate*}{\endlist}

itemize* 13254 \@namedef{itemize*}{%
13255   \ifnum\@itemdepth>\thr@@
13256     \@toodeep
13257   \else
13258     \advance\@itemdepth\@ne
13259     \edef\@itemitem{labelitem\romannumeral\the\@itemdepth}%
13260     \@xa\list\csize\@itemitem\endcsize{%
13261       \partopsep\topsep \topsep\z@ \leftmargin\z@
13262       \itemindent\@parindent
13263       \labelwidth\@parindent
13264       \advance\labelwidth-\labelsep
13265       \listparindent\@parindent
13266       \def\makelabel##1{##1\hfil }}%
13267   \fi}
13268 \@namedef{enditemize*}{\endlist}

13271 </typos>
```

## The `gmparts` package—in/exclusion of parts of one file analogous to `\include`

```
13278 <utils> \gmu@PackOptionX{parts}
13279 <*parts>

13281 \RequirePackage{gmcommand}
```

## `\include not only .tex's`

`\include` modified by me below lets you to include files of any extension provided that extension in the argument.

If you want to `\include` a non-`.tex` file and deal with it with `\includeonly`, give the latter command full file name, with the extension that is.

```
\gmu@gettext 13291 \def\gmu@gettext#1.#2\@nil{%
13292   \def\gmu@filename{#1}%
13293   \def\gmu@fileext{#2}}

13295 \def\include#1{\relax
13296   \ifnum\@auxout=\@partaux
13297   \@latex@error{\string\include\space cannot be nested}\@eha
13298   \else \@include#1 \fi}

13300 \def\@include#1 {%
13301   \gmu@gettext#1.\@nil
13303   \ifx\gmu@fileext\empty\def\gmu@fileext{tex}\fi
13304   \clearpage
13305   \if@filesw
13306     \immediate\write\@mainaux{\string\@input{\gmu@filename.aux}}%
13307   \fi
13308   \@tempwattrue
13309   \if@partsw
13310     \@tempwafalse
13311     \edef\reserved@a{#1}%
13312     \@for\reserved@a:=\@partlist\do{%
13313       \ifx\reserved@a\reserved@b\@tempwattrue\fi}%
13314   \fi
13315   \if@tempswa
13316     \let\@auxout\@partaux
13317     \if@filesw
13318       \immediate\openout\@partaux \gmu@filename.aux
13319       \immediate\write\@partaux{\relax}%
13320     \fi
13321     \@input@{\gmu@filename.\gmu@fileext}%
13322     \inclasthook
13323     \clearpage
13324     \@writeckpt{\gmu@filename}%
13325     \if@filesw
13326       \immediate\closeout\@partaux
13327     \fi
13328   \else

13332     \deadcycles\z@
13333     \@nameuse{cp@\gmu@filename}%
13334   \fi
13335   \let\@auxout\@mainaux
13336 }

\whenonly 13339 \newcommand\whenonly[3]{%
13340   \def\gmu@whonly{#1,}%
```

```
13341 \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.

```
13345 \def\inclasthook{\thispagestyle{closing}}
```

### Switching on and off parts of one file

The `\include` facility is very nice only it forces you to split your source in many files. Therefore I provide a tool analogous to `\include` and using the same `\includeonly` mechanism/list to switch on and off parts of the same source file.

```
13354 \def\filepart#1{\relax
13355 \ifnum\@auxout=\@partaux
13356 \@latex@error{\string\filepart\space cannot be nested}\@eha
13357 \else\afterfi{\@filepart#1 }\fi}

13359 \def\@filepart#1 {%
13360 \clearpage
13361 \edef\gmu@filepartname{#1}% we'll use it later
13362 \if@filesw
13363 \immediate\write\@mainaux{\string\@input{#1.aux}}%
13364 \fi
13365 \@tempswattrue
13366 \if@partsw
13367 \@tempswafalse
13368 \@for\gmu@filepart@resa:=\@partlist\do{%
13369 \ifx\gmu@filepart@resa\gmu@filepartname\@tempswattrue\fi}%
13370 \fi
13371 \if@tempswa
13372 \let\@auxout\@partaux
13373 \if@filesw
13374 \immediate\openout\@partaux #1.aux
13375 \immediate\write\@partaux{\relax}%
13376 \fi
13377 \@xa\@firstoftwo
13379 \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.

```
13383 \deadcycles\z@
13384 \@nameuse{cp@\gmu@filepartname}%
13385 \let\@auxout\@mainaux
13386 \@xa\@secondoftwo
13387 \fi
13388 {\iftrue}%
13389 {\let\endfilepart\fi
13390 \csname gm@skipped@#1\endcsname
13391 \def\next{\Restore@MacroSt {endfilepart}}%
13392 \@ifnextchar\bgroup{\show\NextBgroup\@gobble}{}}%
13393 \@xa\next\iffalse}%
13394 }
```

```
\endfilepart 13397 \DeclareCommand\endfilepart{b}{% Note the argument is not used really. Maybe
later we'll use it for checking of proper matching. Or maybe not.
```

```
13399 \inclasthook
```

```

13400 \clearpage
13401 \@writeckpt{\gmu@filepartname}%
13402 \if@filesw
13403 \immediate\closeout\@partaux
13404 \fi
13405 \fi% this \fi closes \Iftrue put by line 13377.
13406 \let\@auxout\@mainaux
13407 }

13409 \Store@Macro\endfilepart

13411 \def\nofileparts{%
13412 \let\filepart\@gobble
\endfilepart 13413 \DeclareCommand\endfilepart{b}{}%
13414 }

```

### Fix of including when fontspec is used

The `fontspec` package creates counters for font families. If a `fontspec` command is used in a part of a document and then such a part is skipped, an error occurs ‘No counter `zf@fam@...` defined’. Now we fix that by ensuring all the counters are defined before they are set.

Note it’s a draft version which doesn’t support resetting of one counter within another.

```

13426 \def\includecountfix{%
13427 \def\@wckptelt##1{%
13428 \immediate\write\@partaux{%
13429 \providecounter{##1}% to provide the font counters defined in parts of the
document.
13432 \string\setcounter{##1}{\the\@nameuse{c@##1}}}%
13433 }

13435 \pdef\providecounter#1{%
#1 13436 \unless\ifcsname c@#1\endcsname\newcounter{#1}\fi}
13438 \</parts>

```

### The gmurl package

```

13445 <utils> \gmu@PackOptionX{url}
13446 <*url>

13448 \RequirePackage{gmcommand}

```

### hyperref’s \nolinkurl into \url\*

```

13452 \def\urlladdstar{%
13453 \AtBeginDocument{%
13454 \@ifpackageloaded{hyperref}{%
13455 \Store@Macro\url
13456 \pdef\url{\gmu@ifstar{\nolinkurl}{\storedcsname{url}}}%
13457 }{}}
13459 \@onlypreamble\urlladdstar

```



## A fix to the url package

It happened that a URLs typeset with the `\url` command of the `url` package came out sort of spaced because kerning was off because of the math mode. So I provide a redefinition of the internal macros of the `url` package which in my version uses not math mode but `\scantokens` and not `\relpenalty` and `\binoppenalty` but `\hyphenpenalty` (as it is in the paragraph) and `\discretionary`. I tried putting explicit penalties after the symbols but that spoiled kerning.

The rules of line breaking are somewhat different, too: in the original `url` package line breaks are forbidden between any two symbols listed in `\UrlBigBreaks`. In my version line breaks are forbidden between any two *identical* ‘URL Breaks’ and ‘URL Big Breaks’.

There are some more differences in formatting some chars, i.a. `~`, `%` and angle brackets which I don’t treat specially and just take from font assuming the font provides ASCII chars and checking whether it provides the angle brackets.

```
13484 \@ifXeTeX{%
13485   \pdef\UrlFix{\AtBeginDocument{%
13486     \@ifpackageloaded{url}{\gmu@UrlFix}}}%
13487   \relaxen\UrlFix}%
13489 \AtBeginDocument{%
13490   \pdef\UrlFix{%
13491     \@ifpackageloaded{url}{\gmu@UrlFix}}}%
13492   \relaxen\UrlFix}}%
13493 }
13494 {%
13495   \pdef\UrlFix{\PackageWarning{gmutils}{!!! The \string\UrlFix\space
13496     declaration works only with XeTeX}}%
13497 }

13500 \@ifXeTeX{}{%
13501   \edef\gmu@restoreUpUpUp{\catcode`\@nx\^^^=\the\catcode`\^^^}%
13502   \AtEndOfPackage\gmu@restoreUpUpUp
13503   \catcode`\^^^=9 }

13505 \def\gmu@UrlFix{%
  default style assignments

13508   \def\UrlBreaks{\do\.\do\@\do\\\do\/\do!\do\_do\|\do\;\do\}%
13509   \do\)\do\,\do\?\do\'\do\"\do\+\do\=\do\#\do\%\do\~\do\_do\|\do\|}%
13510   \do\{\do\}\do\$\}%
13511   \def\UrlBigBreaks{\do\:}%
13512   \def\UrlNoBreaks{\do\(\do\[\do\{}%
13513   \def\UrlSpecials{%
13514     \do\ {\hbox{\visiblespace}}\do\^M{\hbox{\visiblespace}}}%

13518   \def\Url@Format##1{%
13519     \UrlFont
13520     \ifdefined\verbatim@specials
13521       \catcode`\>\active
13522       \verbatim@specials
13523       \verbatim@mathhack
13524     \fi % setting of the escape char, begin and end group and optionally math shift,
           defined in gmverb.

13527   \gmu@UrlSetup
13528   \UrlLeft
```

```

13529 \edef\gmu@theendlinechar{\the\endlinechar}%
13530 \endlinechar\m@ne
13531 \kern\z@% to forbid hyphenating the first word if the URL begins with a word
13533 \hyphenchar\font=\UrlHyphenchar\relax
13534 \let-\gmu@discretionaryhyphen
13535 \scantokens{##1}%
13536 \endlinechar\gmu@theendlinechar\relax
13537 \UrlRight
13538 }% of \Url@Format.

13540 \edef\UrlHyphenchar{%
13541 \ifdefined\gmv@hyphenchar\gmv@hyphenchar
13542 \else"A6 \fi}% broken bar, | or the same as provided in gmverb for verbatims.
    You can redefine it as you please. This char is used as the hyphenation char
    in URLs and therefore should be different from - (hyphen), which is often a
    part of an URL. The broken bar seems to be quite unlikely in URLs and/or
    file names.

13550 \def\verbatim@mathhack{%
13551 \ifdefined\verbatim@specials@list
13552 \@xa\verbatim@mathhack@\verbatim@specials@list
13553 \fi
13554 }%

13556 \def\verbatim@mathhack@##1##2##3##4##5##6{%
13557 \gmuIfValueT{##4}{%
13558 \edef\gmu@thinmuskip{\the\thinmuskip}%
13559 \edef\gmu@medmuskip{\the\medmuskip}%
13560 \edef\gmu@thickmuskip{\the\thickmuskip}%
13561 \begingroup
13562 \lccode`~=#4\lowercase{%
13563 \endgroup\def~###1~}%
13564 {$\thinmuskip\gmu@thinmuskip\relax
13565 \medmuskip\gmu@medmuskip\relax
13566 \thickmuskip\gmu@thickmuskip\relax
13567 ###1%
13568 $}%
13569 \catcode`##4\active
13570 }%
13571 }%

13573 \def\gmu@UrlSetup{%
13574 \medmuskip\Urlmuskip \thickmuskip\medmuskip \thinmuskip0mu%
13575 \relpenalty\UrlBigBreakPenalty \binoppenalty\UrlBreakPenalty
13576 \def\do{\gmu@doUrlMath\UrlBreakPenalty}\UrlBreaks % bin ( \hyphenpenalty
    anyway)
13578 \def\do{\gmu@doUrlMath\UrlBigBreakPenalty}\UrlBigBreaks % rel ( \hyphen|
    penalty anyway)
13580 \def\do{\gmu@doUrlMath@M}\UrlNoBreaks % open (no break)
13581 \def\do{\gmu@doUrlMathAc\UrlBreakPenalty}% ( \hyphenpenalty)
13582 \UrlSpecials
13583 \if \iffontchar\font"2329 1\else0\fi\iffontchar\font"232A
    1\else2\fi
    we check whether the font provides both left and right angle brackets.
13586 \gmu@measurewd{^^^2329}%

```

```

13587 \edef\gmu@tempa{%
13588 \@nx\gmu@doUrlMathAc\@M\@nx\<{%
13589 \hbox to\gmu@tempb{\unexpanded{\hss\char"2329 \hss}}}%
13590 }\gmu@tempa
13591 \gmu@measurewd{^^^232a}%
13592 \edef\gmu@tempa{%
13593 \@nx\do\@nx\>{%
13594 \hbox to\gmu@tempb{\unexpanded{\hss\char"232A \hss}}}%
13595 }\gmu@tempa
13596 \else
13597 \gmu@doUrlMathAc\@M\<{\langle}\do\>{\rangle}%
13598 \fi
13599 \iffontchar\font"22C6 % low star
13600 \do\*{\hbox{\char"22C6 }}%
13601 \else \do\**%
13602 \fi
13603 \ifx\do@url@hyp\@empty
13604 \gmu@measurewd{-}% this macro is defined in line 3849.
13605 \edef\gmu@tempa{%
13606 \unexpanded{\gmu@doUrlMathAc\@M\-%
13607 {\hbox to \gmu@tempb{\unexpanded{\hss-\hss}}}%
13608 \@nx\-% hyphen is a good point for hyphenation, but the hyphenation
char should be sth. else, and it is indeed: | (broken bar, \char"A6).
See also line 13542
13612 }\gmu@tempa
13613 \fi
13614 \addfontfeature{Ligatures=NoCommon, Mapping=none}% instead of 'doing' \ver|
% batim@nolig@list.
13616 }% of \gmu@UrlSetup.
13620 \def\gmu@doUrlMath##1##2{%
% #1 value of the penalty (used as a Boolean: if < 10 000,
% \hyphenpenalty will be used anyway, if ≥ 10 000, there will be no \dis|
cretionary),
% #2 the char, given as \langle char\rangle.
13627 \begingroup
13628 \lccode\~=#2\lowercase{%
13629 \endgroup\def~{\@ifnextchar~}%
13630 \@xa\addtomacro\@xa~}% of \lowercase.
13631 \ifnum##1<\@M
13632 {%
13633 {\char`##2\csname gmu@dbl\string##2kern\endcsname}% if next is the same
char
13634 {\ifmmode\char`##2% else
13635 \else\gmu@urlbreakable{##1}{##2}%
13636 \fi}%
13637 }% of \addtomacro's argument \ifnum true.
13638 \else
13639 {%
13640 {\char`##2\csname gmu@dbl\string##2kern\endcsname}{\char`##2}%
13641 }% of \addtomacro's argument \ifnum false.
13642 \fi
13643 \catcode`##2=\active

```

```

13644 }% of \gmu@doUrlMath.
13646 \def\gmu@doUrlMathAc##1##2##3{%
      % #1 (value of) a penalty (see the remark to ##1 of the previous macro),
      % #2 the char (as \langle char\rangle),
      % #3 the definition.
13653 \begingroup
13654 \lccode\~=`##2\lowercase{%
13655 \endgroup\def~{\@ifnextchar~}%
13656 \@xa\addtomacro\@xa~}% of \lowercase.
13657 \ifnum ##1<\@M
13658 {%
13659 {\ifmmode\char`##2\else$##3\m@th$\fi}%
13660 {\ifmmode\char`##2%
13661 \else\discretionary{\hbox{$##3\m@th$}}{\hbox{$##3\m@th$}}%
13662 \fi}%
13663 }% of \addtomacro's argument if num true.
13664 \else
13665 {%
13666 {\ifmmode\char`##2\else$##3\m@th$\fi}{%
      \ifmmode\char`##2\else$##3\m@th$\fi}%
13667 }% of \addtomacro's argument if num false.
13668 \fi
13669 \catcode`##2=\active
13670 }% of \gmu@doUrlMathAc.
13672 \pdef\gmu@url@rigidbreak##1##2{\discretionary{\char`##2}}{\char`##2}}%
13674 \pdef\gmu@url@flexbreak##1##2{\penalty\@M \hskip\z@ plus0,03em
13675 \char`##2\penalty##1\hskip\z@ plus0,03em\relax}%
13677 \let\gmu@urlbreakable\gmu@url@flexbreak
13679 \def\Url@z##1{%

```

Do any hyper referencing due to hyperref (or perform a url-def)

```

13681 \Url@HyperHook

```

Now do the formatting in a group (can also have \Url@HyperHook take this as an argument).

```

13684 {\Url@Format{##1}}%
13685 \endgroup}%

```

```

13687 \DeclareUrlCommand\file{\urlstyle{sf}}%

```

\emptify\Url@moving% with our settings \url is pretty allowed in moving arguments, I hope.

```

13691 }% of \gmu@UrlFix.

```

```

\UrlSlashKern 13693 \DeclareCommand\UrlSlashKern{O{tt}m}%
13694 {\AtBeginDocument{%
13695 \nameedef{url@#1style}{\def\@nx\UrlFont{%
13696 \@xanxcs{#1family}%
13697 \def\@xanxcs{gmu@dbl\string\/kern}%
13698 {\kern#2\relax}}%
13699 }% of \UrlFont

```

```

13700     }% of \url#1style
13701     \urlstyle{#1}%
13702     }% of \AtBeginDocument
13703 }% of \UrlSlashKern

13707 \def\DeclareUrlCommand#1#2{\pdef#1{\leavevmode\begingroup #2\Url}}
      %% [#1][#1]{\def}{#1}{\pdef#1}

13710 \foolc ~ : {%
13711   \ifXeTeX{%
13712     \def\metaat~{%
13713       \penalty\@M \hskip\z@skip
13714       \meta{at}% it's a Cyrillic »a«!
13715       \penalty\exhyphenpenalty
13716       \hskip\z@skip
13717     }%
13719     \def\metadot~{%
13720       \penalty\@M \hskip\z@skip
13721       \meta{dot}% it's a Cyrillic »o«!
13722       \penalty\exhyphenpenalty
13723       \hskip\z@skip
13724     }%
13725   }% of if XeTeX
13726   {%
13727     \def\metaat~{\PackageError{gmurl}{Command \backslash metaat
13728       works only in XeTeX}{}%
13730     \def\metadot~{\PackageError{gmurl}{Command \backslash metaat
13731       works only in XeTeX}{}%
13732   }% of if not XeTeX
13733 }% of \foolc

13735 </url>

```

## The gmRCS package

```

13741 <utils>     \gmu@PackOptionX{RCS}
13742 <*RCS>

13744 \def\ParseRCSVersion$Id%
13745 : #1.#2,v #3 #4 #5 #6 #7${%
13746   \def\RCSFileName{#1}%
13747   \def\RCSFileExt{#2}%
13748   \def\RCSFile{#1.#2}%
13749   \def\RCSVersion{#3}%
13750   \def\RCSDate{#4}%
13751   \def\RCSTime{#5}%
13752   \def\RCSAuthor{#6}%
13754 }

```

And the expandable version of those above (when we only one datum at one place)

```

13759 \def\defRCSeDatum
13760 #1% datum name

```

```

13761 #2% datum definition (basically #number)
13762 {%
13763   \@namedef{eRCS#1}$Id%
13764   : ##1.##2,v ##3 ##4 ##5 ##6 ##7${#2}%
13765 }

13767 \defRCSDatum{File}{#1.#2} % \eRCSFile
13768 \defRCSDatum{Version}{#3}% \eRCSVersion
13769 \defRCSDatum{Date}{#4}% \eRCSVDate
13770 \defRCSDatum{Time}{#5}% \eRCSVTime
13771 \defRCSDatum{Author}{#6}% \eRCSVAuthor

13773 </RCS>

```

## Back to gmutils

```

13777 <*utils>

13779 \ExecuteOptionsX{command, envir, ampulex, relsize, meta, logos,
13780   notonlypream, %
13781   %% mw=off,
13782   typos, parts, url}

13783 \ProcessOptionsX

13786 \def\doifdefined#1{\ifdefined#1\@xa#1\fi}

13788 \doifdefined\gmu@Require@command
13789 \doifdefined\gmu@Require@envir
13790 \doifdefined\gmu@Require@ampulex
13791 \doifdefined\gmu@Require@relsize
13792 \doifdefined\gmu@Require@meta
13793 \doifdefined\gmu@Require@logos
13794 \doifdefined\gmu@Require@notonlypream
13795 \doifdefined\gmu@Require@mw
13796 \doifdefined\gmu@Require@typos
13797 \doifdefined\gmu@Require@parts
13798 \doifdefined\gmu@Require@url
13799 \doifdefined\gmu@Require@RCS

```

## Third person pronouns

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

Previous versions of this documentation were consequently alternating ‘he’ and ‘she’ and provided specific macros for that purpose. Now I’m not that queer-and-gender so I take what is normally used these days, ‘they’ that is.

(The issue of human sexes and genders (certainly much more numerous than 2) is complex and delicate and a  $\TeX$  macro package is probably not the best place to discuss it.)

```

13816 \def\heshe{they}
13817 \def\hisher{their}

```

```

13818 \def\himher{them}
13819 \def\hishers{theirs}

13821 \def\HeShe{They}
13822 \def\HisHer{Their}
13823 \def\HimHer{Them}
13824 \def\HisHers{Theirs}

13893 </utils>
13896 \endinput

```

End of file ‘gmutils.gmd’.

□

## Change History

<p><b>gmampulex v0.93</b>  <b>\ampulexlet:</b>  added, 9122</p> <p><b>gmampulex v0.94</b>  <b>\ampulexlet:</b>  made <b>xparse</b>-ish and <b>\ampulexset</b>  removed, 9122</p> <p><b>gmampulex v0.98</b>  <b>\ampulexlet:</b>  first argument (the prefix) made of the  <math>\mathcal{Q}</math> type, 9122</p> <p><b>gmampulex v0.994</b>  <b>\ampulexdef:</b>  rewritten not to use the arguments 7–9  explicitly not to wrap them in a  temporary macros, and to accept  single hashes (instead of quadruple) in  arguments 5 and 6. The temporary  macros renamed to <b>\gmu@reserveda</b>  and <b>\gmu@reservedb</b>. The body  moved to <b>\ampulexlet</b> with adding  another CS argument to let let a  macro not necessarily redefine it itself  and this command made a particular  case of that new one, 9220</p> <p><b>\ampulexlet:</b>  added as a more general version of  <b>\ampulexdef</b> (which now is a  particular use of this command), 9104</p> <p><b>gmbase v0.75</b>  <b>\@ifnif:</b>  added, 2809</p> <p><b>\@xifncat:</b>  <b>\let</b> for <b>#1</b> changed to <b>\def</b> to allow  things like <b>\noexpand~</b>, 2769</p> <p><b>\xiibackslash:</b>  <b>\let</b> for <b>#1</b> changed to <b>\def</b> to allow  things like <b>\noexpand~</b>, 2729</p> <p><b>gmbase v0.88</b></p>	<p><b>\ResetMacros:</b>  added, 3445, 3455, 3459</p> <p><b>gmbase v0.92</b>  <b>\@gif:</b>  added redefinition so that now switches  defined with it are <b>\protected</b> so  they won’t expand to a further  expanding or unbalanced  <b>\iftrue/false</b> in an <b>\edef</b>, 2304</p> <p><b>\@parindent:</b>  added, 3883</p> <p><b>gmbase v0.93</b>  <b>\@gobbleeight:</b>  added, 981, 1010</p> <p><b>\pdef:</b>  added, 935</p> <p><b>\pprovide:</b>  added, 997</p> <p><b>gmbase v0.94</b>  <b>\@parindent:</b>  <b>\includegraphics</b> works well in  <math>\text{\LaTeX}</math> so I remove the complicated  version with <b>\XeTeXpicfile</b>, 3792</p> <p><b>\@xau:</b>  added, 847</p> <p><b>\addto@forlist:</b>  added. All <b>\@ifundefineds</b> used by  me changed to this, 2492  made robust to unbalanced <b>\ifs</b> and  <b>\fis</b> the same way as <math>\text{\LaTeX}</math>’s  <b>\@ifundefined</b> (after a heavy debug  :-), 2492</p> <p><b>\addtomacro:</b>  order of arguments reversed, 2416</p> <p><b>\ifgmuXeTeX:</b>  the <math>\text{\LaTeX}</math> version enriched with  <b>\iffontchar</b> due to lack of bullets  with the default settings reported by  Morten Høgholm and Edd Barrett, 3617</p>
--	--

`\newdef:`  
`\if` removed from parameters' string, 1334  
**gmbase v0.96**  
`\ifgmuXeTeX:`  
moved here from my private document class, 3622  
**gmbase v0.97**  
`\newdef:`  
added, 1298  
**gmbase v0.98**  
`\@XAtoks:`  
moved here from `gmdoc` and rewritten, including split to `\IfAmong` because the latter is needed in `\DeclareCommand`, 1804  
`\@ifnif:`  
added test for `\egroup`, 2894  
extracted from `\@ifnextac`, 2889  
renamed from `\@ifstar` since something redefines `\@ifstar`, 2868  
`\@parindent:`  
added, 3760  
moved here from my personal macro package since I needed it in the documentation of `gmutils`, 3754  
rewritten not to make entries in the hash table, thanks to `\detokenize` and made robust to open `\ifs` in the arguments thanks to substitution of explicit parameters with `\@firstoftwo` and `\@secondoftwo`, 3953  
`\g@relaxen:`  
split from `\IfAmong`, 1438  
`\newdef:`  
renamed from `\@secondofmany`, 1298  
**gmbase v0.99**  
`\@parindent:`  
moved to a separate macro from `\@ifenvir` and made symmetric to both arguments, 3953  
**gmbase v0.991**  
`\@xau:`  
made 1-parameter, 847  
`\@xifncat:`  
inner macro fixed to handle active chars more properly (more as `\if` would do it), 2769  
**gmbase v0.992**  
`\sups :`  
added, 2578  
**gmbase v0.993**  
`\@parindent:`  
made `\protected` after an error at work, 3883  
redefined deeply and made expandable thanks to `\strcmp`. Also renamed from `\gmu@ifedetokens` (the `\gmu` prefix added), 3953  
since `\gmu@ifedetokens` turned to be expandable, we make this macro `\protected`, 3900  
General:  
package cut out from `gmutils` due to need of using `\DeclareCommand` with minimal chance of interference of other stuff, 4277  
`\RestoreMacros:`  
added, 3395  
`\StoreMacro:`  
fixed asymmetry with the unstarred version: till today the unstarred version for an undefined CS undefine the storage CS and the starred version made it `\relax`. Now both undefine, 3262  
**gmcommand v0.90**  
`\XeTeXthree:`  
adjusted to the redefinition of `\verb` in `xlxtra` 2008/07/29, 8352  
**gmcommand v0.91**  
General:  
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, 9062  
**gmcommand v0.94**  
General:  
removed `\unex@namedef` and `\unex@nameuse`, probably never really used since they were incomplete:  
`\edef@other` undefined, 9062  
The code from ancient `xparse` (1999) of `TEXLive` 2007 rewritten here, 9062  
**gmcommand v0.98**  
`\ArgumentCatcher@PLoop:`  
the `xparse` tests for the presence of value redefined and much simplified (43 CS'es less). Moreover, they are now fully expandable:  
`\gmuIfNoValueTF`, `\gmuIfNoValueT`,  
`\gmuIfNoValueF`, `\gmuIfValueTF`,  
`\gmuIfValueT`, `\gmuIfValueF`, 7463  
`\DeclareCommand:`  
added the `S/T` spec parsing, the `s` spec parsing rewritten to be a shorthand for `S{★}`, unused code removed, 5810  
`\enoughpage:`  
`\par` removed since to let it be used for `\pagebreak` inside a paragraph, 8445  
made  $\varepsilon$ -`TEX`-ish, 8445



made ifthenelse-like with ‘then’ and ‘else’ optional default *<nothing>* and `\newpage` resp., 8445

`\gmuIfValueF`:  
 added the `\@bsphack—\@esphack` option, 7604  
 added the `\Q{<tokens>}` argument type (a word over the alphabet *<tokens>*), 7604

`gmcommand` v0.993  
`\@UseStored`:  
 added, 8700

General:  
 the pseudo-argument type  
`\afterassignment` renamed to `=`, 9062

`\DeclareEnvironment`:  
 we make the `\end...` command  
`\DeclareCommand` ed for the symmetry, for simplifying definition of `\envirlet` in particular, 8246

`\gmuIfValueF`:  
 added to modify naming of the inner macro of a `\DeclareCommand` ed commands: preceding the names with a backslash is OK for the letter CS’ses, but it may cause a collision for the active chars. Therefore we both precede the name *and* succede it with a bslash, 7592

This strikingly simple idea of storing the specifiers’ sequence in a macro came to my head only after some three years of developing  
`\DeclareCommand`, 7549

`\if@dc@Mmode@`:  
 made for-eaching to make it behave as expected, i.e. that latter settings prævalent over the former, 6429

`\if@dc@xaeacher@`:  
 implemented the `\SameAs` feature, 5810

`KV@gmutils.gmd@gm#1`:  
 Incompatibilities with earlier versions: because of making all the specifiers “case-insensitive”, i.e. aliasing the uppercases as lowercase, the `S` spec. ceased to be a `Single-token-catcher` and became a `Star-token-catcher`. Moreover, the parameters for *all* the one-parameter specifiers became optional (not only for the lowercase as earlier). Moreover, catcher names of CS specifiers are now created by `\stringing` the CS and stripping its backslash (earlier: by crude detokenising it)., 5203  
 moved here from `gmutils.sty` (a bug fix), 5223

`gmenvir` v0.74

`begin*`:  
 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`, 9401

`gmenvir` v0.92  
`begin*`:  
 added, 9426  
 shortened thanks to `\@ifenvir`, 9461

`gmenvir` v0.993  
`begin*`:  
 made use `\@envirstack`, 9426  
 to be precise, it’s not a change but rather a staus quo action:  
`\gmu@ifedetokens` suddenly turned to be expandable and `un\protected` so we make *this* macro `\protected`, 9426

`gmlogos` v0.96  
`\LuaTeX`:  
 added, 10198

`gmlogos` v0.97  
`\pdfLaTeX`:  
 added, 10130

`gmlogos` v0.993  
`\LaTeXpar`:  
 added `\leavevmode` (a bug fix), 10074

`\XeTeXpar`:  
 added, 10161

`gmlogos` v0.995  
`\ConTeXt`:  
 added, 10201

`gmmeta` v0.96  
`\gmu@pswords`:  
`\textup` removed to allow slanting the name in titles (that are usually typeset in Italic font), 9709

`gmmeta` v0.98  
`\arg@dc`:  
 made iterating thanks to  
`\DeclareCommand\arg@dc`, 9951

`\cs`:  
 added `\verbatim@specials`, 9723  
 made `\-` switch to `\normalfont` because IMHO this underlines the fact that a CS belongs to the narrative.  
`\hyphenchar` set to 45 as in usual texts, 9723

`gmmeta` v0.99n  
`\cat`:  
 added, 9968, 9974, 9976

`gmnotonlypream` 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`, 10222
- gmnotonlypream v0.93**  
**\nocite:**  
 a bug fixed: with `natbib` an ‘extra `}`’ error. Now it fixes only the standard version of `\nocite`, 10270
- gmRCS v0.74**  
 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, 13824
- gmRCS 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, 13824
- gmRCS 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, 13824
- gmRCS v0.78**  
 General:  
`\@namelet` renamed to `\n@melet` to solve a conflict with the `beamer` class. The package contents regrouped, 13824
- gmRCS v0.85**  
 General:  
 fixed behaviour of too clever headings with `gmdoc` by adding an `\ifdim` test, 13824
- gmRCS v0.87**  
 General:  
 the package goes  $\epsilon$ -TeX even more, making use of `\ifdefined` and the code using UTF-8 chars is wrapped in a  $X_{\text{TeX}}$ -condition, 13824
- gmRCS v0.89**  
 General:  
 removed obsolete adjustment of `pgf` for  $X_{\text{TeX}}$ , 13824
- gmRCS v0.91**  
 General:  
 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, 13824
- gmRCS v0.93**  
 General:  
 A couple of `\DeclareRobustCommand*` changed to `\pdef`, 13824  
 The numerical macros commented out as obsolete and never really used, 13824
- gmRCS v0.94**  
 General:  
`\bgroup` and `\egroup` in the macro storing commands and in `\foone` changed to `\beginngroup` and `\endngroup` since the former produce an empty `\mathord` in math mode while the latter don’t, 13824  
 removed `\unex@namedef` and `\unex@nameuse`, probably never really used since they were incomplete:  
`\edef@other` undefined, 13824  
 The code from ancient `xparse` (1999) of TeXLive 2007 rewritten here, 13824
- gmtypos v0.76**  
**\freeze@actives:**  
 added, 11972
- gmtypos v0.80**  
**\hfillneg:**  
 added, 11936
- gmtypos v0.81**  
**\dekfracslash:**  
 moved here from `pmlectionis.cls`, 12153  
**\uscacro:**  
 moved here from `pmlectionis.cls`, 12123
- gmtypos v0.82**  
**\ikern:**  
 added, 12161
- gmtypos v0.83**  
**\gmu@tilde:**  
 postponed to `\begin{document}` to avoid overwriting by a text command and made sensible to a subsequent `/`, 11903
- gmtypos v0.86**  
**\gmu@tilde:**  
 renamed from `\~` since the latter is one of LaTeX’s accents, 11903
- gmtypos v0.93**  
**\dilitkern:**  
 added, 11625, 11650, 11659  
 copied here from E. Szarzyński’s *The Letters*, 11608, 11633  
**\gmu@RPfor:**  
 renamed from `\gmu@RPif` and `#3` changed from a csnake to CS, 12036  
**\whernup:**  
 added, 12884

**gmtypos v0.94**  
`\date@left:`  
`\leftline` replaced with `\par...\par` to work well with `floatflt`, 12636  
`\gmu@dogmathbase:`  
removed definition of `\langle letter \rangle s` and `\langle digit \rangle s`, 11388  
`\if@gmu@mmhbox:`  
made to work also in math mode, even with math-active digits, 10874  
**gmtypos v0.96**  
`\gmath:`  
added, 11501  
`\gmu@dogmathbase:`  
Greek letters completed. Wrapped with `\adddotoks` to allow using in any order with `\garamath` and others, 11388  
the `\everymath`'s left brace moved here: earlier all the stuff was put into `\everymath`, 11456  
**gmtypos v0.97**  
`\*:`  
removed (it was a text tilde, available as `\texttilde`), 11896  
**gmtypos v0.98**  
`\@makefntext:`  
added, 13037  
`\ATfootnotes:`  
moved here from my own private macro package to allow the beauty of these footnotes for the general audience, 12976  
`\fakeonum:`  
added, 11683  
`\napapierkicore:`  
added optional star controlling globalness, 12296  
**gmutils**  
`\gmFileKind:`  
Checksum 10678 , 369  
**gmutils v0.80**  
`\gmFileKind:`  
Checksum 1689 , 369  
**gmutils v0.84**  
`\gmFileKind:`  
Checksum 2684 , 369  
**gmutils v0.85**  
`\gmFileKind:`  
Checksum 2795 , 369  
**gmutils v0.87**  
`\gmFileKind:`  
Checksum 4027 , 369  
**gmutils v0.88**  
`\gmFileKind:`  
Checksum 4040 , 369  
**gmutils v0.90**  
`\gmFileKind:`  
Checksum 4035 , 369  
**gmutils v0.91**  
`\gmFileKind:`  
Checksum 4055 , 369  
**gmutils v0.92**  
`\gmFileKind:`  
Checksum 4133 , 369  
**gmutils v0.93**  
`\gmFileKind:`  
Checksum 4140 , 369  
Checksum 4501 , 369  
**gmutils v0.94**  
`\gmFileKind:`  
Checksum 4880 , 369  
**gmutils v0.95**  
`\gmFileKind:`  
Checksum 4908 , 369  
**gmutils v0.96**  
`\gmFileKind:`  
Checksum 5363 , 369  
put to CTAN on 2008/11/21, 369  
**gmutils v0.97**  
`\gmFileKind:`  
Checksum 5375 , 369  
put to CTAN on 2008/11/22, 369  
**gmutils v0.98**  
`\gmFileKind:`  
Checksum 5429 , 369  
Checksum 5577 because of `\DeclareCommand`, 369  
Checksum 5627 because of `\fakeonum`, 369  
Checksum 6035 because of `\DeclareCommand`, `\arg`, 369  
Checksum 6129 because of `\DeclareEnvironment`, 369  
Checksum 6147 because of `\arg` in verbatims, 369  
Checksum 6535 because of `\UrlFix`, 369  
Checksum 6656 because of type settings for `gmdoc` (`\narrativett`) and for `verbatimspecials`, 369  
**gmutils v0.991**  
`\gmFileKind:`  
Checksum 8257 because of some shorthands and major development of `\DeclareCommand`, including 'Knuthian' and general optional arguments and '`\afterassignment`' pseudo-argument, 369  
put to CTAN on 2010/03/04, 369  
**gmutils v0.993**  
`\gmFileKind:`  
Checksum 12497 , 369  
Checksum 12591 , 369  
Checksum 12970 , 369  
Checksum 8257 because of `gmbase` cutting out from `gmutils`, 369

Checksum 8257 because of  
**gmcommand** cut out from **gmutils**, 369  
put to CTAN on 2011/02/02, 369  
**gmutils** v0.994  
**\gmFileKind**:

Checksum 13019 , 369  
**gmutils** v0.996  
**\gmFileKind**:  
Checksum 13254 , 369  
put to CTAN on 2011/10/14, 369

## 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 preceded with ‘p.’ are page numbers. All the numbers are hyperlinks.

<b>\!</b> , 13508	<b>\@M</b> , 3686, 3687, 3688, 4149,	<b>\@currenvir</b> , 3896, 9349,
<b>\"</b> , 13509	4744, 4910, 4912,	9376, 9378, 9502,
<b>\#</b> , 2665, 3198, 9773, 13509	9003, 9723, 10520,	12773, 12774
<b>#1</b> , <u>5170</u> , <u>13436</u>	12161, 12603, 12872,	<b>\@currenvline</b> , 9351, 9383
<b>\\$</b> , 3197, 13510	13580, 13588, 13597,	<b>\@currxt</b> , 4163, 7587
<b>\%</b> , 2655, 3198, 13509	13606, 13631, 13657,	<b>\@currname</b> , 4162, 7587
<b>\&amp;</b> , 2659, 3197	13674, 13713, 13720	<b>\@currnamestack</b> , 10252
<b>\'</b> , 13509	<b>\@RestoreEnvironment</b> ,	<b>\@currsize</b> , 9546, 9547,
<b>\(</b> , 13185, 13512	8670, 8674	9548, 9549, 9550,
<b>\)</b> , 13185, 13509	<b>\@StoreCommand</b> , 8603,	9551, 9552, 9553,
<b>\*</b> , 2581, 2583, 2859, 2925,	8628, 8661, 8663	9554, 9555
2926, 2927, 2929,	<b>\@StoreEnvironment</b> , 8656,	<b>\@dblarg</b> , 10743
8410, 8413, 12027,	8660	<b>\@dc@</b> , 5834, 5848, 7735
12028, 13111, 13600,	<b>\@UseStored</b> , 8684, 8688	<b>\@dc@AddAndParse</b> , 6261,
13601	<b>\@XA</b> , 1797, 1821, 2438,	6274, 6285
<b>\+</b> , 12029, 13509	2957, 2990, 3022,	<b>\@dc@Alias</b> , 6526, 6548,
<b>\,</b> , 11896, 12161, 12933, 13509	3949, 3951, 3953,	6549, 6558
<b>\-</b> , 8894, 8940, 8942, 9736,	5990, 6125, 7786,	<b>\@dc@AliasCatcher</b> , 6554,
10207, 12506, 13534,	7827, 8558, 8568,	6563, 6667, 6893,
13606, 13608	8730, 9278, 9357, 9502	7102, 7107, 7159,
<b>\.</b> , 12242, 12243, 13508	<b>\@XAtoks</b> , 1795, 1798, 1799	7160, 7166, 7167, 7455
<b>\/</b> , 9634, 13508, 13697	<b>\@aarg</b> , 9839, 9840, 9841,	<b>\@dc@AliasPLower</b> , 6561,
<b>\:</b> , 2749, 2752, 2807, 2809,	9843, 9846	6636, 6646, 6666,
4763, 13511	<b>\@aarga</b> , 9839, 9841	7037, 7057, 7068,
<b>\;</b> , 13508	<b>\@afterheading</b> , 10513	7089, 7099, 7104,
<b>\&lt;</b> , 9663, 13588, 13597	<b>\@allbutfirstof</b> , 1298	7135, 7318
<b>\&lt;...&gt;</b> , <u>9663</u>	<b>\@auxout</b> , 13296, 13316,	<b>\@dc@BareSpace@false</b> , 5929
<b>\=</b> , 2645, 11467, 13509	13335, 13355, 13372,	<b>\@dc@BareSpace@true</b> , 6500
<b>\&gt;</b> , 13521, 13593, 13597	13385, 13406	<b>\@dc@BareSpaceArg</b> , 6158,
<b>\?</b> , 13509	<b>\@badend</b> , 9460	6410, 7044, 7131
<b>\@</b> , 9998, 13508	<b>\@begnamedgroup</b> , 9369,	<b>\@dc@DCbody</b> , 7603, 7815, 7829
<b>\@@gm@ifxa@asiter</b> ,	9394, 9399	<b>\@dc@DCprefix@list</b> , 7762,
1468, 1504	<b>\@begnamedgroup@ifcs</b> ,	7796, 7828
<b>\@@input</b> , 4176	9395, 9398	<b>\@dc@DeclareDefault</b> ,
<b>\@@par</b> , 12634	<b>\@bsphack</b> , 7657	6297, 6306, 6307,
<b>\@CofacPrzedData@false</b> ,	<b>\@car</b> , 10061	6310, 6319, 6321,
12619	<b>\@cclv</b> , 12010	6329, 6332, 6337
<b>\@CofacPrzedData@true</b> ,	<b>\@cclvi</b> , 11995	<b>\@dc@DefAsGeneral</b> , 7367,
12606, 12617	<b>\@checkend</b> , 9457, 9475, 9483	7382, 7385, 7392
<b>\@Hy!</b>	<b>\@clsextension</b> , 3739	<b>\@dc@Gname</b> , 7331, 7332,
<b>\@maketitle@amp@plexed@true</b>	<b>\@closepenalty</b> , 10525	7339, 7344
12997	<b>\@csn</b> , 841	<b>\@dc@Gparams</b> , 7340, 7348

\@dc@GroteLang@true, 6479	6265, 7706, 7912,	\@dc@argtypes, 5934, 5944,
\@dc@GroteNegere@false,	7977, 8020, 8148	6000
5883, 6474	\@dc@ParseSpecifier,	\@dc@arguments, 5754,
\@dc@GroteNegere@true, 6468	5950, 6186, 6257, 6511	5859, 6397, 6398,
\@dc@InnerEndName, 7547,	\@dc@ParsedSpecs, 5863,	6577, 6585, 6594,
8206	5868, 6357, 6359,	6597, 6613, 6614,
\@dc@InnerName, 5835,	6369, 7714	6627, 6631, 6742,
5840, 7542, 7610,	\@dc@ResetParseAuxilia,	6983, 6995, 7016,
7749, 7776, 7777,	5828, 5866, 7703	7017, 7145, 7181,
8051, 8052, 8109,	\@dc@ResetStepAuxilia,	7190, 7299, 7306,
8115, 8177, 8500,	5923, 6256	7315, 7329, 7340,
8501, 8563, 8564	\@dc@SameAs, 6016, 8136	7345, 7400, 7404,
\@dc@K@defaultrepl, 7862,	\@dc@Scope@shipout, 7435,	7410, 7413, 7419,
7892, 7909	7447	7427, 7441, 7507,
\@dc@Loop@CareOfSpace,	\@dc@Specs@tempa, 7642,	7510, 7853, 7895,
6918, 6931, 6962	7645, 7650, 7656,	8176, 8205
\@dc@Loop@CareOfSpace@NoBraces,	7665, 7681, 7689, 7698	\@dc@basicdelims, 13179,
6934	\@dc@SpecsArName, 7558,	13190, 13196
\@dc@Loop@CareOfSpaceAndGroup,	7616, 7779, 7780,	\@dc@basicdelimsp, 13190,
6959, 6974, 6996	8054, 8055, 8106,	13192
\@dc@Loop@addallfalse,	8150, 8182, 8211,	\@dc@bsphack@, 7657, 7660,
6780, 6783	8507, 8508	7666, 7731
\@dc@Loop@addalltrue, 6781	\@dc@acpunctanddelims,	\@dc@buckstopshere, 5831,
\@dc@Loop@careofspace@aas,	13198, 13200	5966, 7707
6921, 6924, 6939, 6942	\@dc@acpunctanddelimsp,	\@dc@commasep, 7418, 7422,
\@dc@Loop@countby, 6755,	13204, 13206, 13209,	7426
6758, 6907	13213	\@dc@define@K, 6105, 7867
\@dc@Loop@drainallfalse,	\@dc@addargum, 6396, 6401,	\@dc@drses, 5941, 5945, 6125
6781, 6783	6404, 6405, 6413,	\@dc@env@endglobal, 8246,
\@dc@Loop@drainalltrue,	6414, 6417, 6585,	8283
6780	6597, 6613, 6614,	\@dc@esphack@, 7658, 7661,
\@dc@Loop@iterallfalse,	6631, 7015, 7179,	7667, 7755
6774	7188, 7297, 7304,	\@dc@falsify@xadeefaults,
\@dc@Loop@iteralltrue, 6762	7340, 7345, 7404,	5802, 5928, 6263, 6390
\@dc@Loop@test, 6804,	7413, 7418, 7419,	\@dc@global@, 7677, 7690,
6831, 6865	7426, 7427, 7439,	7726, 7748, 7898,
\@dc@LostWaxA, 7990, 7997,	7440, 7509, 7887	8246, 8269
7998, 8006, 8013,	\@dc@addinnerhash, 6080,	\@dc@grab@Loop, 6131, 7917
8018, 8020	6152, 6237	\@dc@hashgoal, 6231, 6236
\@dc@LostWaxList, 7948,	\@dc@addtoParsed@bare,	\@dc@ignore@false, 6475
7949, 7998, 8005, 8014	6376, 8029, 8035	\@dc@ignore@true, 6040,
\@dc@LowerAliases, 6546,	\@dc@addtoparsed@BareSpace,	6448, 6469
6564, 6600, 7379	6047, 6157	\@dc@ignorearg, 6147,
\@dc@Mmode@false, 5878, 5917	\@dc@addtoparsed@Ignore,	6403, 7450
\@dc@Mmode@true, 5894	6059, 6146	\@dc@innerhashes, 5841,
\@dc@Ms@init, 5891, 6196	\@dc@addtospecs, 6355,	5880, 5975, 7751, 8272
\@dc@Ms@initial@listate,	6364, 7902, 7906,	\@dc@long@letter, 5775,
5901, 6168	7907, 7909	6065, 7899
\@dc@Ms@listate, 5902,	\@dc@addtospecs@bare,	\@dc@long@true, 6444, 6480
5906, 6168	6055, 6062, 6148,	\@dc@maybe@expandafter,
\@dc@Ms@num, 5889, 5895,	6159, 6367, 6379, 7957	6340, 6349, 6356
5914, 5915, 5916, 6228	\@dc@addtotoks@bare,	\@dc@messages, 7565, 7733
\@dc@Ms@shipout, 5910,	6372, 6378, 7958,	\@dc@optparam@deft@, 6299
5972, 6025, 6177	8032, 8038	\@dc@outer@, 7678, 7682,
\@dc@ParseNextSpecifier,	\@dc@alllong@false, 7637	7726, 7737
5830, 6005, 6115,	\@dc@alllong@true, 5827,	\@dc@parse@next, 6522,
6240, 6250, 6255,	7646	7017, 7181, 7190,

7419, 7427, 7441, 7511, 7895	\@endpfalse, 9375	6309, 6331, 7039, 7070, 8356, 8952,
\@dc@process@matchsign, 7942, 7953, 8024	\@endptrue, 3834	10910, 12150, 13484, 13500, 13711
\@dc@process@ms, 6098, 6163	\@enumctr, 13241, 13242, 13248	\@ifclassloaded, 10243
\@dc@process@ms@fin, 6212, 6214, 6217	\@enumdepth, 13237, 13240, 13241	\@ifendswithpdf, 3798, 3803
\@dc@process@ms@grabnum, 6172, 6201, 6209	\@envirstack, 9344, 9348, 9352, 9357, 9364, 9445, 9477	\@ifenvir, 3883, 9458, 9486, 12874
\@dc@process@ms@hash, 6218, 6224, 6225, 6228, 6233	\@esphack, 7658	\@ifinmeaning, 1804, 11759
\@dc@process@ms@left, 6219, 6226, 6245	\@filepart, 13357, 13359	\@ifjobname, 3900
\@dc@punctanddelims, 13195, 13198, 13204	\@fileswithoptions, 3739	\@ifl@aded, 10244
\@dc@quiet@false, 7638	\@firstofmany, 1293, 1834, 1839, 4269, 4692, 7471, 12088, 12098	\@ifncat, 2735, 2737, 2752
\@dc@quiet@true, 7651	\@firstofone, 876, 1191, 2472, 3945, 4956, 4961, 5855	\@ifnch, 3180
\@dc@sequence, 6746, 6982, 7007, 7008, 7011, 7012, 7015, 7437, 7438	\@firstofthree, 1282	\@ifnextMac, 3134, 12420, 12441
\@dc@sequence@shipout, 7446, 7447, 7449	\@firstoftwo, 111, 948, 952, 954, 1048, 1064, 1080, 1088, 1399, 1426, 1445, 1516, 1648, 1668, 1779, 1848, 1850, 1942, 1997, 2040, 2076, 2500, 2504, 2529, 2827, 2872, 2876, 3600, 3813, 3913, 4197, 4215, 4424, 4619, 4647, 4796, 4940, 5130, 6343, 7342, 7471, 8120, 10968, 10973, 11813, 11817, 12901, 13377	\@ifnextac, 2820
\@dc@sequence@shipout@@, 7446, 7449	\@firstt@hensecond, 1304	\@ifnextanyIS, 3078, 3086, 6210, 12348
\@dc@setThrowArgs, 7668, 7674, 7753	\@for, 13312, 13368	\@ifnextanyRS, 3033, 3107, 12408, 13149, 13150
\@dc@specsname, 7585, 7616, 7697, 7713, 7722	\@for@num, 8582	\@ifnextcat, 2729, 2766, 2893, 2894, 6579, 6595, 9970, 9974, 9976
\@dc@tempa, 6279, 6285	\@fourthofmany, 9423, 9445	\@ifnextchar, 2875, 2954, 3135, 6274, 9827, 9834, 9839, 9888, 10745, 11519, 11521, 11522, 11910, 13147, 13392, 13629, 13655
\@dc@xadefault@i@false, 5804	\@gif, 2290, 2291, 2298	\@ifnextcharRS, 3015
\@dc@xadefault@i@true, 6456	\@gmu@mmhboxtrue, 10876, 12952	\@ifnextgroup, 2898, 2906, 13096
\@dc@xadefault@ii@false, 5805	\@gobble, 875, 973, 993, 2471, 3654, 4271, 4958, 5174, 5857, 6997, 7853, 8692, 10020, 10735, 11830, 11832, 13392, 13412	\@ifnextgroupRS, 3112, 3123
\@dc@xadefault@ii@true, 6460, 6464	\@gobble@eight, 977	\@ifnextif, 2769, 2871, 9894, 9896
\@dc@xadefault@false, 5803, 7971	\@gobble@two, 974, 2299, 2301, 2318, 2319, 11149	\@ifnextnoneRS, 3102
\@dc@xadefault@true, 6387, 6390, 6452, 7971	\@if, 2313, 2314, 2317	\@ifnextnotgroup, 2821, 2889, 2900, 6628
\@def@token, 2950, 2979, 2989, 2990, 3021, 3022, 3038, 3039, 6874, 6876	\@if@E@nextchar, 3175, 7342, 7402, 7411	\@ifnextnotgroupRS, 3122
\@dischyph, 8943	\@if@Queer@EOL, 9924, 9945	\@ifnextspace, 2996, 12440
\@eha, 9400, 10264, 13297, 13356	\@if@Xe@TeX, 3598, 3609, 3616, 3629, 5943,	\@ifnif, 2779, 2782, 2809
\@elt, 5098, 5101		\@ifnonempty, 1405, 1880, 1895
\@emptify, 1359, 1360, 1364		\@ifnotmw, 10355, 10360, 10508, 10613, 10702, 10760
\@empty, 1359, 2470, 7007, 7011, 7437, 7471, 7475, 7477, 10396, 10711, 11830, 11832, 13603		\@ifpackageloaded, 8358, 8371, 8379, 9773, 10242, 10328, 10798, 11246, 12049, 12205, 13454, 13486, 13491
\@endif, 9486		\@ifprevenvir, 9426, 9488
\@endifprev, 9488		\@ignorefalse, 9374
		\@ignoretrue, 8286
		\@include, 13298, 13300
		\@input, 9286, 13306, 13363
		\@input@, 13321

<code>\@itemdepth</code> , 13255, 13258, 13259	4440, 4609, 4640, 5014, 5976, 6074,	5793, 5834, 5836, 6283, 6662, 6816,
<code>\@itemitem</code> , 13259, 13260	6153, 6210, 6214,	6825, 6848, 7179,
<code>\@iwma</code> , 4488	6219, 7170, 7201,	7181, 7188, 7190,
<code>\@iwru</code> , 4455, 4463, 4466, 4469, 4473, 7774, 7776, 7779	7209, 8967, 8978, 8979, 9496, 9580, 9581, 9582, 9583,	7378, 7498, 7560, 7657, 7658, 7735, 7743, 7872, 7887,
<code>\@iwru@EC</code> , 1107	9850, 12000, 12011,	7895, 8067, 8068,
<code>\@iwruJ</code> , 4463, 7773, 7775, 7778, 7781	12321, 12323, 12908, 13240, 13258	8083, 8089, 8115, 8120, 8154, 8254,
<code>\@iwruU</code> , 4469	<code>\@nil</code> , 1293, 1296, 1298,	8256, 8301, 8313,
<code>\@iwruif</code> , 4471, 6398, 6961, 6983, 6995, 7016	1834, 1839, 1880, 1883, 1895, 1898,	8523, 8524, 8551, 8563, 8878, 8888,
<code>\@iwrum</code> , 1108, 4465	1947, 1959, 2002,	9152, 9170, 9173,
<code>\@latex@error</code> , 9400, 13297, 13356	2015, 2044, 2055, 2080, 2090, 2111,	9182, 9185, 9186, 9191, 9203, 10025,
<code>\@let@token</code> , 2735, 2738, 2741, 2752, 2779, 2783, 2786, 2797, 2809, 2953, 2956, 2962, 2963, 2971, 2991, 3002, 3025, 3041, 3055, 3084, 3091, 3097, 3115, 3116, 3180, 6886, 6961	2121, 2142, 2172, 2209, 2218, 2250, 3155, 3160, 3171, 4156, 4161, 4257, 4262, 4267, 4269, 4692, 7471, 7585, 8068, 8089, 9357, 9360, 9423, 9445, 9679, 9681, 9683,	10238, 10261, 10263, 10264, 10269, 10593, 10594, 11017, 11018, 11021, 11024, 11026, 11029, 11035, 11037, 11039, 11047, 11049, 11051, 11058, 11059, 11061, 11063, 11066, 11090, 11093, 11154, 11201, 11473, 11619, 11639, 11774, 11797, 11827, 11828, 11830, 11837, 11838, 11840, 11842, 11848, 11849, 11851, 11853, 12170, 13501, 13588, 13593, 13608, 13695
<code>\@listbegvskipping</code> , 4291	10061, 10957, 10960,	<code>\@nx@xa</code> , 5792, 6459
<code>\@listdepth</code> , 5129	11777, 11778, 11808,	<code>\@nxcsn</code> , 4276
<code>\@m</code> , 3689	11809, 11811, 11815,	<code>\@nxxa</code> , 6459
<code>\@mainaux</code> , 4492, 13306, 13335, 13363, 13385, 13406	11823, 11825, 11828, 11838, 11840, 11842, 11849, 11851, 11853, 11858, 11864, 12088,	<code>\@oarg</code> , 9827, 9829, 9830, 9884 <code>\@oargsq</code> , 9827, 9830 <code>\@onlypreamble</code> , 10296, 11381, 11963, 13459
<code>\@makefnmark</code> , 12986, 13007	12098, 12181, 12186, 12189, 12190, 12194, 12595, 12623, 13291, 13301	<code>\@parg</code> , 9834, 9836, 9837, 9885 <code>\@pargp</code> , 9834, 9837 <code>\@parindent</code> , 3745, 3746, 3747, 12715, 12753, 12761, 13021, 13244, 13245, 13247, 13262, 13263, 13265
<code>\@makefntext</code> , 13011, 13020	<code>\@nbreakfalse</code> , 10519, 11959 <code>\@nbreaktrue</code> , 10514 <code>\@nomath</code> , 9543 <code>\@normalcr</code> , 12682 <code>\@nx</code> , 835, 837, 864, 887, 898, 899, 901, 909, 926, 927, 929, 1029, 1878, 1880, 1883, 1893, 1895, 1898, 2250, 2741, 2786, 2794, 2826, 2868, 2871, 2872, 2875, 2876, 3146, 3189, 3256, 3267, 3268, 3309, 3344, 3356, 3369, 3370, 3561, 3567, 3574, 3654, 3780, 3798, 3799, 3803, 3818, 3819, 4060, 4071, 4107, 4277, 5101, 5792,	<code>\@partaux</code> , 13296, 13316, 13318, 13319, 13326, 13355, 13372, 13374, 13375, 13403, 13428 <code>\@partlist</code> , 13312, 13341, 13368 <code>\@pkgextension</code> , 4983, 4985, 10245 <code>\@plus</code> , 10616, 10624, 10628, 10630, 10632,
<code>\@makeother</code> , 301, 302, 303, 304, 2564, 2577, 2589, 2599, 2609, 2613, 2614, 2655, 2659, 2662, 2665, 5138, 9769, 9849, 9915, 9941, 11483, 11484, 12488		
<code>\@minus</code> , 10624, 10630, 10636, 10638, 10643, 10645, 10652, 10657		
<code>\@namedef</code> , 5175, 5178, 7504, 10460, 10478, 10599, 10714, 10753, 10826, 11714, 13236, 13251, 13254, 13268, 13763		
<code>\@nameedef</code> , 1010, 4085, 4088, 4091, 4094, 4097, 4100, 13695		
<code>\@nameuse</code> , 13333, 13384, 13432		
<code>\@ne</code> , 3675, 3676, 4129, 4187, 4398, 4416,		

10636, 10638, 10640,	12221, 12223, 12225,	<code>\@xa</code> , 834, 837, 839, 844,
10643, 10645, 10647,	12513, 12514, 12545,	847, 873, 874, 875,
10652, 10657	12576, 12894, 12908,	876, 901, 909, 916,
<code>\@popenvir</code> , 9356, 9482, 9497	12909, 12911, 12914,	929, 948, 950, 954,
<code>\@popenvir@</code> , 9357, 9360	12922, 12954, 12955,	956, 1011, 1048,
<code>\@popfilename</code> , 4172, 10250	12956, 12958	1049, 1080, 1082,
<code>\@preamblecmds</code> , 10239,	<code>\@tempcntb</code> , 12222, 12223,	1087, 1088, 1090,
10241, 10262, 10265	12224, 12226, 12227,	1191, 1193, 1279,
<code>\@prevenvir</code> , 9376	12228, 12909, 12910,	1399, 1400, 1426,
<code>\@prevgrouplevel</code> , 9370	12911	1427, 1778, 1799,
<code>\@pushenvir</code> , 9346, 9384, 9504	<code>\@tempdima</code> , 2266, 11612,	1821, 1834, 1839,
<code>\@pushfilename</code> , 4168, 10249	11619, 11636, 11639,	1882, 1897, 2205,
<code>\@relaxen</code> , 1368, 1369, 1373	12212, 12216, 12217,	2235, 2250, 2251,
<code>\@rmfromreset</code> , 5092	12219, 12220, 12224	2299, 2301, 2318,
<code>\@sN</code> , 4664	<code>\@tempdimb</code> , 3851, 3856,	2319, 2418, 2423,
<code>\@sanitize</code> , 12488	12213, 12215, 12216	2471, 2472, 2502,
<code>\@sdN</code> , 4665	<code>\@tempmuskipa</code> , 2257, 2267	2542, 2543, 2752,
<code>\@secondoffive</code> , 1286	<code>\@tempmuskipa</code> , 2256, 2265,	2809, 2827, 2828,
<code>\@secondofmany</code> , 1296, 7580	9032, 9033, 9035,	2875, 2913, 2914,
<code>\@secondofthree</code> , 1283, 8149	12660, 12661	2990, 3022, 3039,
<code>\@secondoftwo</code> , 112, 950,	<code>\@tempswafalse</code> , 1872,	3170, 3171, 3248,
956, 1049, 1064,	13310, 13367	3255, 3256, 3267,
1082, 1090, 1400,	<code>\@tempswatruer</code> , 1876, 1891,	3341, 3369, 3370,
1427, 1445, 1463,	1904, 1911, 1942,	3404, 3406, 3416,
1516, 1648, 1660,	1997, 2040, 2076,	3417, 3421, 3449,
1776, 1851, 2507,	13308, 13313, 13365,	3480, 3501, 3532,
2523, 2526, 2828,	13369	3535, 3536, 3638,
3600, 3601, 3813,	<code>\@temptokenb</code> , 5737, 5738	3648, 3654, 3707,
3915, 4198, 4216,	<code>\@textsuperscript</code> , 10903,	3713, 3721, 3799,
4424, 4427, 4621,	10904	3813, 3863, 3864,
4649, 4798, 4942,	<code>\@thefnmark</code> , 12987, 13025	3865, 3871, 3872,
5132, 6343, 6345,	<code>\@thirdofthree</code> , 1284,	3873, 3874, 3875,
7472, 8099, 10974,	2543, 8181, 8210	3913, 3915, 3949,
11813, 11817, 12241,	<code>\@toodeep</code> , 13238, 13256	3951, 3953, 4156,
12902, 13386	<code>\@topnewpage</code> , 10417	4197, 4198, 4215,
<code>\@secondthenfirst</code> , 1306	<code>\@topnum</code> , 10388	4216, 4262, 4267,
<code>\@spaces</code> , 4315	<code>\@topsepadd</code> , 4292, 4294, 4297	4269, 4271, 4277,
<code>\@sptoken</code> , 2738, 2749,	<code>\@twoofthree</code> , 1285, 2542	4324, 4583, 4619,
2783, 2807	<code>\@typeset@protect</code> , 5854	4621, 4647, 4649,
<code>\@starttoc</code> , 9286, 11952	<code>\@undefined</code> , 785, 947,	4657, 4660, 4662,
<code>\@stripstring</code> , 213, 216, 243	1248, 2950, 2962,	4692, 4796, 4798,
<code>\@temdima</code> , 9038, 9039, 9041	3055, 3268, 3408,	4851, 4940, 4942,
<code>\@tempcnta</code> , 2264, 4083,	3421, 3449, 3536,	4955, 4956, 4958,
4084, 4085, 4086,	3574, 3708, 3714,	4961, 4970, 5130,
4088, 4089, 4091,	3723, 4256, 4388,	5132, 5174, 5792,
4092, 4094, 4095,	4971, 8050, 8518,	5793, 5794, 5855,
4097, 4098, 4100,	8733, 12088, 12098	5857, 5915, 5991,
4101, 4105, 4106,	<code>\@unused</code> , 4460, 9281	6064, 6125, 6285,
4111, 4129, 6321,	<code>\@verbaarga</code> , 9843, 9899	6319, 6351, 6352,
7170, 7171, 7177,	<code>\@verbaargact</code> , 9846, 9898	6357, 6426, 6451,
7178, 7180, 7186,	<code>\@verbmargm</code> , 9852, 9894	6455, 6459, 6536,
7187, 7189, 7195,	<code>\@wckptelt</code> , 13427	6537, 6662, 6875,
7201, 7202, 7203,	<code>\@whilenum</code> , 4084, 6236,	6997, 7015, 7339,
7204, 7205, 7207,	7171, 7202, 8591,	7342, 7418, 7426,
7209, 9034, 9040,	11995, 12010	7438, 7471, 7579,
9044, 9545, 9559,	<code>\@writeckpt</code> , 13324, 13401	7580, 7696, 7697,
9561, 9563, 12220,		7698, 7713, 7714,



7722, 7723, 7738,	<code>\@xanxcssimple</code> , 906	8413, 9845, 9897,
7796, 7828, 7850,	<code>\@xanxtri</code> , 920, 3563, 3573,	11706, 11919, 11996,
7853, 7871, 7906,	8083, 8549, 8551	12481, 12485, 12486,
7909, 7948, 7957,	<code>\@xau</code> , 847, 1152, 2397,	12505, 12506, 13095,
7997, 8004, 8005,	2482, 2603, 3571,	13140, 13141, 13142,
8013, 8014, 8020,	5841, 7751, 7884,	13143, 13144, 13151,
8085, 8086, 8116,	7890, 7892, 8272,	13152, 13159, 13521,
8147, 8149, 8176,	8285, 8519, 12912	13569, 13643, 13669
8181, 8205, 8210,	<code>\@xaucs</code> , 915, 6282	<code>\activeM</code> , 2687
8240, 8254, 8519,	<code>\@xifncat</code> , 2739, 2752	<code>\activepunctsQ</code> , 13154, 13201
8532, 8535, 8546,	<code>\@xifnif</code> , 2784, 2809	<code>\activequotes</code> , 13146
8549, 8551, 8558,	<code>\@xiispaces</code> , 3155, 3157,	<code>\adashes</code> , 12482, 12484, 12492
8559, 8563, 8564,	3160, 3170	<code>\adddefaultfontfeatures</code> ,
8568, 8569, 8648,	<code>\[</code> , 2613, 11519, 11521,	8387
9155, 9156, 9255,	13185, 13512	<code>\addfontfeature</code> , 10800,
9256, 9257, 9266,	<code>\]</code> , 152, 3197, 12682, 12819,	10832, 10834, 10865,
9270, 9349, 9445,	13186, 13508	10915, 11533, 11547,
9477, 9562, 9627,	<code>\]</code> , 2614, 11521, 13185, 13508	11619, 11639, 11693,
9640, 9811, 9889,	<code>\^</code> , 150, 301, 302, 303, 304,	11797, 12121, 12144,
9898, 9899, 9908,	320, 321, 2561, 2577,	12230, 13614
10020, 10492, 10539,	2599, 2692, 2693,	<code>\addto@forlist</code> , 2467
10591, 10727, 10739,	3191, 3198, 3780,	<code>\addtoheading</code> , 10490
10781, 10957, 10968,	5138, 9769, 9915,	<code>\addtomacro</code> , 2394, 2412,
10988, 10999, 11042,	9941, 11502, 11919,	2474, 2475, 2569,
11068, 11119, 11149,	11920, 13095, 13501,	2571, 2573, 2582,
11217, 11322, 11358,	13503, 13514	2584, 2590, 2593,
11472, 11699, 11708,	<code>\_</code> , 2557, 2564, 2568, 2570,	2600, 5944, 5945,
11709, 11758, 11759,	2572, 2606, 2629,	6982, 7645, 7650,
11772, 11778, 11813,	2632, 3198, 11502,	7656, 7665, 7681,
11817, 11823, 11825,	13508, 13509	7689, 7713, 7722,
11858, 11866, 11877,	<code>\`</code> , 2609	7998, 8006, 8013,
11956, 12041, 12088,	<code>\ </code> , 13508, 13509	8018, 8237, 8388,
12178, 12241, 12244,	<code>\~</code> , 3198, 8410, 11997,	11514, 11694, 12487,
12901, 12902, 12925,	13509, 13562, 13628,	12488, 12489, 13192,
12941, 12960, 13062,	13654	13200, 13206, 13209,
13063, 13128, 13129,	<code>\ ,</code> 319, 2589, 2662, 2694,	13630, 13656
13130, 13179, 13192,	3197, 4143, 4144,	<code>\addtotoks</code> , 1734, 1746,
13193, 13195, 13196,	9679, 12007, 12012,	2422, 3843, 6064,
13200, 13201, 13206,	12186, 12546, 12547,	6147, 6158, 6357,
13207, 13213, 13218,	12548, 13076, 13514	6359, 6369, 6373,
13242, 13260, 13377,	<code>\aarg</code> , 9839, 9887	6397, 11456, 11502,
13386, 13393, 13552,	<code>\abovedisplayshortskip</code> ,	11506
13630, 13656, 13786	2035	<code>\addvspace</code> , 4297, 10585,
<code>\@xa@four</code> , 3868	<code>\abovedisplayskip</code> , 2034	10594, 12596
<code>\@xa@nx</code> , 5793, 6455	<code>\ac</code> , 13163	<code>\adjdemerits</code> , 1984
<code>\@xa@three</code> , 3860	<code>\ac@kernel</code> , 13171, 13214	<code>\advance</code> , 4129, 4294, 4398,
<code>\@xa@xa</code> , 5794, 6455, 6459	<code>\ac@u</code> , 13168, 13212	6074, 6153, 6185,
<code>\@xadef</code> , 839	<code>\acro</code> , 9969, 12063, 12118,	6907, 7195, 7209,
<code>\@xaeacher</code> , 6463	12119, 13166	9496, 9505, 9559,
<code>\@xanx</code> , 837, 6455, 12098	<code>\acrocore</code> , 12103, 12112,	10170, 10181, 10186,
<code>\@xanxcs</code> , 895, 4104, 5835,	12123	11613, 12000, 12011,
5840, 7176, 7185,	<code>\acropresetting</code> , 12065,	12224, 12292, 12321,
7736, 7738, 7749,	12071	12664, 12681, 12908,
7873, 7882, 8115,	<code>\active</code> , 2629, 2632, 2859,	13240, 13246, 13258,
8255, 8257, 8271,	4143, 4144, 5142,	13264
8282, 11147, 11472,	<code>\AE</code> , 12199	
13696, 13697		

<code>\afterassignment</code> , 1506, 1728, 2268, 3087, 4399, 6924, 6942, 7511	<code>\ArgumentCatcher@c</code> , 7407	<code>\ArgumentCatcher@Pm@i</code> , 6596
<code>\afterfi</code> , 986, 1331, 3149, 3160, 3247, 3340, 3447, 3601, 4188, 4764, 4765, 5177, 7471, 7472, 9399, 9400, 9702, 9921, 10979, 12079, 12186, 12407, 12439, 12927, 12995, 13099, 13341, 13357	<code>\ArgumentCatcher@c@</code> , 7403, 7416	<code>\ArgumentCatcher@Pm@ix</code> , 7302
<code>\afterfifi</code> , 983, 984, 1334, 3317, 3322, 3600, 10973, 10974, 12503	<code>\ArgumentCatcher@D</code> , 7062	<code>\ArgumentCatcher@Q</code> , 7021, 7063, 7448
<code>\afterfififi</code> , 1342	<code>\ArgumentCatcher@G</code> , 7356, 7372	<code>\ArgumentCatcher@S</code> , 6659
<code>\afterfiffifi</code> , 1336	<code>\Argument </code> <code>Catcher@G@longorshort</code> , 7324, 7359, 7364	<code>\ArgumentCatcher@Scope</code> , 7444
<code>\afterfiffififi</code> , 1341	<code>\ArgumentCatcher@genM</code> , 7141, 7156, 7157	<code>\ArgumentCatcher@T</code> , 6617, 6662, 7451, 7527
<code>\afteriffiffififi</code> , 1340	<code>\Argument </code> <code>Catcher@gobblespace</code> , 7526, 7529	<code>\ArgumentCatcher@T@</code> , 6603, 6630
<code>\ahyphen</code> , 12506	<code>\ArgumentCatcher@item</code> , 7111	<code>\ArgumentCatcher@U</code> , 7093
<code>\AKA</code> , 12119	<code>\ArgumentCatcher@K</code> , 7310	<code>\ArgumentCatcher@O</code> , 7040
<code>\all@other</code> , 3648	<code>\ArgumentCatcher@Loop</code> , 6641, 6674, 7025, 7045, 7077, 7094, 7118, 7132	<code>\ArgumentCatcher@D</code> , 7072
<code>\all@spaces</code> , 2588, 2590, 2593, 2603, 13193, 13207	<code>\Argument </code> <code>Catcher@Loop@add</code> , 6816, 6836, 6968	<code>\ArgumentCatcher@I</code> , 6639
<code>\all@stars</code> , 2580, 2582, 2584, 6319, 6662, 13063	<code>\Argument </code> <code>Catcher@Loop@add@</code> , 6975, 6978	<code>\argv</code> , 9938
<code>\all@unders</code> , 2567, 2569, 2571, 2573	<code>\Argument </code> <code>Catcher@Loop@afterpeep</code> , 6786, 6909	<code>\askforoverwritefalse</code> , 141
<code>\AllowFiBreak</code> , 8997	<code>\Argument </code> <code>Catcher@Loop@check</code> , 6855, 6899, 6984, 6997	<code>\AtBeginDocument</code> , 2675, 3744, 8378, 9772, 9916, 10000, 10260, 10300, 10326, 10797, 11371, 11663, 11665, 11901, 12046, 12359, 12403, 12482, 12484, 12760, 13453, 13485, 13489, 13694
<code>\alpha</code> , 11326	<code>\Argument </code> <code>Catcher@Loop@defcheck</code> , 6768, 6769, 6770, 6854, 6895	<code>\AtEndOfPackage</code> , 13502
<code>\among</code> , 1434, 1821, 7498	<code>\Argument </code> <code>Catcher@Loop@drain</code> , 6825, 6835, 6988	<code>\ATf@font</code> , 13024, 13030
<code>\ampulex@Args</code> , 9149, 9152, 9157	<code>\Argument </code> <code>Catcher@Loop@shipout</code> , 6771, 6848, 6913, 7001	<code>\ATfootnotes</code> , 12976
<code>\ampulexdef</code> , 9220, 9268, 9286, 9475, 10294, 11483, 13007, 13011	<code>\ArgumentCatcher@M</code> , 7156	<code>\backquote</code> , 2610
<code>\ampulexlet</code> , 9104, 9241, 9264, 9278	<code>\ArgumentCatcher@m@i</code> , 6584	<code>\balsmiley</code> , 3754
<code>\AmSTeX</code> , 10100	<code>\ArgumentCatcher@m@ix</code> , 7292	<code>\BareSpace</code> , 6159, 6499
<code>\any</code> , 6642, 7027, 7029, 7047, 7049, 7080, 7082, 7918, 7927, 8028, 8029, 8032, 8042	<code>\ArgumentCatcher@PB</code> , 6591	<code>\baselineskip</code> , 2034, 8439, 8458, 8879, 8889, 12292, 12303, 12604, 12835
<code>\approx</code> , 11273	<code>\ArgumentCatcher@PC</code> , 7408, 7410	<code>\batchfile</code> , 131
<code>\arabic</code> , 12240, 12242, 12243, 12244	<code>\ArgumentCatcher@Pc</code> , 7408	<code>\beforeDot</code> , 194, 219
<code>\arg</code> , p. 142, 9877, 9917, 9918	<code>\ArgumentCatcher@Pc@</code> , 7412, 7424	<code>\begin</code> , 9394, 11953, 12007, 13183
<code>\arg@dc</code> , 9843, 9846, 9859, 9869, 9889, 9932, 9950	<code>\ArgumentCatcher@PG</code> , 7361, 7376	<code>\begin*</code> , 9394
<code>\arg@wrap</code> , 9821, 9829, 9836, 9840, 9865, 9942	<code>\Argument </code> <code>Catcher@Pgobblespace</code> , 7529	<code>\begingroup</code> , 146, 319, 1278, 1349, 2792, 3225, 3288, 3332, 3378, 3397, 3427, 3441, 4304, 8410, 8600, 8654, 8662, 8668, 8682, 9375, 9501, 9544, 9665, 9734, 9922, 9939, 10914, 10986, 11688, 11693, 11874, 12064, 12209, 12691, 12722, 13040, 13043, 13182,
<code>\ArgumentCatcher@</code> , 6536, 7130	<code>\ArgumentCatcher@PM</code> , 7157	
<code>\ArgumentCatcher@B</code> , 6574		
<code>\ArgumentCatcher@C</code> , 7397, 7407		

13561, 13627, 13653, 13707	<code>\c@#1</code> , 13436	<code>\chardef</code> , 2606
<code>\belowdisplayskip</code> , 2036	<code>\c@dc@argnum</code> , 5760, 5871, 5976, 6153, 6185, 6219, 6232, 6236, 6246, 7723, 7964, 7968	<code>\check@mathfonts</code> , 10035, 10078
<code>\belowdisplayskip</code> , 2035	<code>\c@dc@catchernum</code> , 5759, 5870, 6074, 7873, 7882	<code>\chi</code> , 11350
<code>\beta</code> , 11327	<code>\c@dc@Loop@bound</code> , 6747, 6751, 6754, 6858, 6900, 6907	<code>\cipolagwa</code> , 12721
<code>\bfseries</code> , 11942	<code>\c@EnvirInterruption</code> , 9491, 9496, 9498, 9503, 9505	<code>\clap</code> , 4671, 5047
<code>\bgcolor</code> , 11537	<code>\c@FiBreakPenalty</code> , 8995, 9000, 9005	<code>\ClassError</code> , 10482
<code>\bgroup</code> , 1716, 2439, 2893, 3115, 6210, 6274, 6579, 6595, 10876, 12536, 12537, 12538, 12562, 12563, 12564, 12952, 13181, 13392	<code>\c@gmu@TokensCount</code> , 4361, 4387, 4398, 4416, 4440	<code>\cleardoublepage</code> , 10342, 13089
<code>\BibTeX</code> , 10103	<code>\c@NoNumSecs</code> , 10328	<code>\clearemptydoublepage</code> , 13088
<code>\bidate</code> , 12588, 12609	<code>\c@page</code> , 4939	<code>\clearpage</code> , 10342, 13304, 13323, 13360, 13400
<code>\bigcirc</code> , 11307	<code>\c@secnumdepth</code> , 10365	<code>\closeout</code> , 13326, 13403
<code>\Biggl</code> , 11452	<code>\cat</code> , 9963	<code>\clubpenalty</code> , 1981, 9780, 10520, 10525
<code>\biggl</code> , 11450	<code>\catcode</code> , 148, 150, 151, 152, 319, 320, 321, 2557, 2561, 2568, 2570, 2572, 2581, 2583, 2613, 2614, 2629, 2632, 2692, 2693, 2694, 2859, 3189, 3196, 3780, 4143, 4144, 5142, 8413, 9845, 9850, 9897, 11502, 11706, 11919, 11920, 11996, 12481, 12485, 12486, 12505, 12506, 13095, 13140, 13141, 13142, 13143, 13144, 13151, 13152, 13159, 13501, 13503, 13521, 13569, 13643, 13669	<code>\cmd</code> , 9811
<code>\Biggr</code> , 11453	<code>\centerthis</code> , 13042	<code>\cmd@to@cs</code> , 9811, 9815
<code>\biggr</code> , 11451	<code>\chapter</code> , 10614	<code>\color</code> , 9649, 12022, 12023
<code>\Bigl</code> , 11448	<code>\chaptermark</code> , 10707	<code>\CommandLet</code> , 8530, 8575, 8576, 8648, 8738, 8942, 8943
<code>\bigl</code> , 11446	<code>\char</code> , 2844, 3620, 3852, 3853, 3854, 3855, 4753, 8928, 9740, 9815, 9821, 10175, 10826, 10833, 11409, 11433, 11461, 11462, 11463, 11536, 11545, 11548, 11551, 11552, 11553, 11554, 11555, 12012, 12150, 12448, 13589, 13594, 13600, 13633, 13634, 13640, 13659, 13660, 13666, 13672, 13675	<code>\condstobin</code> , 4528
<code>\Bigr</code> , 11449		<code>\condstobin@iter</code> , 4531, 4534, 4543
<code>\bigr</code> , 11447		<code>\ConTeXt</code> , 10201
<code>\bigskipamount</code> , 11933		<code>\continuum</code> , 12057
<code>\bihyphen</code> , 8899		<code>\copy</code> , 8840, 10041, 10076, 10090, 10164, 10193, 11536, 11549
<code>\binoppenalty</code> , 1980, 13575		<code>\copyRightLeaf</code> , 284
<code>\blekotnizza</code> , 1834, 1839		<code>\cos</code> , 11365
<code>\blekotnizza@</code> , 1152		<code>\count</code> , 1975, 7931, 7984, 8018, 10047, 10048, 10049, 10050, 10051, 10052, 10053, 10054, 11993, 11995, 11996, 11997, 12000, 12009, 12010, 12011, 12012
<code>\boldmath</code> , 10061		<code>\count@</code> , 2311, 2315
<code>\boolean</code> , 1825, 1902		<code>\CS</code> , 9968, 9974, 9976
<code>\boolstobin</code> , 4507, 5907		<code>\cs</code> , 9723, 9765, 9770, 9773, 9811
<code>\boolstobin@iter</code> , 4510, 4513, 4521		<code>\cs@inner</code> , 9747, 9754
<code>\box</code> , 8767, 8857, 10084, 10171, 10182, 10187, 11445		<code>\CSes</code> , 9976
<code>\boxmaxdepth</code> , 1937		<code>\csn</code> , 4825, 4831
<code>\bracefirstofone</code> , 1275		<code>\csname</code> , 204, 213, 224, 234, 243, 252, 257, 262, 364, 365, 805, 841, 844, 901, 909, 916, 929, 1011, 1047, 1087, 2205, 2299, 2318, 2502, 2545, 3248, 3256, 3267, 3341, 3369, 3370, 3406, 3417, 3418, 3421, 3448, 3480,
<code>\breadth</code> , 8956, 8963		
<code>\broccoli</code> , 1152		
<code>\brokenpenalty</code> , 1982		
<code>\bslash</code> , 889, 2651, 3365, 7203, 7205, 7544, 7548, 7840, 8191, 8255, 8271, 9751, 10483, 10484, 10485, 12393, 12394, 12436, 12437, 13727, 13730		
<code>\bslash@or@ac</code> , 884, 3248, 3256, 3341, 3352, 3363, 3370, 3405, 3406, 3415, 3418, 3446, 3448, 3480, 3502, 7543, 7560, 8613		
<code>\bullet</code> , 3620, 3627		

3501, 3532, 3533,	<code>\DC@EndNotDeclaredErr,</code>	<code>\DeclareLogo,</code> 10012,
3536, 3707, 3713,	8185, 8189, 8214	10032, 10060, 10071,
3721, 3798, 3799,	<code>\dc@gobblespace@dummy,</code>	10103, 10109, 10112,
4277, 4522, 4544,	7527, 7531, 7533	10114, 10117, 10120,
4657, 4660, 4662,	<code>\dc@innername,</code> 7576, 7610,	10127, 10129, 10130,
4664, 4665, 4697,	7622, 7629, 7736,	10136, 10149, 10161,
4698, 4955, 4970,	7873, 7882, 7964, 7968	10198, 10201
5104, 6065, 6536,	<code>\dc@parse@next,</code> 7343, 7344	<code>\DeclareMathVersion,</code> 11010
6538, 6875, 6877,	<code>\dc@ThrowArgs,</code> 7669, 7842	<code>\DeclareOption,</code> 10785
7147, 7344, 7585,	<code>\DCcoordinate,</code> p. 78, 7395	<code>\DeclareOptionX,</code> 5157,
7850, 7852, 8150,	<code>\DCMessagesfalse,</code> p. 81	5170, 5188
8177, 8180, 8182,	<code>\DCMessagestrue,</code> p. 81	<code>\declarepreamble,</code> 281
8206, 8209, 8211,	<code>\DCnocoordinate,</code> p. 78,	<code>\DeclareRobustCommand,</code>
8240, 8730, 8764,	7392, 7393	9580, 9581, 9582, 9583
9004, 9006, 9278,	<code>\DCUse,</code> 8163, 8299, 8311,	<code>\DeclareSymbolFont,</code>
9385, 9503, 10362,	8323, 8331, 8341, 9241	11018, 11021, 11035,
10394, 10546, 10547,	<code>\DCUseEnd,</code> 8198	11047, 11059
10561, 10576, 10727,	<code>\deadcycles,</code> 1977, 13332,	<code>\DeclareTextCommand,</code> 10025
10739, 10781, 10856,	13383	<code>\DeclareTextCommandDe </code>
10989, 11042, 11068,	<code>\DeclareCommand,</code> 5810,	<code>fault,</code>
11119, 11207, 11217,	7623, 7629, 7786,	10027
11322, 11496, 11709,	7827, 7867, 7917,	<code>\DeclareUrlCommand,</code>
11956, 11957, 12241,	7981, 8155, 8163,	13687, 13707
12774, 13242, 13260,	8192, 8198, 8223,	<code>\def,</code> 103, 109, 111, 112,
13390, 13633, 13640	8225, 8240, 8282,	123, 127, 136, 143,
<code>\CSNameIf,</code> 2534	8295, 8296, 8299,	144, 194, 196, 197,
<code>\csnameIf,</code> 202, 228	8307, 8308, 8311,	200, 202, 208, 209,
<code>\CSs,</code> 9974	8318, 8319, 8323,	210, 212, 248, 308,
<code>\ctan,</code> 11369	8327, 8337, 8352,	317, 325, 331, 805,
<code>\ctg,</code> 11367	8394, 8417, 8434,	837, 839, 841, 844,
<code>\cup,</code> 11546	8530, 8548, 8574,	847, 860, 862, 884,
<code>\currentBundle,</code> 127	8582, 8628, 8688,	895, 906, 915, 920,
<code>\currentgrouplevel,</code> 8623,	8737, 8761, 8788,	935, 938, 942, 962,
8645, 9350, 9370	8874, 8883, 8899,	964, 971, 977, 983,
<code>\czas,</code> 12511	8902, 8953, 8960,	986, 990, 991, 997,
<code>\czat,</code> 9199	8975, 8997, 9025,	1010, 1018, 1042,
<code>\czer,</code> 12024, 12028, 12029	9029, 9104, 9220,	1057, 1072, 1073,
<code>\czerwo,</code> 12023, 12024	9723, 9869, 9963,	1075, 1086, 1113,
	10614, 10911, 10998,	1127, 1134, 1135,
<code>\date@left,</code> 12631, 12638	11004, 11075, 11077,	1136, 1137, 1140,
<code>\date@line,</code> 12609, 12628,	11109, 11161, 11210,	1147, 1178, 1203,
12638	11493, 11528, 11679,	1232, 1244, 1273,
<code>\DateFont,</code> 12609, 12621	11800, 11870, 12535,	1275, 1282, 1283,
<code>\dateskipamount,</code> 12596,	12561, 12592, 12687,	1284, 1285, 1286,
12625, 12626	12716, 12733, 12744,	1293, 1296, 1298,
<code>\day,</code> 1992	12779, 12804, 12838,	1304, 1306, 1330,
<code>\dc@addtoParsed@,</code> 5915,	12946, 12976, 13062,	1331, 1334, 1336,
6262, 6362, 6388,	13111, 13163, 13212,	1340, 1341, 1342,
7944, 7955, 7960,	13397, 13413, 13693	1349, 1351, 1353,
7967, 7972	<code>\DeclareEnvironment,</code>	1385, 1424, 1444,
<code>\dc@addtotoks@,</code> 6348, 6363	p. 81, 8223, 8328,	1493, 1504, 1607,
<code>\dc@DefParseNext,</code> 6520,	8331, 8338, 8341, 12296	1611, 1627, 1640,
6584, 6596, 6745,	<code>\DeclareFontFamily,</code>	1651, 1666, 1671,
7146, 7316, 7403,	11024, 11037, 11049,	1673, 1675, 1678,
7412, 7508	11061	1682, 1686, 1715,
<code>\dc@EAName,</code> 7838, 7843,	<code>\DeclareFontShape,</code> 11026,	1724, 1727, 1732,
7850, 7852, 8257, 8263	11039, 11051, 11063	1741, 1827, 1845,

1905, 1912, 1944,  
2000, 2042, 2078,  
2109, 2139, 2207,  
2260, 2272, 2298,  
2317, 2318, 2378,  
2416, 2467, 2492,  
2511, 2521, 2534,  
2565, 2577, 2580,  
2588, 2597, 2610,  
2616, 2617, 2633,  
2656, 2660, 2663,  
2666, 2678, 2685,  
2687, 2737, 2749,  
2752, 2776, 2777,  
2778, 2782, 2807,  
2809, 2825, 2858,  
2860, 2926, 2979,  
3048, 3063, 3145,  
3155, 3175, 3189,  
3196, 3212, 3242,  
3300, 3309, 3350,  
3445, 3478, 3499,  
3562, 3598, 3627,  
3629, 3638, 3648,  
3654, 3665, 3670,  
3674, 3685, 3694,  
3736, 3754, 3759,  
3778, 3803, 3818,  
3819, 3826, 3834,  
3837, 3849, 3860,  
3868, 3906, 3919,  
3932, 3940, 3996,  
4005, 4015, 4030,  
4038, 4047, 4057,  
4068, 4104, 4162,  
4163, 4167, 4182,  
4203, 4222, 4240,  
4257, 4261, 4276,  
4291, 4303, 4308,  
4313, 4322, 4327,  
4341, 4393, 4406,  
4434, 4449, 4497,  
4507, 4513, 4528,  
4534, 4562, 4598,  
4613, 4626, 4643,  
4657, 4660, 4662,  
4664, 4665, 4677,  
4707, 4708, 4711,  
4712, 4763, 4788,  
4789, 4792, 4794,  
4811, 4823, 4829,  
4836, 4845, 4847,  
4854, 4855, 4889,  
4904, 4909, 4916,  
4936, 4951, 4963,  
4966, 4989, 5007,  
5034, 5041, 5054,

5056, 5098, 5126,  
5139, 5144, 5145,  
5165, 5775, 5792,  
5793, 5794, 5802,  
5840, 5848, 5866,  
5889, 5891, 5895,  
5906, 5910, 5916,  
5923, 5934, 5941,  
5950, 6146, 6152,  
6157, 6163, 6209,  
6217, 6255, 6273,  
6297, 6340, 6348,  
6355, 6362, 6367,  
6372, 6376, 6383,  
6396, 6404, 6413,  
6425, 6429, 6520,  
6526, 6546, 6554,  
6561, 6574, 6591,  
6603, 6617, 6639,  
6674, 6769, 6865,  
6895, 6968, 6978,  
6988, 7001, 7008,  
7021, 7040, 7062,  
7072, 7093, 7111,  
7130, 7141, 7156,  
7157, 7176, 7185,  
7292, 7302, 7310,  
7321, 7324, 7339,  
7343, 7348, 7367,  
7392, 7397, 7410,  
7416, 7422, 7424,  
7435, 7437, 7444,  
7463, 7475, 7477,  
7479, 7481, 7526,  
7531, 7542, 7547,  
7657, 7658, 7749,  
7762, 7772, 7838,  
7862, 7882, 7964,  
7990, 8024, 8067,  
8093, 8136, 8234,  
8269, 8409, 8411,  
8421, 8427, 8491,  
8513, 8607, 8617,  
8660, 8674, 8767,  
8798, 8801, 8804,  
8909, 8913, 9111,  
9130, 9149, 9154,  
9182, 9230, 9251,  
9260, 9346, 9356,  
9360, 9369, 9394,  
9398, 9423, 9457,  
9542, 9584, 9585,  
9640, 9649, 9651,  
9663, 9666, 9679,  
9681, 9690, 9697,  
9740, 9762, 9773,  
9776, 9778, 9811,

9815, 9836, 9837,  
9840, 9841, 9843,  
9846, 9852, 9865,  
9881, 9883, 9884,  
9885, 9886, 9887,  
9893, 9925, 9946,  
10020, 10022, 10100,  
10206, 10238, 10258,  
10261, 10263, 10264,  
10267, 10288, 10331,  
10334, 10335, 10339,  
10341, 10355, 10361,  
10378, 10379, 10380,  
10381, 10454, 10455,  
10469, 10480, 10490,  
10513, 10534, 10541,  
10572, 10575, 10585,  
10590, 10599, 10705,  
10710, 10721, 10733,  
10738, 10743, 10744,  
10749, 10754, 10755,  
10808, 10823, 10824,  
10831, 10862, 10880,  
10881, 10904, 10909,  
10919, 10952, 10956,  
10960, 10962, 10964,  
10965, 10970, 10984,  
10996, 11142, 11183,  
11192, 11361, 11362,  
11363, 11364, 11365,  
11366, 11367, 11368,  
11369, 11395, 11396,  
11413, 11418, 11419,  
11436, 11437, 11459,  
11460, 11467, 11468,  
11470, 11472, 11476,  
11510, 11511, 11514,  
11517, 11518, 11519,  
11521, 11522, 11535,  
11539, 11540, 11541,  
11569, 11598, 11599,  
11601, 11606, 11676,  
11704, 11705, 11713,  
11716, 11734, 11748,  
11757, 11774, 11781,  
11808, 11811, 11815,  
11827, 11837, 11848,  
11862, 11864, 11895,  
11920, 11929, 11930,  
11931, 11932, 11933,  
11936, 11952, 11992,  
12028, 12029, 12039,  
12062, 12075, 12087,  
12097, 12118, 12119,  
12123, 12135, 12143,  
12147, 12150, 12151,  
12168, 12181, 12189,

12199, 12206, 12207,	<code>\degrouplettoken</code> , 1716,	<code>\discre</code> , 4743, 4746, 4751,
12208, 12210, 12233,	1718, 1725, 1733	4765, 8920, 9673, 9690
12290, 12292, 12346,	<code>\degrouptoks</code> , 1707, 1711,	<code>\discret</code> , 4746
12355, 12448, 12451,	1734, 1746	<code>\discretionary</code> , 4744,
12452, 12457, 12458,	<code>\dekbigskip</code> , 11933	8920, 9736, 12396,
12461, 12465, 12467,	<code>\dekfracc</code> , 10874	12442, 13661, 13672
12468, 12469, 12482,	<code>\dekfraccargs</code> , 10853,	<code>\disobeylines</code> , 12820, 13093
12484, 12485, 12486,	10863, 10877, 12148	<code>\displayindent</code> , 1940
12502, 12506, 12519,	<code>\dekfraccsimple</code> , 12147,	<code>\displaywidowpenalty</code> , 1981
12623, 12631, 12671,	12153	<code>\displaywidth</code> , 1939
12713, 12715, 12761,	<code>\dekfraccslash</code> , 12138,	<code>\divide</code> , 10049, 10052,
12771, 12826, 12899,	12150, 12151	10053, 12215, 12216,
12906, 12933, 12945,	<code>\dekmedbigskip</code> , 11931	12221, 12910
13007, 13011, 13061,	<code>\dekmedskip</code> , 11932	<code>\dName</code> , 4662
13093, 13097, 13123,	<code>\deksmallskip</code> , 11929	<code>\do</code> , 3189, 3191, 3196, 3197,
13146, 13154, 13171,	<code>\delimitfactor</code> , 1992	3198, 3478, 3499,
13179, 13195, 13249,	<code>\delimitershortfall</code> , 1938	4084, 4792, 4804,
13266, 13291, 13292,	<code>\Delta</code> , 11312	4806, 4808, 6236,
13293, 13295, 13300,	<code>\delta</code> , 11329	7171, 7202, 8592,
13303, 13340, 13345,	<code>\depth</code> , 8954, 8961	10238, 10261, 10263,
13354, 13359, 13391,	<code>\destdir</code> , 208	10758, 10759, 10760,
13411, 13426, 13427,	<code>\detoken@xa</code> , 3638	11395, 11409, 11410,
13452, 13505, 13508,	<code>\detokenize</code> , 223, 233, 251,	11416, 11418, 11424,
13511, 13512, 13513,	256, 261, 1190, 1191,	11427, 11433, 11436,
13518, 13550, 13556,	1205, 1206, 1393,	11446, 11447, 11448,
13563, 13573, 13576,	1879, 1883, 1894,	11449, 11450, 11451,
13578, 13580, 13581,	1898, 2250, 3171,	11452, 11453, 11459,
13620, 13629, 13646,	3638, 3807, 3810,	11464, 11465, 11470,
13655, 13679, 13695,	3818, 3819, 3928,	11475, 11483, 11484,
13697, 13707, 13712,	4324, 4660, 4662,	11540, 11544, 11546,
13719, 13727, 13730,	4665, 7496, 8068,	11547, 11551, 11552,
13744, 13746, 13747,	8089, 9171, 9349,	11553, 11554, 11555,
13748, 13749, 13750,	10854, 10856, 11759,	11556, 11705, 11710,
13751, 13752, 13759,	11772, 11775, 11827,	11713, 11717, 11735,
13786, 13816, 13817,	11828, 11837, 11838,	11749, 11995, 12010,
13818, 13819, 13821,	11848, 11849	12487, 12489, 13312,
13822, 13823, 13824		13368, 13508, 13509,
<code>\Default</code> , 7934	<code>\di</code> , 1152	13510, 13511, 13512,
<code>\default</code> , 7934, 13213	<code>\dilitkern</code> , 11606	13514, 13576, 13578,
<code>\defaultshyphenchar</code> , 1990	<code>\dimen</code> , 1930, 8823, 10170,	13580, 13581, 13593,
<code>\defaultskewchar</code> , 1990	10171, 10181, 10182,	13597, 13600, 13601
<code>\defEacher</code> , 7937, 7962	10186, 10187	<code>\do@url@hyp</code> , 13603
<code>\defENumLoop</code> , 9260	<code>\dimexpr</code> , 1909, 2266, 3829,	<code>\document</code> , 10296
<code>\defHashy</code> , 1824, 1888	3830, 3856, 4192,	<code>\doifdefined</code> , 13786,
<code>\defLowStarFake</code> , 8414, 8417	4241, 4722, 4723,	13788, 13789, 13790,
<code>\defNoHash</code> , 1823, 1874	4815, 4816, 4817,	13791, 13792, 13793,
<code>\defobeylines</code> , 11920	4836, 4854, 4865,	13794, 13795, 13796,
<code>\defRCSeDatum</code> , 13759,	4876, 4895, 4905,	13797, 13798, 13799
13767, 13768, 13769,	8401, 8402, 8457,	<code>\dospecials</code> , 3191, 12487
13770, 13771	8808, 8878, 8879,	<code>\dots</code> , 12749
<code>\degroupp@addanditer</code> ,	8888, 8889, 11401,	<code>\doublehyphendemerits</code> , 1984
1720, 1732	11406, 11414, 11444,	<code>\downarrow</code> , 11295
<code>\degroupp@addanditer@i</code> ,	11537, 11726, 11738,	<code>\dp</code> , 4233, 4235, 4905, 8792,
1737, 1741	11783, 11793, 11797,	8808, 10170, 10181,
<code>\degroupp@drainanditer</code> ,	11883, 11884, 12738,	10186, 11441, 11444,
1717, 1719, 1727, 1735	12752, 12789, 13065,	11537
	13069	<code>\Drain</code> , 7926

<code>\drain</code> , 7926	11792, 11796, 11819,	2179, 2180, 2181,
<code>\Drop</code> , 7926	12240, 12738, 12751,	2182, 2187, 2188,
<code>\drop</code> , 7926	12912, 12921, 13241,	2189, 2190, 2191,
<code>\drs</code> , 5938, 5941	13259, 13311, 13361,	2192, 2193, 2194,
<code>\dywiz</code> , 12503	13501, 13529, 13540,	2211, 2212, 2220,
	13558, 13559, 13560,	2221, 2222, 2236,
	13587, 13592, 13605	2472, 2543, 2635,
<code>\Eacher</code> , 7937		2637, 2680, 2740,
<code>\eacher</code> , 7937, 8793, 13214	<code>\edefInfo</code> , 239, 240, 241, 248	2743, 2785, 2788,
<code>\eaddtomacro</code> , 2479	<code>\edefU</code> , 1615, 2717, 2732,	2799, 2828, 2844,
<code>\edef</code> , 131, 208, 216, 218,	2733, 2734, 2952,	2914, 3148, 3158,
222, 227, 231, 249,	3083, 3086, 3178,	3268, 3303, 3314,
962, 1011, 1470,	3179, 4259, 4260,	3318, 3407, 3420,
1829, 1877, 1892,	6299, 6302, 6522,	3449, 3485, 3563,
2246, 2397, 2418,	6899, 7395, 7603,	3569, 3574, 3600,
2482, 2603, 2719,	7696, 7948, 7949	3601, 3620, 3708,
2793, 2867, 3168,	<code>\eequals</code> , 11970	3714, 3723, 3807,
3169, 3190, 3254,	<code>\egfirstofone</code> , 1349, 1351	3810, 3813, 3914,
3367, 3563, 3570,	<code>\egReset@Macro</code> , 3397, 3399	4198, 4216, 4243,
3653, 3773, 3779,	<code>\egReset@MacroSt</code> , 3397, 3400	4522, 4544, 4620,
3798, 3802, 3850,	<code>\egRestore@Macro</code> , 3332, 3334	4648, 4765, 4797,
3856, 4103, 4270,	<code>\egRestore@Macros</code> , 3379,	4837, 4845, 4898,
4323, 4324, 4500,	3381	4919, 4941, 4957,
4911, 5833, 5839,	<code>\egRestore@MacroSt</code> , 3332,	4960, 5131, 5176,
5901, 5975, 6218,	3335	5856, 7472, 7743,
6219, 6231, 6279,	<code>\egroup</code> , 1718, 2894, 3116,	7840, 8464, 8471,
6659, 6786, 6921,	9888, 10857, 10867,	8478, 8720, 8722,
6939, 7173, 7331,	12141, 12959, 13182	8856, 8862, 9053,
7496, 7497, 7558,	<code>\egStore@Macro</code> , 3226, 3231	9055, 9057, 9212,
7610, 7616, 7668,	<code>\egStore@Macros</code> , 3288, 3290	9281, 9400, 9543,
7730, 7747, 7870,	<code>\egStore@MacroSt</code> , 3226, 3232	9546, 9547, 9548,
7880, 8066, 8078,	<code>\else</code> , 214, 254, 259, 876,	9549, 9550, 9551,
8083, 8114, 8253,	888, 900, 928, 949,	9552, 9553, 9554,
8268, 8281, 8500,	955, 984, 986, 1049,	9555, 9565, 9682,
8501, 8507, 8508,	1081, 1089, 1400,	9683, 9700, 9738,
8519, 8520, 8522,	1427, 1478, 1950,	9745, 9890, 9899,
8532, 8535, 8547,	1951, 1952, 1953,	9921, 10021, 10046,
8588, 8589, 8593,	1954, 1961, 1962,	10121, 10176, 10179,
8620, 8641, 8707,	1963, 1964, 1965,	10185, 10238, 10340,
8711, 8715, 8825,	1966, 2004, 2005,	10342, 10383, 10386,
8877, 8887, 9111,	2006, 2007, 2008,	10395, 10403, 10416,
9152, 9181, 9200,	2017, 2018, 2019,	10419, 10423, 10453,
9214, 9230, 9276,	2020, 2021, 2022,	10466, 10521, 10524,
9370, 9376, 9378,	2046, 2047, 2048,	10544, 10552, 10827,
9383, 9736, 9742,	2049, 2057, 2058,	10963, 10969, 10974,
10024, 10262, 10362,	2059, 2060, 2061,	11094, 11121, 11125,
10492, 10592, 10999,	2082, 2083, 2084,	11128, 11131, 11301,
11016, 11034, 11046,	2085, 2092, 2093,	11302, 11431, 11432,
11057, 11086, 11092,	2094, 2095, 2096,	11463, 11652, 11654,
11095, 11100, 11145,	2113, 2114, 2115,	11677, 11796, 11809,
11169, 11171, 11175,	2116, 2123, 2124,	11813, 11817, 11831,
11201, 11215, 11219,	2125, 2126, 2127,	11841, 11852, 11942,
11324, 11357, 11399,	2145, 2146, 2147,	12039, 12077, 12169,
11404, 11440, 11441,	2148, 2149, 2150,	12174, 12176, 12183,
11471, 11572, 11574,	2151, 2156, 2157,	12186, 12211, 12365,
11618, 11689, 11723,	2158, 2159, 2160,	12374, 12395, 12424,
11758, 11772, 11773,	2161, 2162, 2175,	12438, 12464, 12491,
11778, 11783, 11787,	2176, 2177, 2178,	

12502, 12503, 12548,  
 12572, 12605, 12743,  
 12875, 12902, 12913,  
 12926, 12960, 13022,  
 13071, 13075, 13099,  
 13218, 13239, 13257,  
 13298, 13328, 13341,  
 13357, 13379, 13542,  
 13583, 13596, 13601,  
 13635, 13638, 13659,  
 13661, 13664, 13666  
 \em, 11650, 11660  
 \emdash, 12448  
 \emergencystretch, 1936  
 \emph, 11659, 12649, 12652  
 \emptify, 1360, 1480, 1495,  
 5880, 6746, 7642,  
 7660, 7661, 7666,  
 7667, 7674, 9344,  
 10204, 10802, 10985,  
 11073, 11509, 11513,  
 11795, 12038, 12071,  
 12892, 13032, 13689  
 \empty, 194, 213, 219, 3784,  
 10452, 10453, 10464,  
 11709, 12210, 13303  
 \encodingdefault, 11019,  
 11022, 11024, 11026,  
 11029, 11036, 11037,  
 11039, 11048, 11049,  
 11051, 11060, 11061,  
 11063, 11066  
 \end, 9475, 9486, 9488,  
 11959, 12013, 13183  
 \endcsname, 203, 204, 213,  
 224, 232, 234, 243,  
 250, 252, 255, 257,  
 260, 262, 364, 365,  
 805, 809, 841, 844,  
 901, 909, 916, 929,  
 1011, 1047, 1087,  
 2205, 2301, 2319,  
 2499, 2502, 2541,  
 2545, 3248, 3256,  
 3262, 3267, 3341,  
 3369, 3370, 3405,  
 3406, 3415, 3417,  
 3418, 3421, 3446,  
 3448, 3480, 3502,  
 3530, 3532, 3533,  
 3536, 3572, 3706,  
 3707, 3712, 3713,  
 3720, 3721, 3799,  
 4277, 4522, 4544,  
 4657, 4660, 4662,  
 4664, 4665, 4697,  
 4698, 4824, 4830,  
 4954, 4955, 4969,  
 4970, 5104, 5222,  
 6067, 6281, 6534,  
 6537, 6538, 6875,  
 6885, 7149, 7332,  
 7344, 7585, 7843,  
 7850, 7852, 8051,  
 8054, 8106, 8109,  
 8150, 8177, 8182,  
 8183, 8206, 8211,  
 8212, 8240, 8263,  
 8521, 8722, 8727,  
 8730, 8764, 9004,  
 9006, 9279, 9385,  
 9399, 9475, 9476,  
 9503, 10362, 10394,  
 10546, 10547, 10561,  
 10576, 10727, 10739,  
 10780, 10781, 10856,  
 10989, 11042, 11068,  
 11123, 11144, 11207,  
 11218, 11323, 11496,  
 11709, 11956, 11957,  
 12241, 12773, 12774,  
 13242, 13260, 13390,  
 13436, 13633, 13640  
 \endexpr, 4851, 4928, 8459,  
 12867, 12876  
 \endfilepart, 13389,  
 13397, 13409, 13413  
 \endfirstcentered, 3846  
 \endgraf, 3766, 3772,  
 12728, 12821, 12827,  
 13040, 13047, 13098  
 \endgroup, 322, 1279, 1351,  
 2796, 3231, 3232,  
 3290, 3334, 3335,  
 3381, 3399, 3400,  
 3429, 3442, 4309,  
 8411, 8634, 8675,  
 8702, 9499, 9562,  
 9665, 9760, 9888,  
 9902, 10916, 10993,  
 11694, 11700, 11887,  
 12067, 12229, 12697,  
 12729, 13040, 13047,  
 13182, 13563, 13629,  
 13655, 13685  
 \endinput, 13896  
 \endlastcentered, 3834, 3846  
 \endlinechar, 147, 1278,  
 1990, 3760, 11696,  
 13529, 13530, 13536  
 \endlist, 13251, 13268  
 \EndLost, 7986  
 \endlost, 7986  
 \endpreamble, 293  
 \endrule, 4931  
 \endskiplines, 140  
 \englishdate, 12560  
 \enlargefsize, 8874  
 \enoughpage, 8434  
 \enspace, 13027  
 \ensuremath, 3620, 3627,  
 9626, 9636, 10122,  
 10905, 11201, 12058  
 enumerate\*, 13236  
 \env, 9765  
 \envhack, 7663, 7665, 7769,  
 8234  
 \envirlet, 3583, 8574  
 \EofTeX, 10002  
 \epsilon, 11330  
 \equals, 11968  
 \errhelp, 2074  
 \errorcontextlines, 299,  
 1993, 7538  
 \escapechar, 873, 1830,  
 1833, 1836, 1990,  
 2288, 2311, 2315,  
 3246, 3338, 3403  
 \eta, 11333  
 \eTeX, 8383, 10120, 10125,  
 10127  
 \everycr, 2074  
 \everydisplay, 2073,  
 11481, 11503, 11507  
 \everyeof, 2074  
 \everyhbox, 2073  
 \everyjob, 2074  
 \everymath, 2073, 11456,  
 11481, 11484, 11502,  
 11503, 11506, 11507  
 \everypar, 2073, 3843,  
 10516, 10526, 12335  
 \everyvbox, 2074  
 \ExecuteOptionsX, 5189,  
 13779  
 \exhyphenpenalty, 1980,  
 13715, 13722  
 \exists, 11416  
 \expandafter, 107, 365,  
 805, 834, 4000, 4001,  
 4002, 4010, 4011,  
 4012, 4022, 4023,  
 4024, 4033, 4034,  
 4035, 4041, 4042,  
 4043, 4050, 4051,  
 4052, 4061, 4062,  
 4063, 4072, 4073,  
 4074, 6451, 6455,  
 6459, 10061  
 \f@encoding, 12007



\f@family, 10989	2844, 2914, 3151,	11135, 11137, 11151,
\f@series, 10061	3161, 3247, 3255,	11156, 11189, 11190,
\f@size, 8878, 8888, 13069	3269, 3305, 3323,	11204, 11251, 11256,
\fake@onum@ii, 11692, 11694	3324, 3340, 3346,	11301, 11302, 11306,
\fakeonum, 11679	3366, 3368, 3404,	11359, 11360, 11409,
\fakern, 11476	3409, 3416, 3422,	11425, 11428, 11431,
\fakesc@extrascale,	3450, 3486, 3564,	11432, 11434, 11463,
12219, 12233	3575, 3578, 3600,	11488, 11600, 11655,
\fakescups, 12194	3601, 3620, 3709,	11656, 11677, 11798,
\fakescupscore, 12172, 12196	3715, 3724, 3748,	11809, 11813, 11817,
\fakescextrascale, 12233	3807, 3810, 3813,	11833, 11843, 11854,
\fam, 1991	3916, 4003, 4013,	11942, 11958, 11999,
\fi, 205, 236, 244, 264, 265,	4025, 4036, 4044,	12040, 12051, 12083,
266, 366, 792, 797,	4053, 4064, 4075,	12141, 12175, 12177,
806, 813, 831, 876,	4187, 4189, 4190,	12179, 12184, 12187,
890, 902, 930, 951,	4191, 4199, 4217,	12211, 12226, 12227,
957, 985, 987, 1019,	4244, 4272, 4295,	12242, 12243, 12244,
1050, 1083, 1091,	4522, 4544, 4622,	12323, 12367, 12374,
1331, 1334, 1336,	4650, 4754, 4767,	12398, 12429, 12445,
1340, 1341, 1342,	4799, 4826, 4832,	12472, 12493, 12503,
1401, 1428, 1481,	4837, 4845, 4899,	12514, 12550, 12574,
1482, 1496, 1950,	4913, 4922, 4943,	12607, 12613, 12664,
1951, 1952, 1953,	4959, 4962, 4972,	12758, 12775, 12858,
1954, 1961, 1962,	4973, 5087, 5133,	12877, 12903, 12918,
1963, 1964, 1965,	5185, 5228, 5739,	12919, 12924, 12929,
1966, 2004, 2005,	5777, 5858, 6387,	12952, 12957, 12961,
2006, 2007, 2008,	6391, 7137, 7213,	12996, 12998, 13023,
2017, 2018, 2019,	7437, 7473, 7744,	13066, 13073, 13077,
2020, 2021, 2022,	7840, 7899, 8053,	13096, 13099, 13100,
2046, 2047, 2048,	8056, 8465, 8473,	13131, 13218, 13221,
2049, 2057, 2058,	8480, 8519, 8520,	13250, 13267, 13298,
2059, 2060, 2061,	8526, 8567, 8612,	13303, 13307, 13313,
2082, 2083, 2084,	8647, 8720, 8722,	13314, 13320, 13327,
2085, 2092, 2093,	8855, 8858, 8864,	13334, 13341, 13357,
2094, 2095, 2096,	8929, 9059, 9060,	13364, 13369, 13370,
2113, 2114, 2115,	9061, 9215, 9258,	13376, 13387, 13389,
2116, 2123, 2124,	9281, 9401, 9557,	13404, 13405, 13436,
2125, 2126, 2127,	9561, 9567, 9627,	13524, 13542, 13553,
2145, 2146, 2147,	9682, 9683, 9702,	13583, 13598, 13602,
2148, 2149, 2150,	9735, 9739, 9746,	13613, 13636, 13642,
2151, 2156, 2157,	9900, 9909, 9931,	13659, 13662, 13666,
2158, 2159, 2160,	9933, 9949, 10006,	13668, 13786
2161, 2162, 2175,	10008, 10023, 10055,	\file, 180, 313, 9695, 13687
2176, 2177, 2178,	10061, 10122, 10183,	\filepart, 13354, 13356,
2179, 2180, 2181,	10184, 10188, 10238,	13412
2182, 2187, 2188,	10340, 10342, 10365,	\finalhyphendemerits, 1984
2189, 2190, 2191,	10369, 10383, 10387,	\firstcentered, 3837
2192, 2193, 2194,	10389, 10399, 10414,	\firstofone, 109, 322,
2211, 2212, 2220,	10418, 10421, 10422,	1273, 6558, 11153, 13218
2221, 2222, 2252,	10425, 10426, 10427,	\firstoftwo, 196, 6798
2256, 2257, 2264,	10459, 10467, 10523,	\floatingpenalty, 1983
2265, 2266, 2267,	10527, 10541, 10554,	\font, 2844, 3620, 3850,
2361, 2368, 2437,	10567, 10568, 10726,	4752, 4753, 5048,
2473, 2544, 2607,	10782, 10829, 10857,	5054, 5063, 8396,
2638, 2639, 2682,	10864, 10867, 10876,	8398, 8400, 8422,
2745, 2746, 2801,	10975, 10976, 10980,	8927, 8928, 9742,
2802, 2803, 2829,	11097, 11121, 11133,	9743, 10045, 10047,

10050, 10121, 10173, 10177, 10825, 10957, 10992, 11025, 11038, 11042, 11050, 11062, 11068, 11399, 11401, 11404, 11406, 11598, 11612, 11613, 11636, 11651, 11653, 11723, 11726, 11738, 11758, 11772, 11783, 11788, 11793, 11823, 11825, 11858, 11866, 11877, 11884, 12212, 12213, 12659, 12665, 12714, 13533, 13583, 13599	<code>\FormatHangHeading</code> , 10623, 10631, 10639, 10646 <code>\FormatRunInHeading</code> , 10653, 10658 <code>\freeze@actives</code> , 11992 <code>\from</code> , 313 <code>\FromDir</code> , 136, 209, 210 <code>\fullpar</code> , 12687 <code>\fullparcore</code> , 12695, 12699 <code>\futurelet</code> , 1725, 2735, 2752, 2779, 2809, 2971, 3097, 3180  <code>\g@emptify</code> , 1364, 1365 <code>\g@relaxen</code> , 1373, 1374 <code>\gaddtomacro</code> , 2412 <code>\Gamma</code> , 11311 <code>\gamma</code> , 11328 <code>\garamath</code> , p. 159, 11528 <code>\gdef</code> , 2299, 2752, 2809, 9111, 9213, 9230, 9361, 9362, 9363, 9364 <code>\ge</code> , 11241, 11242, 11243, 11255 <code>\geeng</code> , 11243 <code>\gemptify</code> , 1365 <code>\generate</code> , 327 <code>\GenericInfo</code> , 9278 <code>\GenericInfoToTerminal</code> , 9276 <code>\GenericWarning</code> , 4314 <code>\geq</code> , 11242, 11253, 11254, 11255 <code>\getprevdepth</code> , 3765, 12822 <code>\getprevdepthlocal</code> , 3771 <code>\glenvirlet</code> , 3588 <code>\glet</code> , 2378, 10540 <code>\global</code> , 1364, 1373, 2289, 2302, 2378, 2412, 2461, 2749, 3247, 3249, 3255, 3270, 3313, 3340, 3342, 3368, 3373, 3404, 3410, 3416, 3423, 3484, 3487, 3514, 3515, 3552, 4731, 4876, 7438, 7439, 7448, 7685, 7690, 7762, 8263, 8646, 8793, 8794, 9108, 9213, 9229, 9496, 9505, 10388, 10985, 12298, 12303 <code>\globaldefs</code> , 1989 <code>\glueexpr</code> , 1909, 2265, 4212, 4213, 4214,	4233, 4235, 8401, 11931, 12660, 12739, 12754, 12867, 12876 <code>\glueshrink</code> , 4214 <code>\gluestretch</code> , 4213, 12661 <code>\glyphbound</code> , 5020, 5030, 5034 <code>\glyphname</code> , 3622, 11558 <code>gm#1</code> , 5188 <code>\gma</code> , 11522 <code>\gma@arrowdash</code> , 11535, 11545, 11551, 11553 <code>\gma@bare</code> , 11518, 11520 <code>\gma@bracket</code> , 11520, 11521 <code>\gma@checkbracket</code> , 11519, 11523 <code>\gma@dollar</code> , 11517, 11518, 11523 <code>\gma@gmathhook</code> , 11479, 11513, 11514, 11539 <code>\gma@quantifierhook</code> , 11410, 11416, 11509, 11511, 11547, 11555, 11556 <code>\gma@tempa</code> , 11399, 11401, 11404, 11406, 11440, 11444, 11471, 11474 <code>\gma@tempb</code> , 11441, 11444 <code>\gmath</code> , p. 159, 11493, 11517, 11521 <code>\gmath@delc</code> , 11161, 11189 <code>\gmath@delcif</code> , 11183 <code>\gmath@delimif</code> , 11192 <code>\gmath@do</code> , 11077, 11124, 11127, 11130, 11224, 11226, 11227, 11228, 11229, 11230, 11231, 11232, 11233, 11234, 11235, 11236, 11258, 11259, 11260, 11261, 11262, 11263, 11264, 11265, 11268, 11278, 11279, 11280, 11303 <code>\gmath@doif</code> , 11109, 11146, 11225, 11238, 11241, 11244, 11245, 11267, 11269, 11270, 11271, 11272, 11273, 11274, 11276, 11277, 11281, 11282, 11283, 11284, 11285, 11286, 11287, 11288, 11291, 11295, 11297, 11304, 11307, 11308, 11309, 11311, 11312, 11313, 11314, 11315, 11316, 11317, 11318, 11319, 11320, 11326, 11327, 11328,
<code>\fontchardp</code> , 8422 <code>\fontcharht</code> , 11723, 11788 <code>\fontcharwd</code> , 3850, 5054, 12714 <code>\fontdimen</code> , 8396, 8398, 8400, 10045, 10047, 10050, 10177, 11356, 11399, 11401, 11404, 11406, 11598, 11612, 11613, 11636, 11651, 11653, 11726, 11738, 11783, 11788, 11793, 11884, 11885, 12212, 12213, 12665 <code>\fontsize</code> , 8878, 8888, 10036, 10079, 10905 <code>\fontspec</code> , 2679, 11829, 11830, 11839, 11840, 11850, 11851, 11876, 11879 <code>\foeatletter</code> , 1353 <code>\fooblanksignored</code> , 2696 <code>\foolc</code> , 4303, 13710 <code>\foone</code> , 1349, 1353, 2557, 2561, 2564, 2568, 2570, 2572, 2577, 2581, 2583, 2589, 2592, 2596, 2599, 2609, 2613, 2629, 2655, 2659, 2662, 2665, 2685, 2687, 2697, 2859, 2909, 3133, 4143, 5138, 5142, 9769, 9845, 9849, 9915, 11919, 12481, 12505, 12837, 13092, 13139 <code>\footnotesize</code> , 9548, 9564, 12621, 12642, 12832 <code>\forall</code> , 11410		

11329, 11330, 11331,  
 11332, 11333, 11334,  
 11335, 11336, 11337,  
 11338, 11339, 11340,  
 11341, 11342, 11343,  
 11344, 11345, 11346,  
 11347, 11348, 11349,  
 11350, 11351, 11352  
 $\backslash$ gmath@fam, 10998, 11081,  
 11115, 11121, 11217,  
 11322  
 $\backslash$ gmath@famm, 11219, 11225,  
 11238, 11241, 11244,  
 11245, 11267, 11269,  
 11270, 11271, 11272,  
 11273, 11274, 11276,  
 11277, 11281, 11282,  
 11283, 11284, 11285,  
 11286, 11287, 11288,  
 11291, 11295, 11297,  
 11304, 11307, 11308,  
 11309, 11311, 11312,  
 11313, 11314, 11315,  
 11316, 11317, 11318,  
 11319, 11320, 11324,  
 11326, 11327, 11328,  
 11329, 11330, 11331,  
 11332, 11333, 11334,  
 11335, 11336, 11337,  
 11338, 11339, 11340,  
 11341, 11342, 11343,  
 11344, 11345, 11346,  
 11347, 11348, 11349,  
 11350, 11351, 11352  
 $\backslash$ gmath@famnum, 10999,  
 11088, 11103, 11170,  
 11177, 11203  
 $\backslash$ gmath@font, 11189, 11199,  
 11301, 11302, 11355,  
 11356, 11409, 11423,  
 11426, 11430, 11431,  
 11432  
 $\backslash$ gmath@ft, 11119, 11124,  
 11127, 11130  
 $\backslash$ gmath@getfamnum, 10998,  
 11084, 11167, 11200  
 $\backslash$ gmath@restore, 11132,  
 11134, 11136, 11142  
 $\backslash$ gmath@version, 11122,  
 11215, 11219, 11322,  
 11324  
 $\backslash$ gmathbase, 11075, 11372,  
 11374, 11381, 11494  
 $\backslash$ gmathcats, 11505  
 $\backslash$ gmathFams, 11004, 11076  
 $\backslash$ gmathfamshook, 11070, 11073  
 $\backslash$ gmathfurther, 11388, 11495  
 $\backslash$ gmathhook, 11514  
 $\backslash$ gmathscripts, 11501  
 $\backslash$ gmBundleFile, 136, 260,  
 262, 277, 313, 325  
 $\backslash$ gmBundleName, 123, 131  
 $\backslash$ gmLeaf, 281, 329  
 $\backslash$ gmfile, 308, 317  
 $\backslash$ gmFileDate, 273  
 $\backslash$ gmFileInfo, 275  
 $\backslash$ gmFileKind, 278, 331  
 $\backslash$ gmFileVersion, 274  
 $\backslash$ gmIStoreCS, 3296, 3319,  
 3387, 3431  
 $\backslash$ gmIstoremacros, 3297,  
 3308, 3317, 3322,  
 3390, 3434  
 $\backslash$ gmlogos@hyphen, 10056,  
 10091, 10127, 10129,  
 10130, 10198, 10201,  
 10204, 10207  
 $\backslash$ gmno@NotprerrMessage,  
 10264, 10267  
 $\backslash$ gmobeyspaces, 4144  
 $\backslash$ gmOutName, 218, 224, 228,  
 232, 234, 250, 252,  
 255, 257, 270  
 $\backslash$ gmOutThanks, 231, 276  
 $\backslash$ gmOutTitle, 222, 271  
 $\backslash$ gmOutYears, 227, 272, 284  
 $\backslash$ gmshowbox, 3670  
 $\backslash$ gmshowlists, 3665, 8953,  
 8960, 8976  
 $\backslash$ GMtextsuperscript, 10909  
 $\backslash$ gmtracingoutput, 3674, 8975  
 $\backslash$ gmtroff, 3694  
 $\backslash$ gmtron, 3685  
 $\backslash$ gmu, 1140, 1153  
 $\backslash$ gmu@absdim, 4854  
 $\backslash$ gmu@absExpr, 4847, 4854,  
 4855  
 $\backslash$ gmu@absnum, 4855  
 $\backslash$ gmu@acroiter, 12076,  
 12087, 12088, 12091  
 $\backslash$ gmu@acrokernel, 12089,  
 12097, 13171  
 $\backslash$ gmu@acrospace, 12066,  
 12075, 12079  
 $\backslash$ gmu@activespace, 2685, 9640  
 $\backslash$ gmu@activespaceblank,  
 9630, 9639  
 $\backslash$ gmu@addtoheading@resa,  
 10492  
 $\backslash$ gmu@AdvancePagegoal, 4881  
 $\backslash$ gmu@among@, 1444, 1446  
 $\backslash$ gmu@AmpulexDelimiter,  
 9133, 9160, 9185  
 $\backslash$ gmu@ampulexlet@resa,  
 9130, 9155, 9181,  
 9182, 9194  
 $\backslash$ gmu@ampulexlet@resb,  
 9200, 9211  
 $\backslash$ gmu@ampulexlet@resc,  
 9154, 9163, 9204, 9214  
 $\backslash$ gmu@AND, 4643, 6223  
 $\backslash$ gmu@AT@ampulex, 12992,  
 12995, 13003  
 $\backslash$ gmu@ATfootnotes, 12985,  
 12991, 13009, 13012,  
 13016  
 $\backslash$ gmu@basefont, 11866, 11878  
 $\backslash$ gmu@bihyphen@char, 8909,  
 8924  
 $\backslash$ gmu@bihyphen@corr, 8913,  
 8933, 8949  
 $\backslash$ gmu@boxA, 8778, 8805,  
 8808, 8809, 8840,  
 8841, 8843, 8850,  
 8852, 8857, 8860  
 $\backslash$ gmu@boxB, 8779, 8843,  
 8848, 8849, 8851  
 $\backslash$ gmu@calc@scale, 10967,  
 10972, 11870  
 $\backslash$ gmu@calculateslant,  
 11718, 11781  
 $\backslash$ gmu@CASE, 1134, 1254,  
 1830, 1833, 1836,  
 1874, 1888, 1902,  
 1930, 1975, 2031,  
 2070, 2105, 2137,  
 2205, 2502, 2525,  
 2953, 2961, 5966,  
 5982, 6013, 8028, 8034  
 $\backslash$ gmu@CASEnot, 1135, 1245,  
 1248, 2499, 2522,  
 8106, 8109  
 $\backslash$ gmu@CASEsbany, 1696,  
 6095, 6104, 6110,  
 6121, 6125, 6130,  
 6426, 8031, 8037  
 $\backslash$ gmu@CASEsbnone, 1698, 5991  
 $\backslash$ gmu@CASEstrany, 1692, 1909  
 $\backslash$ gmu@CASEstrnone, 1694  
 $\backslash$ gmu@CASExany, 1136  
 $\backslash$ gmu@CASExnone, 1137  
 $\backslash$ gmu@cepstnof, 11800,  
 11876, 11879  
 $\backslash$ gmu@cepstnof@resa,  
 11819, 11857  
 $\backslash$ gmu@cepstnof@resb,  
 11827, 11837, 11848,  
 11858

<code>\gmu@cepstnof@resc,</code> 11808, 11840, 11842, 11851, 11853	<code>\gmu@dekfracc,</code> 10831, 10855, 10859	<code>\gmu@extreme,</code> 4677, 4699, 4700, 4707, 4708, 4711, 4712
<code>\gmu@cepstnof@resd,</code> 11811, 11823	<code>\gmu@dekfraccsimple,</code> 10859, 12135, 12148	<code>\gmu@extremebox,</code> 8788
<code>\gmu@cepstnof@rese,</code> 11815, 11825	<code>\gmu@delim,</code> 4510, 4517, 4523, 4531, 4539, 4545, 4572, 4587, 4608, 4639	<code>\gmu@Fake,</code> 4949, 4963, 4970 <code>\gmu@FakeLoaded,</code> 4951, 4983 <code>\gmu@FakeUnloaded,</code> 4966, 4985
<code>\gmu@checkaftersec,</code> 10534, 10593	<code>\gmu@denominatorkern,</code> 10833, 10881, 12138	<code>\gmu@fileext,</code> 13293, 13303, 13321
<code>\gmu@checkaftersec@resa,</code> 10540, 10541, 10544	<code>\gmu@dima,</code> 8780, 8807, 8813, 8814, 8815	<code>\gmu@filename,</code> 13292, 13306, 13318, 13321, 13324, 13333
<code>\gmu@checkM,</code> 2906, 2911	<code>\gmu@dimb,</code> 8781, 8797, 8800, 8813, 8815, 8824, 8825	<code>\gmu@filepart@resa,</code> 13368, 13369
<code>\gmu@CL@left,</code> 8549, 8559, 8569	<code>\gmu@dimextreme@comp,</code> 8798, 8801, 8813	<code>\gmu@filepartname,</code> 13361, 13369, 13384, 13401
<code>\gmu@CL@PrepareArg,</code> 8513, 8543, 8544	<code>\gmu@dimextreme@step,</code> 8793, 8804	<code>\gmu@firstandspace,</code> 860, 874 <code>\gmu@fitto,</code> 4716, 4731 <code>\gmu@flexhyphen,</code> 4750, 8945, 13118, 13127
<code>\gmu@CL@right,</code> 8546, 8551, 8559, 8563, 8564, 8569	<code>\gmu@dimratio,</code> 4811	<code>\gmu@fnhook,</code> 13026, 13032 <code>\gmu@fontscale,</code> 10977, 11881 <code>\gmu@fontscalebr,</code> 10977, 10978, 10985, 11027, 11040, 11052, 11064
<code>\gmu@clearpagesduetoopenright,</code> 10341, 10384	<code>\gmu@discretionaryhyphen,</code> 8894, 8902, 8942, 8943, 8945, 13534	<code>\gmu@fontstring@,</code> 10954, 10956, 11223 <code>\gmu@fontstring@@,</code> 10957, 10960
<code>\gmu@CountToken@iter,</code> 4389, 4393, 4399	<code>\gmu@discretionaryslash,</code> 9690, 9701	<code>\gmu@forallkerning,</code> 11411, 11413
<code>\gmu@CountToken@token,</code> 4388, 4394, 4400	<code>\gmu@disMinner,</code> 13096, 13097	<code>\gmu@forarg,</code> 1615, 1778, 6426 <code>\gmu@foreach,</code> 1607, 1777, 6440
<code>\gmu@CountTokens,</code> 4363, 4415, 4439	<code>\gmu@dofakeonum,</code> 11698, 11704	<code>\gmu@foreach@break,</code> 1627, 1780
<code>\gmu@CountTokens@end,</code> 4360, 4390, 4394	<code>\gmu@dogmathbase,</code> 11210, 11371, 11372	<code>\gmu@foreach@delim,</code> 1607, 1612, 1623, 1628, 1777, 6440
<code>\gmu@currfont@descaled,</code> 11877, 11885	<code>\gmu@dontnumbersectionsoutofmargin,</code> 10339, 10368	<code>\gmu@forer,</code> 1611, 1618, 1623 <code>\gmu@fornum@curr,</code> 8590, 8591, 8593, 8594
<code>\gmu@datef,</code> 12642	<code>\gmu@doUrlMath,</code> 13576, 13578, 13580, 13620	<code>\gmu@fornum@lim,</code> 8589, 8591 <code>\gmu@fornum@min,</code> 8588, 8590
<code>\gmu@datefsl,</code> 12535, 12541, 12561, 12566, 12589	<code>\gmu@doUrlMathAc,</code> 13581, 13588, 13597, 13606, 13646	<code>\gmu@fracfontsetup,</code> 12137, 12140, 12143
<code>\gmu@debug@msgsfalse,</code> 3700	<code>\gmu@drain,</code> 1729, 3088, 6925, 6943, 9482, 9498	<code>\gmu@futureifany,</code> 3060, 3063 <code>\gmu@g@enlargeto,</code> 4728 <code>\gmu@generalCASE,</code> 1113, 1134, 1135, 1136, 1137, 1692, 1694, 1696, 1698
<code>\gmu@debug@msgstrue,</code> 3690	<code>\gmu@dywiz,</code> 12502, 12506	<code>\gmu@getadvsv,</code> 10585, 10591
<code>\gmu@DeclareCommand@resa,</code> 5839, 5844, 7747, 7757	<code>\gmu@EatCases,</code> 1122, 1127, 1132, 2528, 6506, 8112	
<code>\gmu@DeclareEnvironment@resa,</code> 8234, 8237, 8241, 8268, 8277	<code>\gmu@EdefCurrnames,</code> 4154, 4169	
<code>\gmu@DeclareEnvironment@resb,</code> 8253, 8285	<code>\gmu@EdefCurrnames@,</code> 4156, 4161	
<code>\gmu@DeclareEnvironment@resc,</code> 8281, 8289	<code>\gmu@EdefCurrnames@resa,</code> 4155, 4156	
<code>\gmu@DeclareLogo@resa,</code> 10020, 10022, 10024, 10025, 10026	<code>\gmu@eloops@resa,</code> 4103, 4122	
<code>\gmu@DefSymbol,</code> 1016, 1500, 1501, 1823, 1824, 1825, 4360, 4949	<code>\gmu@ENumLoop,</code> 9251, 9255, 9264, 9270	
<code>\gmu@deggroup,</code> 1709	<code>\gmu@ESAC,</code> 1129, 1259, 1840, 2232, 2508, 2530, 2969, 6141, 6508, 8045, 8131	
<code>\gmu@deggroup@afterlet,</code> 1715, 1725		
<code>\gmu@deggroup@end,</code> 1712, 1743		
<code>\gmu@deggroup@iter,</code> 1712, 1724, 1728, 1747		

<code>\gmu@geteschar</code> , 1827, 1924	6779, 6792, 6812,	1912, 1942, 1944,
<code>\gmu@getext</code> , 13291, 13301	6821, 6846, 6874,	1947, 1959, 1997,
<code>\gmu@getfontdata</code> , 10984,	6900, 7007, 7011,	2000, 2002, 2015,
11014, 11033, 11045,	7332, 7378, 7574,	2040, 2042, 2044,
11056	7619, 7627, 7843,	2055, 2076, 2078,
<code>\gmu@getfontscale</code> , 10962,	7883, 7889, 7905,	2080, 2090, 2109,
10963, 10979, 10988,	7946, 7947, 7962,	2111, 2121, 2139,
10990	7970, 7975, 8072,	2141, 2171, 2207,
<code>\gmu@getfontstring</code> ,	8263, 8451, 8523,	2209, 2218, 2235,
10952, 10991	8727, 8796, 8807,	2240, 2247
<code>\gmu@getht</code> , 8836	8812, 8814, 12098, 12099	<code>\gmu@ifIs@resb</code> , 2246, 2251
<code>\gmu@getslant</code> , 11771, 11782	<code>\gmu@if@onum</code> , 11687, 11757	<code>\gmu@iflastglue</code> , 9031
<code>\gmu@GobbleMinus</code> , 4845, 4851	<code>\gmu@if@PreLimit</code> , 1487,	<code>\gmu@iflastkern</code> , 9037
<code>\gmu@gobdef</code> , 984, 990, 1030	1500	<code>\gmu@iflist</code> , 5126
<code>\gmu@gobmacro</code> , 3648, 3654	<code>\gmu@ifBooleanMacro</code> ,	<code>\gmu@ifLong@resa</code> , 7496,
<code>\gmu@halfcomb</code> , 5041	1845, 1905	7497, 7498, 7499
<code>\gmu@halfwd</code> , 5046, 5051, 5054	<code>\gmu@ifbox</code> , 4222	<code>\gmu@ifna@afterlet</code> , 3083,
<code>\gmu@Hashes</code> , 4005, 4089	<code>\gmu@ifCSdefined</code> , 2521,	3097
<code>\gmu@hashes</code> , 4015, 4022,	7618, 8298, 8310, 8321	<code>\gmu@ifna@resa</code> , 3086, 3087
4086, 4106, 5976,	<code>\gmu@ifDe@next</code> , 8099,	<code>\gmu@ifnac</code> , 2822, 2825
7178, 7187	8120, 8132	<code>\gmu@ifnextStrXany</code> , 3048,
<code>\gmu@Hashesbraced</code> , 4038,	<code>\gmu@ifDe@resa</code> , 8114, 8126	3092
4098	<code>\gmu@ifDeclared</code> , 8093,	<code>\gmu@ifpageodd</code> , 4936
<code>\gmu@hashesbraced</code> , 4047,	8140, 8170, 8202, 8546	<code>\gmu@ifpeeped</code> , 2985, 6920,
4050, 4095, 7180, 7189	<code>\gmu@ifdefinable</code> , 942,	6938
<code>\gmu@hashesOut</code> , 4057,	1026, 1360, 1365,	<code>\gmu@ifQUANT@iter</code> , 4562,
4061, 4072, 4110	1369, 1374	4583, 4601, 4633
<code>\gmu@hashesOutU</code> , 4068	<code>\gmu@ifdefined</code> , 2511,	<code>\gmu@ifquantfalse</code> , 1467
<code>\gmu@HHashes</code> , 3996, 4000,	8330, 8340	<code>\gmu@ifquanttrue</code> , 1474
4010, 4092	<code>\gmu@ifdetokens</code> , 3919,	<code>\gmu@ifsbany</code> , 1673, 1683,
<code>\gmu@HHashesbraced</code> , 4030,	3949, 5099, 6881, 7992	1696, 1790, 6040,
4033, 4041, 4101	<code>\gmu@ifdim</code> , 4182, 4212,	7663, 7680, 7685
<code>\gmu@hyperrefstepcounter</code> ,	4213, 4214	<code>\gmu@ifsbintersect</code> , 1789,
10331, 10334, 10397	<code>\gmu@ifedetokens</code> , 1847,	7644, 7649, 7654
<code>\gmu@if</code> , 864, 1042, 1063,	1849, 3896, 3900,	<code>\gmu@ifsbnone</code> , 1682, 1698,
1072, 1134, 1179,	3940, 3953, 9444	6024
1488, 1612, 1656,	<code>\gmu@ifempty</code> , 863, 1385,	<code>\gmu@ifskip</code> , 4203, 4232
1659, 1716, 1718,	1405, 1445, 1463,	<code>\gmu@ifstar</code> , 2868, 3226,
1733, 1743, 1948,	2273, 4258, 4519,	3238, 3332, 3397,
1960, 2003, 2016,	4541, 4574, 6747,	9394, 10754, 10808,
2045, 2056, 2081,	6760, 6778, 8072,	10859, 12828, 13456
2091, 2112, 2122,	8908, 8912, 8924,	<code>\gmu@ifstored</code> , 3339, 3350,
2144, 2155, 2174,	8933, 9139, 9477,	3357, 11033, 11045,
2186, 2210, 2219,	10456, 10476, 12553,	11091
2989, 2990, 2991,	12579	<code>\gmu@ifstrany</code> , 1671, 1679,
3002, 3021, 3022,	<code>\gmu@ifexempty</code> , 1424,	1692, 1787, 3073
3025, 3038, 3084,	1945, 2001, 2043,	<code>\gmu@ifstrcmp</code> , 3906, 3928,
3115, 3116, 3356,	2079, 2110, 2140, 2208	3937, 3945
3530, 4394, 4416,	<code>\gmu@ifHashless@</code> , 8067, 8085	<code>\gmu@ifstrintersect</code> ,
4421, 4440, 4450,	<code>\gmu@ifHashlessMacro</code> ,	1786, 8004
4472, 4578, 4586,	8078, 8113	<code>\gmu@ifstrnone</code> , 1678, 1694
4692, 4697, 4722,	<code>\gmu@ifincsname</code> , 1075, 4341	<code>\gmu@ifStrXany</code> , 1675,
4863, 4874, 5014,	<code>\gmu@ifintersect</code> , 1765,	1687, 3039, 6630
5015, 5025, 5026,	1768, 1787, 1790	<code>\gmu@ifStrXnone</code> , 1686
5914, 6031, 6164,	<code>\gmu@ifIs@resa</code> , 1877,	<code>\gmu@ifsw</code> , 1072, 6046,
6168, 6280, 6341,	1878, 1882, 1892,	6054, 6058
6533, 6751, 6761,	1893, 1897, 1905,	

<code>\gmu@ifSXXany</code> , 1640, 1671, 1673, 1675	<code>\gmu@luzniej</code> , 12318, 12321, 12323	<code>\gmu@prependtotoks@ambig</code> , 2429, 2454, 2461
<code>\gmu@ifSXXany@</code> , 1647, 1651, 1661	<code>\gmu@MakeScopePrefix</code> , 3509, 3529, 3559	<code>\gmu@prevsec</code> , 10515, 10517, 10539, 10546, 10576
<code>\gmu@ifSXXany@end</code> , 1647, 1659, 1667	<code>\gmu@maketabtab</code> , 5144	<code>\gmu@printslashes</code> , 9695, 9697, 9699, 9702
<code>\gmu@ifSXXany@found</code> , 1657, 1666	<code>\gmu@maxdim</code> , 4711	<code>\gmu@pswords</code> , 9679, 9681, 9683
<code>\gmu@ifSystemX</code> , 4780	<code>\gmu@maxnum</code> , 4707	<code>\gmu@quiettrue</code> , 5157, 10785
<code>\gmu@ifundefined</code> , 2492, 2512, 3351, 3363, 10099, 10135, 10137, 10148, 10150, 10169, 10355, 10368, 10384, 10460, 10481, 10535, 10548, 10619, 10750, 10854, 12218, 12240	<code>\gmu@measurewd</code> , 3849, 11720, 11736, 11750, 13586, 13591, 13604	<code>\gmu@removeelement</code> , 4251
<code>\gmu@ifutokens</code> , 3932, 3951, 6342, 9148	<code>\gmu@medmskip</code> , 13559, 13565	<code>\gmu@removeelement@resa</code> , 4257, 4262, 4267
<code>\gmu@ifxa@as</code> , 1470, 1477, 1506	<code>\gmu@mindim</code> , 4712	<code>\gmu@Require@ampulex</code> , 13790
<code>\gmu@ifxa@asiter</code> , 1468, 1480, 1483, 1487, 1495, 1497	<code>\gmu@minnum</code> , 4708, 9780	<code>\gmu@Require@command</code> , 13788
<code>\gmu@ifxa@drainer</code> , 1477, 1493	<code>\gmu@namelet</code> , 3523, 3540, 3552, 8263, 8503, 8510	<code>\gmu@Require@envir</code> , 13789
<code>\gmu@ifxa@Limit</code> , 1479, 1487, 1494, 1501	<code>\gmu@namelet@scpref</code> , 3529, 3532, 3535	<code>\gmu@Require@logos</code> , 13793
<code>\gmu@ifxa@token</code> , 1473, 1479, 1494, 1507	<code>\gmu@nobound</code> , 5007, 5061	<code>\gmu@Require@meta</code> , 13792
<code>\gmu@ifxany</code> , 1136, 1454, 1516, 1765, 1933, 1977, 2033, 2072, 3041, 3091, 3514, 7417, 7425, 7438, 8115, 9213	<code>\gmu@nocite@ampulex</code> , 10288, 10300	<code>\gmu@Require@mw</code> , 13795
<code>\gmu@ifxnone</code> , 1137, 1512	<code>\gmu@notif</code> , 1057, 1073, 1135, 3054, 6079, 7572, 12088	<code>\gmu@Require@notonlypream</code> , 13794
<code>\gmu@InterruptEnvir</code> , 9493	<code>\gmu@notprerr</code> , 10258, 10269	<code>\gmu@Require@parts</code> , 13797
<code>\gmu@lastCASE</code> , 1132, 1257, 1838, 2230, 2506, 2967, 6138, 8040	<code>\gmu@notsw</code> , 1073	<code>\gmu@Require@RCS</code> , 13799
<code>\gmu@LC@InnerLeft</code> , 8500, 8503, 8507, 8510	<code>\gmu@numeratorkern</code> , 10832, 10880, 12137	<code>\gmu@Require@relsize</code> , 13791
<code>\gmu@LC@InnerRight</code> , 8501, 8503, 8508, 8510	<code>\gmu@OR</code> , 4613	<code>\gmu@Require@typos</code> , 13796
<code>\gmu@LC@LetInners</code> , 8491, 8558	<code>\gmu@PackOptionX</code> , 5165, 5201, 9091, 9296, 9516, 9599, 9986, 10218, 10308, 10776, 13278, 13445, 13741	<code>\gmu@Require@url</code> , 13798
<code>\gmu@LC@resa</code> , 8547, 8555	<code>\gmu@pageremain</code> , 4836, 8455, 8467	<code>\gmu@resa</code> , 12038, 12039, 12041
<code>\gmu@leftskipcorr</code> , 12771	<code>\gmu@passbraced</code> , 1109, 4406, 6250, 6408, 6412, 6416, 6963	<code>\gmu@reserved@firstofmany</code> , 1140, 1152
<code>\gmu@letspace</code> , 1733, 2928, 2953, 2956, 3002, 3084, 6920, 6938	<code>\gmu@passbracedNotSp</code> , 4434, 6412	<code>\gmu@reserveda</code> , 2925, 2927
<code>\gmu@llapcomb</code> , 5056	<code>\gmu@PDFdetector</code> , 3809, 3819	<code>\gmu@restorespecials</code> , 3190
<code>\gmu@lowstar</code> , 2843, 2864, 8411, 13126	<code>\gmu@pdfdetector</code> , 3806, 3818	<code>\gmu@restoreUpUpUp</code> , 13501, 13502
<code>\gmu@lowstarfake</code> , 2844, 8421, 8427	<code>\gmu@peep@hash</code> , 2965, 2974	<code>\gmu@RPfor</code> , 12022, 12036, 12047, 12058
	<code>\gmu@peep@next</code> , 2939, 3001, 3020, 3037, 3114, 6908	<code>\gmu@SC@currname</code> , 8715, 8722, 8727, 8730
	<code>\gmu@peepnext@inner</code> , 2952, 2971	<code>\gmu@SC@grouplevel</code> , 8620, 8642, 8645, 8707, 8708, 8711, 8713, 8717, 8720
	<code>\gmu@pifdetokens</code> , 3949	<code>\gmu@SC@resa</code> , 8641, 8648
	<code>\gmu@pifedetokens</code> , 3953	<code>\gmu@SC@scope</code> , 8644, 8646, 8648
	<code>\gmu@pifutokens</code> , 3951	<code>\gmu@SC@setgrouplevel</code> , 8617, 8639, 8706
	<code>\gmu@predetokstring</code> , 1178, 1205, 1206	<code>\gmu@SC@StorageName</code> , 8607, 8642, 8717
	<code>\gmu@prependtoks@aux</code> , 2425	<code>\gmu@scalar</code> , 12206, 12210, 12211, 12217
		<code>\gmu@scalematchX</code> , 12196, 12208, 12231
		<code>\gmu@scapLetters</code> , 12168, 12178, 12183
		<code>\gmu@scapSpaces</code> , 12181, 12186, 12190

<code>\gmu@scapss</code> , 12189, 12194	<code>\gmu@straightensec</code> ,	<code>\gmu@theskewchar</code> , 10992,
<code>\gmu@scscale</code> , 12225, 12230	10749, 10758	11025, 11038, 11050,
<code>\gmu@sec</code> , 10743, 10754, 10755	<code>\gmu@stripchar</code> , 10996,	11062
<code>\gmu@secini</code> , 10705, 10727,	10999, 11358	<code>\gmu@thickmuskip</code> , 13560,
10733, 10739, 10752	<code>\gmu@stripcomma</code> , 11862	13566
<code>\gmu@secmarkh</code> , 10735	<code>\gmu@targetheading</code> ,	<code>\gmu@thinmuskip</code> , 13558,
<code>\gmu@secstar</code> , 10711,	10332, 10335	13564
10721, 10728, 10740,	<code>\gmu@tempa</code> , 991, 2793,	<code>\gmu@thou@fiver</code> , 12888,
10754, 10755	2795, 2858, 2867,	12891, 12899
<code>\gmu@secx</code> , 10743, 10744	2871, 2881, 3653,	<code>\gmu@thou@i</code> , 12893, 12915,
<code>\gmu@secxx</code> , 10710, 10738,	3655, 3798, 3800,	12923
10745	3802, 3820, 3850,	<code>\gmu@thou@ii</code> , 12893,
<code>\gmu@septify</code> , 3193, 11506,	3856, 7173, 7194,	12915, 12923
12489	7870, 7878, 7880,	<code>\gmu@thou@o</code> , 12893, 12915,
<code>\gmu@setbasefont</code> , 11866,	7900, 8066, 8069,	12923
11868	8877, 8880, 8887,	<code>\gmu@thou@put</code> , 12892,
<code>\gmu@setheading</code> , 10590,	8890, 9254, 9266,	12896, 12912
10596, 10597	10964, 10966, 10970,	<code>\gmu@thou@putter</code> , 12895,
<code>\gmu@SetPagegoal</code> , 4859, 4885	10971, 11016, 11030,	12906, 12913, 12920,
<code>\gmu@SetPagegoalGlobal</code> ,	11034, 11041, 11046,	12928
4870	11053, 11057, 11067,	<code>\gmu@thousep</code> , 12925, 12933
<code>\gmu@setsetSMglobal</code> ,	11086, 11092, 11093,	<code>\gmu@tilde</code> , 11895, 11910
3295, 3300, 3386, 3430	11095, 11096, 11100,	<code>\gmu@totalht</code> , 4904
<code>\gmu@setSMglobal</code> , 3302,	11106, 11145, 11148,	<code>\gmu@TrashSkip</code> , 9049, 9058
3304, 3322	11169, 11171, 11172,	<code>\gmu@tri@let@scpref</code> ,
<code>\gmu@smarttilde</code> , 13062,	11175, 11180, 11569,	3559, 3562, 3563
13080	11572, 11574, 11618,	<code>\gmu@twostring</code> , 7321, 7331
<code>\gmu@smashbox</code> , 8759, 8764,	11622, 11638, 11642,	<code>\gmu@unless</code> , 1086, 2143,
8767	11689, 11699, 11716,	2173, 5012, 5023,
<code>\gmu@SMdo@scope</code> , 3479,	11725, 11730, 11731,	5048, 5063
3481, 3484, 3485, 3500	11732, 11734, 11742,	<code>\gmu@url@flexbreak</code> ,
<code>\gmu@SMdo@setscope</code> , 3477,	11743, 11744, 11745,	13674, 13677
3483, 3498	11746, 11748, 11753,	<code>\gmu@url@rigidbreak</code> , 13672
<code>\gmu@SMglobalfalse</code> , 3249,	11754, 11758, 11759,	<code>\gmu@urlbreakable</code> , 13635,
3270, 3304, 3313,	11772, 11778, 11783,	13677
3342, 3373, 3410,	11787, 11793, 11800,	<code>\gmu@UrlFix</code> , 13486, 13491,
3423, 3487	11876, 11879, 12240,	13505
<code>\gmu@SMglobaltrue</code> , 3210,	12242, 12243, 12244,	<code>\gmu@UrlSetup</code> , 13527, 13573
3302	12738, 12739, 12750,	<code>\gmu@VeryBadBadness</code> ,
<code>\gmu@smtempa</code> , 3254, 3272,	12751, 12753, 12754,	4911, 4917, 4918
3367, 3372	12921, 12925, 13587,	<code>\gmu@WHL@arg</code> , 8083, 8086,
<code>\gmu@star@loop</code> , 13119,	13590, 13592, 13595,	8116
13123, 13128	13605, 13612	<code>\gmu@whonly</code> , 13340, 13341
<code>\gmu@storeifnotyet</code> , 3355,	<code>\gmu@tempb</code> , 2860, 2875,	<code>\gmu@xedefracplain</code> ,
8940, 11118, 11248,	3856, 10965, 10966,	10808, 10862
11253, 11395, 11418,	10971, 11724, 11739,	<code>\gmu@xedefracstar</code> ,
11436, 11459, 11470,	11751, 11773, 11774,	10808, 10823
11540	11777, 11778, 13589,	<code>\gmu@xefraccddef</code> , 10824,
<code>\gmu@storemacros@resa</code> ,	13594, 13607	10838, 10839, 10840,
3309, 3312, 3315	<code>\gmu@tempc</code> , 11723, 11726	10841, 10842, 10843,
<code>\gmu@storeprefix</code> , 3242,	<code>\gmu@tempd</code> , 11727, 11792,	10844, 10845, 10846,
3248, 3256, 3341,	11795, 11796, 11797	10847, 10848, 10849,
3351, 3363, 3370,	<code>\gmu@testtopenalty</code> , 2268,	10850, 10851, 10852
3405, 3406, 3415,	2272	<code>\gmu@xiieschar</code> , 1829,
3418, 3446, 3448,	<code>\gmu@theendlinechar</code> ,	1945, 1961, 2001,
3480, 3502, 8611	13529, 13536	2017, 2043, 2057,
<code>\gmu@storespecials</code> , 3187		2079, 2092, 2110,

2123, 2140, 2175,  
 2187, 2208, 2220  
 \gmuIfNoValueF, 7477, 7484  
 \gmuIfNoValueT, 7475, 7487  
 \gmuIfNoValueTF, 7463,  
 7475, 7477, 7479  
 \gmuIfValueF, 7487, 10618,  
 12543, 12568  
 \gmuIfValueT, 7481, 7484,  
 8236, 8357, 8907,  
 8911, 8932, 9882,  
 9884, 9885, 9886,  
 9887, 11009, 11020,  
 11028, 11032, 11065,  
 11143, 11213, 11496,  
 11691, 12298, 12303,  
 12544, 12551, 12569,  
 12577, 12692, 12723,  
 12849, 12867, 12876,  
 12982, 13557  
 \gmuIfValueTF, 7479, 7888,  
 7904, 8458, 8622,  
 8879, 8889, 8920,  
 8922, 10912, 11017,  
 11058, 11085, 11126,  
 11129, 11168, 12692,  
 12724, 12785, 12817,  
 12861, 12953, 13064,  
 13164  
 \gmuPutIfValue, 7481,  
 11019, 11022, 11024,  
 11026, 11035, 11036,  
 11037, 11039, 11042,  
 11047, 11048, 11049,  
 11051, 11060, 11061,  
 11063, 11068, 11215,  
 12543, 12552, 12568,  
 12578  
 \gmuXeTeXtrue, 3609  
 \gmv@hyphen, 9737  
 \gmv@hyphenchar, 9744, 13541  
 \gn@melet, 3552, 3589, 3590  
 \gobble, 971, 973, 7738, 11472  
 \gobblespace, 5935, 6040,  
 6110, 7533, 8886,  
 12735, 12746  
 \gobbletwo, 974  
 \gprependtotoks, 2457  
 \grab@LostWax, 7976, 7981  
 \grab@optparam, 6122,  
 6126, 6273, 6392  
 \grab@prefix, 5986, 6429  
 \grab@prefix@CASE, 6425,  
 6442, 6446, 6450,  
 6454, 6458, 6462,  
 6466, 6471, 6477,  
 6482, 6498  
 \grab@twoparams, 6140, 6383  
 \greater@dim, 4240  
 \grefstepcounter, 2345, 2361  
 \grelaxen, 1374, 10517  
 \GrossI, 6467  
 \GrossIgnore, 6467  
 \GrossIgnoreStop, 6472  
 \GrossIStop, 6473  
 \GrossL, 6478  
 \GrossLangStop, 6483  
 \GrossLong, 6478  
 \GrossLStop, 6484  
 \GroteL, 6478  
 \GroteLang, 6478  
 \GroteLangStop, 6483  
 \GroteLStop, 6484  
 \GroteN, 6467  
 \GroteNegere, 6467  
 \GroteNegereStop, 6472  
 \GroteNStop, 6473  
 \hangafter, 1991  
 \hangindent, 1940, 13066  
 \hash, 9773  
 \hathat, 9770  
 \hbadness, 1979  
 \hbox, 4671, 4673, 4774,  
 8422, 8805, 9690,  
 9998, 10007, 10035,  
 10073, 10078, 10100,  
 10101, 10109, 10138,  
 10139, 10151, 10152,  
 10162, 10168, 10170,  
 10171, 10181, 10182,  
 10186, 10187, 10191,  
 10864, 10876, 11397,  
 11402, 11407, 11413,  
 11420, 11421, 11437,  
 11439, 11443, 11461,  
 11462, 11536, 11542,  
 11543, 11547, 11548,  
 11724, 11739, 11751,  
 11896, 12136, 12458,  
 12749, 12952, 13514,  
 13589, 13594, 13600,  
 13607, 13661  
 \hboxreflected, 4773  
 \HeadingNumber, 10394, 10396  
 \HeadingNumberedfalse,  
 10340, 10365  
 \HeadingRHeadText, 10378  
 \HeadingText, 10380  
 \HeadingTOCText, 10379  
 \HeShe, 13821  
 \heshe, 13816  
 \hfil, 9776, 13249, 13266  
 \hfillneg, 11936  
 \hfilneg, 9776  
 \hfuzz, 1936  
 \hgrefstepcounter, 2360  
 \hidden@iffalse, 2323  
 \hidden@iftrue, 2324  
 \HimHer, 13823  
 \himher, 13818  
 \HisHer, 13822  
 \hisher, 13817  
 \HisHers, 13824  
 \hishers, 13819  
 \hoffset, 1940  
 \holdinginserts, 1985  
 \hrule, 4788, 4989, 10564  
 \hrule@zero, 4788, 4989  
 \hsize, 1936, 4673, 12714,  
 12752, 12789, 12809,  
 13040, 13044, 13045,  
 13065  
 \hsizecline, 4673  
 \hskip, 3843, 4149, 4744,  
 9054, 9723, 11569,  
 11570, 11576, 11600,  
 11936, 12161, 12346,  
 12353, 12355, 12356,  
 12363, 12364, 12375,  
 12382, 12421, 12428,  
 12788, 12874, 13674,  
 13675, 13713, 13716,  
 13720, 13723  
 \hss, 4671, 4673, 11725,  
 11727, 11739, 11751,  
 13589, 13594, 13607  
 \ht, 4233, 4235, 4905, 8792,  
 8808, 8849, 8861,  
 10041, 10084, 10170,  
 10181, 10186, 11414,  
 11440, 11444, 11537  
 \Huge, 9555, 9565, 9566  
 \huge, 9554, 9565, 11942  
 \hunskip, 2368, 11968,  
 11970, 11972, 12700,  
 12737, 12748  
 \HyOrg@maketitle, 12994,  
 12995  
 \hypertarget, 10333  
 \HyphenateLogo, 10206  
 \hyphenchar, 4752, 4753,  
 8927, 8928, 9742,  
 9743, 12659, 13533  
 \hyphenpenalty, 1980,  
 9683, 12421, 12667  
 \IamDeclaredGlobally, 7689  
 \if, 1950, 1951, 1952, 1953,  
 1954, 1961, 1962,  
 1963, 1964, 1965,



1966, 2004, 2005,	<code>\if@dc@xadefault@ii@,</code>	10177, 10549, 10577,
2006, 2007, 2008,	5798, 6389	11651, 11653, 11795,
2017, 2018, 2019,	<code>\if@dc@xaeacher@,</code>	12661, 13065, 13066,
2020, 2021, 2022,	<code>\if@debugacro@,</code>	13069
2046, 2047, 2048,	<code>\if@filesw,</code>	<code>\iffalse,</code>
2049, 2057, 2058,	11956, 13305,	1246, 1258, 2289,
2059, 2060, 2061,	13317, 13325, 13362,	2291, 2312, 2314,
2082, 2083, 2084,	13373, 13402	2323, 2437, 3629,
2085, 2092, 2093,	<code>\if@gmu@mmhbox,</code>	11160, 13393
2094, 2095, 2096,	10867, 10871, 12141,	<code>\IfFileExists,</code>
2113, 2114, 2115,	12959	4783, 5179,
2116, 2124, 2125,	<code>\if@HyOrg@maketitle@ampulexed@iffontchar,</code>	5187
2126, 2127, 2145,	12993, 13001	2844, 3620,
2146, 2147, 2148,	<code>\if@mainmatter,</code>	10121, 10825, 11124,
2149, 2150, 2151,	10340	11127, 11130, 11189,
2156, 2157, 2158,	<code>\if@nobreak,</code>	11199, 11301, 11302,
2159, 2160, 2161,	10518	11355, 11409, 11423,
2162, 2175, 2176,	<code>\if@openright,</code>	11426, 11430, 11431,
2177, 2178, 2179,	10342	11432, 13583, 13599
2180, 2181, 2182,	<code>\if@partsw,</code>	<code>\ifgmu@debug@msgs,</code>
2187, 2188, 2189,	13309, 13366	3682
2190, 2191, 2192,	<code>\if@specialpage,</code>	<code>\ifgmu@ifquant,</code>
2193, 2194, 2211,	10383	1450
2212, 2220, 2221,	<code>\if@tempswa,</code>	<code>\ifgmu@postsec,</code>
2222, 2786, 3146,	2234, 13315,	10575, 10583
3312, 3315, 3806,	13371	<code>\ifgmu@quiet,</code>
3809, 4845, 5174,	<code>\if@twocolumn,</code>	3343, 3364
7200, 7840, 7899,	10405,	<code>\ifgmu@SMglobal,</code>
8563, 9889, 10006,	13008, 13009, 13012	3208,
10008, 10060, 10382,	<code>\if@twoside,</code>	3247, 3255, 3301,
10385, 10401, 10404,	10410	3340, 3368, 3404,
10406, 10421, 10425,	<code>\if@XeTeX,</code>	3416, 3484
10541, 10544, 11090,	3629, 7128	<code>\ifgmuXeTeX,</code>
11154, 11301, 11354,	<code>\ifAllClauses,</code>	3608, 4999
11431, 11942, 13583	4626, 4646	<code>\ifHeadingNumbered,</code>
<code>\if@afterindent,</code>	<code>\IfAmong,</code>	10364, 10392
10521	1432, 1821, 7498	<code>\ifhmode,</code>
<code>\if@CofacPrzedData@,</code>	<code>\ifAnyClause,</code>	2361, 2368, 9054,
12602, 12616	4598, 4618	11600, 12361, 12374,
<code>\if@dc@alllong@,</code>	<code>\ifcase,</code>	12405, 13096, 13099
5743,	9563, 12546,	<code>\ifincsname,</code>
5884, 6486, 6487	12570, 12914, 12922	1079, 2634
<code>\if@dc@BareSpace@,</code>	<code>\ifcat,</code>	<code>\ifinner,</code>
5769	887, 898, 926, 2741,	12855, 12866
<code>\if@dc@GroteLang@,</code>	2826, 3561, 3567, 12170	<code>\IfIntersect,</code>
5884, 5926, 6486	<code>\ifcsname,</code>	1754
<code>\if@dc@GroteNegere@,</code>	203, 232, 250,	<code>\IfIntersect@next,</code>
5763, 5925	255, 260, 809, 2541,	1779, 1783
<code>\if@dc@ignore@,</code>	3262, 3405, 3415,	<code>\IfIs,</code>
5762, 5925	3446, 3572, 3706,	1856, 8823
<code>\if@dc@long@,</code>	3712, 3720, 4824,	<code>\IfIsExpression,</code>
5767, 5777,	4830, 4954, 4969,	1914, 2260
5926, 6487	5222, 8051, 8054,	<code>\IfLong,</code>
<code>\if@dc@Loop@addall,</code>	8521, 8722, 9399,	p. 80, 7489
6861	10780, 11144, 12773,	<code>\ifmmode,</code>
<code>\if@dc@Loop@drainall,</code>	13436	2636, 4764, 9056,
6862	<code>\ifdate,</code>	9543, 9627, 9919,
<code>\if@dc@Loop@iterall,</code>	12586, 12600	10864, 10876, 11460,
6860	<code>\IfDCMessages,</code>	12502, 12952, 13218,
<code>\if@dc@Mmode@,</code>	p. 81	13634, 13659, 13660,
5888	<code>\ifDCMessages,</code>	13666
<code>\if@dc@quiet@,</code>	5749	<code>\ifnum,</code>
5746	<code>\ifdefined,</code>	114, 873, 1151,
<code>\if@dc@xadefault,</code>	795, 804, 829,	1204, 1233, 1393,
5780	982, 1017, 2256,	1425, 2123, 3805,
<code>\if@dc@xadefault@i@,</code>	2257, 2605, 2676,	3912, 3998, 4008,
5797, 6387	3599, 3745, 4917,	4020, 4031, 4039,
	5737, 9735, 9737,	4048, 4058, 4069,
	9744, 9924, 9945,	4187, 4211, 4600,
	11247, 11252, 11482,	4632, 4752, 4795,
	12048, 12401, 12475,	4910, 5129, 8612,
	12732, 12994, 13021,	8645, 8718, 8720,
	13134, 13520, 13541,	8927, 9253, 9281,
	13551, 13786	
	<code>\ifdim,</code>	
	4190, 4241, 4837,	
	4896, 8453, 8467,	
	8475, 8849, 10045,	

9561, 9897, 10173,  
 10174, 10365, 11677,  
 11996, 12171, 12226,  
 12227, 12242, 12243,  
 12244, 12323, 12514,  
 12955, 13095, 13125,  
 13237, 13255, 13296,  
 13355, 13631, 13657  
 \ifodd, 4939  
 \ifSecondClass, 3737  
 \ifsingletoken, 1147  
 \ifstribs, 1232  
 \ifstrings, 1203  
 \ifStrX, 1244  
 \iftrue, 1250, 1255, 2290,  
 2313, 2324, 3629,  
 10297, 10583, 13388  
 \ifvbox, 8841  
 \ifvmode, 4293, 9052,  
 12392, 12435, 12503  
 \ifvoid, 8860  
 \ifx, 213, 784, 947, 953,  
 983, 1473, 1479,  
 1494, 2264, 2265,  
 2266, 2267, 2470,  
 2738, 2783, 2797,  
 2912, 3157, 3600,  
 4187, 4190, 4191,  
 4269, 4955, 4970,  
 5854, 7437, 7471,  
 7737, 8519, 8520,  
 9199, 9546, 9547,  
 9548, 9549, 9550,  
 9551, 9552, 9553,  
 9554, 9555, 9682,  
 9683, 9699, 10020,  
 10238, 10295, 10453,  
 10464, 10711, 10963,  
 10966, 10971, 11121,  
 11809, 11813, 11817,  
 11829, 11839, 11850,  
 12038, 12077, 12169,  
 12182, 12185, 12210,  
 12901, 12913, 12920,  
 13098, 13303, 13313,  
 13341, 13369, 13603  
 \IgnInfo, 4478, 6004, 6507  
 \ignoreactiveM, 2905, 2913  
 \IgnorePars, 9025  
 \ignorespaces, 9640,  
 11968, 11970, 11972,  
 12244, 12353, 12356,  
 12364, 12375, 12382,  
 12428, 13027  
 iI, p. 80  
 \ikern, 12161  
 \IMHO, 12118  
 \immediate, 4460, 4492,  
 11957, 13306, 13318,  
 13319, 13326, 13363,  
 13374, 13375, 13403,  
 13428  
 \in, 11297, 11555  
 \inclasthook, 13322,  
 13345, 13399  
 \include, 13295, 13297  
 \includecountfix, 13426  
 \incsddef, 4336, 13147,  
 13149, 13150  
 \indent, 13024  
 \infty, 11272  
 \input, 134  
 \inputlineno, 4460  
 \insertpenalties, 1978  
 \interlinepenalty, 1983  
 \iota, 11336  
 \it, 9651, 11633  
 \itemindent, 13244, 13262  
 itemize\*, [13254](#)  
 \iteracro, 12062, 12073  
 \itshape, 10121, 11014,  
 11045, 11611, 11651,  
 12621, 12642  
 \jobname, 114, 3900, 11955,  
 11957  
 \justified, 12676  
 \kappa, 11337  
 \keepsilent, 295  
 \kern, 4148, 4743, 5020,  
 5030, 5046, 5051,  
 5066, 8422, 9039,  
 9041, 9998, 10006,  
 10008, 10039, 10042,  
 10054, 10062, 10077,  
 10082, 10087, 10090,  
 10091, 10100, 10101,  
 10103, 10104, 10109,  
 10110, 10112, 10122,  
 10136, 10138, 10139,  
 10149, 10151, 10152,  
 10165, 10168, 10192,  
 10194, 10880, 10881,  
 11207, 11414, 11433,  
 11536, 11537, 11544,  
 11545, 11549, 11550,  
 11551, 11553, 11569,  
 11576, 11605, 11606,  
 11910, 12135, 12337,  
 12458, 12468, 12671,  
 12772, 12774, 12786,  
 12797, 12855, 12873,  
 13531, 13698  
 \KV@gmutils.gmd#1, [5170](#)  
 \KV@gmutils.gmd@gm#1, [5188](#)  
 \KV@gmutils.gmd@quiet, [5157](#)  
 \labelsep, 13246, 13264  
 \labelwidth, 13245, 13246,  
 13263, 13264  
 \Lambda, 11314  
 \lambda, 11338  
 \langle, 9626, 11281, 13597  
 \language, 1988  
 \LARGE, 9551, 9565, 10623  
 \Large, 9550, 9565, 10631  
 \large, 9549, 9565, 10639  
 \larger, p. 138, 9580,  
 11400, 11405, 11446,  
 11447, 11450, 11451,  
 11452, 11453  
 \largerr, p. 138, 9584,  
 11448, 11449  
 \lastbox, 10522  
 \lastcentered, 3826  
 \lastkern, 9038  
 \lastlinefit, 1994, 3827,  
 3838  
 \lastnodetype, 4795  
 \lastskip, 9032, 10577  
 \LaTeX, 8380, 9995, 10032,  
 10062, 10064, 10130,  
 10153  
 \LaTeXe, 9996, 10060  
 \LaTeXpar, 10071  
 \lccode, 2794, 4304, 8410,  
 12171, 13562, 13628,  
 13654  
 \lcfirstofone, 4305, 4308  
 \ldate, 12636, 12646  
 \le, 11238, 11239, 11240,  
 11250  
 \leavevmode, 2361, 4148,  
 4149, 4743, 5019,  
 8422, 10074, 10100,  
 10163, 11604, 11717,  
 11735, 11749, 11896,  
 12024, 12135, 12374,  
 12381, 12428, 13707  
 \leeng, 11240  
 \leftarrow, 11286, 11551  
 \lefthyphenmin, 1989  
 \leftline, 12785, 12859  
 \leftmargin, 13243, 13261  
 \leftrightarrow, 11288,  
 11433  
 \leftskip, 2036, 3829,  
 3841, 11613, 12633,  
 12660, 12677, 12714,  
 12752, 12772, 12786,

12789, 12805, 12817,	11239, 11240, 11242,	1405, 1424, 1432,
12855, 12873, 13045,	11243, 11249, 11250,	1444, 1454, 1512,
13065	11254, 11255, 11275,	1607, 1611, 1627,
<code>\leftslanting</code> , 11663, 11665	11322, 11372, 11469,	1640, 1651, 1666,
<code>\leftslanting@</code> , 11597,	11483, 11484, 11665,	1678, 1682, 1686,
11663, 11665	11708, 11866, 11868,	1709, 1715, 1741,
<code>\leq</code> , 11239, 11248, 11249,	11877, 11998, 12123,	1797, 1804, 1845,
11250	12148, 12196, 12211,	2105, 2260, 2416,
<code>\let</code> , 107, 108, 139, 140,	12231, 12453, 12459,	2467, 2492, 2511,
270, 271, 272, 273,	12506, 12628, 12638,	2521, 2534, 2717,
274, 275, 276, 277,	12646, 12682, 12819,	2723, 2729, 2769,
278, 785, 830, 834,	12832, 12925, 13030,	2820, 2825, 2868,
835, 939, 973, 974,	13080, 13190, 13198,	2889, 2898, 2911,
1132, 1359, 1360,	13204, 13316, 13335,	2996, 3015, 3033,
1365, 1368, 1369,	13372, 13385, 13389,	3048, 3063, 3078,
1374, 1468, 1477,	13406, 13412, 13534,	3102, 3112, 3122,
1507, 1729, 1776,	13677	3134, 3145, 3163,
1779, 1942, 1997,	<code>\licenseNoteLeaf</code> , 288	3175, 3350, 3445,
2040, 2076, 2264,	<code>\lim</code> , 11363	3478, 3499, 3554,
2265, 2266, 2267,	<code>\linebreak</code> , 12819	3583, 3588, 3638,
2289, 2302, 2312,	<code>\linedate</code> , 12592, 12628,	3648, 3759, 3883,
2320, 2346, 2378,	12639	3906, 4104, 4154,
2558, 2562, 2567,	<code>\linedate@hook</code> , 12595, 12623	4251, 4276, 4308,
2651, 2677, 2681,	<code>\linepenalty</code> , 1979	4313, 4322, 4327,
2739, 2742, 2744,	<code>\lineskip</code> , 2034	4336, 4406, 4434,
2749, 2784, 2787,	<code>\lineskiplimit</code> , 1937	4449, 4478, 4528,
2798, 2800, 2807,	<code>\list</code> , 13242, 13260	4534, 4562, 4598,
2864, 2925, 2927,	<code>\listparindent</code> , 13247, 13265	4613, 4626, 4643,
2928, 2950, 2956,	<code>\lit</code> , 11628	4657, 4660, 4662,
3088, 3177, 3248,	<code>\litcorrection</code> , 11599,	4664, 4665, 4677,
3256, 3296, 3302,	11610, 11617, 11632,	5126, 5840, 5848,
3304, 3341, 3369,	11637	6297, 6340, 6348,
3387, 3406, 3408,	<code>\litdimen</code> , 11598, 11600,	6355, 6362, 6367,
3417, 3421, 3431,	11605, 11606	6372, 6376, 6383,
3480, 3481, 3484,	<code>\litkern</code> , 11601, 11606	6396, 6404, 6429,
3485, 3501, 3513,	<code>\litshape</code> , 11608, 11626,	6443, 6520, 6574,
3515, 3532, 3536,	11628, 11653	6591, 6603, 6617,
3562, 3563, 3691,	<code>\liturgiques</code> , 12021	6639, 6674, 6865,
3701, 3846, 4144,	<code>\llap</code> , 5062, 12458, 12468,	6895, 6968, 6978,
4172, 4256, 4388,	13148	6988, 7001, 7021,
4400, 4971, 4996,	<code>\LoadPackagePolski</code> , 4982	7040, 7111, 7141,
5884, 5925, 5926,	<code>\long</code> , 109, 111, 112, 839,	7185, 7302, 7310,
6412, 6486, 6487,	841, 844, 847, 862,	7321, 7324, 7348,
6536, 6755, 6758,	884, 895, 906, 915,	7364, 7397, 7410,
6925, 6943, 7407,	920, 938, 942, 971,	7424, 7463, 7475,
7408, 7446, 7447,	977, 981, 990, 1010,	7477, 7479, 7481,
7449, 7484, 7487,	1042, 1057, 1086,	7489, 7504, 7526,
7529, 7533, 7682,	1113, 1127, 1140,	7542, 7547, 7558,
7690, 8050, 8099,	1147, 1178, 1203,	7644, 7645, 7749,
8120, 8246, 8518,	1232, 1244, 1273,	7762, 7772, 7827,
8590, 8644, 8646,	1275, 1277, 1282,	7838, 7867, 7899,
8894, 9502, 9679,	1283, 1284, 1285,	7917, 8049, 8078,
9751, 9917, 9995,	1286, 1293, 1296,	8093, 8136, 8223,
9996, 10118, 10207,	1298, 1304, 1306,	8269, 8491, 8513,
10396, 10583, 10735,	1331, 1334, 1336,	8530, 8574, 8607,
10758, 10859, 10907,	1340, 1341, 1342,	8688, 8804, 9104,
10920, 11119, 11217,	1349, 1351, 1385,	9108, 9130, 9154,

9182, 9220, 9229,	7932, 9280, 9281,	<code>\mathit</code> , 11029
9260, 9278, 9369,	9505, 11696, 12009,	<code>\mathop</code> , 11207, 11278, 11397
9423, 9765, 9811,	13530	<code>\mathopen</code> , 11262, 11264,
10355, 13011, 13020	<code>\m@th</code> , 9964, 10060, 10905,	11281, 11446, 11448,
<code>\longafterfi</code> , 1330, 1332	13659, 13661, 13666	11450, 11452, 11464
<code>\longpauza</code> , 12451, 12455,	<code>\macro</code> , 3654	<code>\mathord</code> , 11227, 11228,
12465, 12466	<code>\mag</code> , 1991	11229, 11230, 11231,
<code>\loop</code> , 5935, 8043, 8710	<code>\makeatletter</code> , 1353, 3212,	11232, 11233, 11234,
<code>\looseness</code> , 1985, 12324,	9286, 11690, 11953	11235, 11236, 11271,
12335, 12692, 12724	<code>\makeblanksignored</code> , 2691,	11272, 11277, 11283,
<code>\Lost</code> , 7926, 7982	2697	11303, 11304
<code>\lost</code> , 7926, 7982, 8793	<code>\MakeFalseIfDefined</code> , 4823	<code>\mathpunct</code> , 11258, 11259,
<code>\lostwax</code> , 5935, 6130, 7918,	<code>\makelabel</code> , 13249, 13266	11260, 11261
7946, 7975, 8037, 8793	<code>\MakePrivateLetters</code> ,	<code>\mathrel</code> , 11226, 11238,
<code>\lostwax@NoVal</code> , 7983, 7992	3212, 3225, 3288,	11241, 11244, 11245,
<code>\lower</code> , 9998, 10007, 10100,	3332, 3378, 3397,	11268, 11273, 11274,
10138, 10139, 10151,	3427, 3441, 8602,	11279, 11280, 11285,
10152, 10167, 11401,	8655, 8669, 8683	11286, 11287, 11288,
11406, 11444, 11549,	<code>\makestarlow</code> , 8409	11291, 11295, 11297,
11737, 11896	<code>\maketitle</code> , 12992	11433, 11473, 11545,
<code>\lowercase</code> , 2796, 4309,	<code>\MakeTrueIfDefined</code> , 4829	11551, 11552, 11553,
5174, 6301, 6548,	<code>\MakeUppercase</code> , 12172	11554, 11558
6549, 8410, 13562,	<code>\mapsto</code> , 11544	<code>\mathrm</code> , 11066, 11207, 11408
13628, 13654	<code>\marg</code> , 9821, 9858, 9886	<code>\mathsurround</code> , 1939
<code>\lozenge</code> , 11303	<code>maszynopis</code> , 12658	<code>\mathversion</code> , 11213
<code>\lpauza</code> , 12380	<code>\math@arg</code> , 9917, 9919	<code>\max</code> , 8791, 8796, 11361
<code>\lpdef</code> , 938, 939, 1107,	<code>\math@fontsfalse</code> , 10037,	<code>\maxdeadcycles</code> , 1991
1692, 1694, 1696,	10080	<code>\maxdepth</code> , 1937
1698, 1768, 2394,	<code>\mathalpha</code> , 11311, 11312,	<code>\maxdimen</code> , 4837, 4863,
2422, 2429, 2450,	11313, 11314, 11315,	4874, 4896, 8475,
2457, 2479, 2939,	11316, 11317, 11318,	8797, 8800, 8843, 8863
2974, 2985, 3231,	11319, 11320, 11326,	<code>\mbox</code> , 10060, 10905, 11896,
3232, 3245, 3253,	11327, 11328, 11329,	12987
3288, 3290, 3294,	11330, 11331, 11332,	<code>\meaning</code> , 1847, 1849, 1882,
3308, 3331, 3334,	11333, 11334, 11335,	1897, 2247, 3648,
3335, 3337, 3355,	11336, 11337, 11338,	3721, 3723, 4451,
3362, 3377, 3381,	11339, 11340, 11341,	6961, 7204, 7206,
3385, 3399, 3400,	11342, 11343, 11344,	7774, 7777, 7780,
3402, 3414, 3427,	11345, 11346, 11347,	8086, 8564, 9174,
3429, 3442, 3497,	11348, 11349, 11350,	10957, 10999, 11358,
3523, 4363, 4773,	11351, 11352	11758, 11772, 11823,
5810, 6261, 6918,	<code>\mathbin</code> , 11224, 11225,	11825, 11858
6934, 6959, 7356,	11267, 11269, 11270,	<code>\MeaningOrUnex</code> , 4449,
7361, 7370, 7374,	11276, 11284, 11307,	4466, 4483
9426, 9493, 10002	11308, 11309, 11424,	<code>\medmuskip</code> , 4764, 13559,
<code>\lsl</code> , 11631	11427, 11467, 11468,	13565, 13574
<code>\LuaTeX</code> , 10198	11546, 11555	<code>\medskipamount</code> , 11931,
<code>luzniej</code> , 12328	<code>\mathcal</code> , 10100, 10101	11932, 12626
<code>luzniej*</code> , 12333	<code>\mathchar@type</code> , 11087,	<code>\MessageBreak</code> , 9571, 9576,
<code>\luzniejcore</code> , 12320, 12328	11102, 11202, 11211,	10483, 10484
<code>\lv</code> , 13148	11491	<code>\meta</code> , 9618, 9663, 9865,
<code>\m@ne</code> , 1278, 2288, 2311,	<code>\mathchoice</code> , 11397, 11420,	13714, 13721
3695, 3696, 3697,	11460, 11542	<code>\meta@font@select</code> , 9631,
3698, 3699, 3760,	<code>\mathclose</code> , 11263, 11265,	9651
4387, 6185, 6758,	11282, 11447, 11449,	<code>\meta@fontsetting</code> , 9626,
7032, 7052, 7084,	11451, 11453, 11465	9636, 9649, 9651, 9666
	<code>\mathfrak</code> , 12058	<code>\metaat</code> , 13712, 13727

<code>\metachar</code> , 9665, 9883	7779, 7780, 7964,	3682, 3737, 5223,
<code>\metacharfont</code> , 9665, 9666	7967, 8518, 8519,	5743, 5746, 5749,
<code>\metadot</code> , 13719, 13730	8520, 8522, 8564, 9497	5762, 5763, 5764,
<code>\MetaPrefix</code> , 143	<code>\NamedInput</code> , 4174, 9286,	5767, 5769, 5780,
<code>\MetaPrefixS</code> , 143	11954	5797, 5798, 5799,
<code>\min</code> , 8791, 11362	<code>\NamedInput@finish</code> , 4172,	5888, 6670, 6860,
<code>\mkern</code> , 11476	4177	6861, 6862, 10781,
<code>\mode@skip</code> , 9033, 9035, 9051	<code>\NamedInput@prepare</code> ,	10871, 12586, 12616,
<code>\month</code> , 1992	4167, 4175	13001
<code>\Msg</code> , 353, 354, 355, 356,	<code>\namemeaning</code> , 3719	<code>\newlinechar</code> , 147, 148, 1991
357, 358, 359, 360, 361	<code>\nameshow</code> , 3705	<code>\newmuskip</code> , 2257
<code>\mskip</code> , 4764, 9056	<code>\nameshowthe</code> , 3711	<code>\newpage</code> , 8444, 10413,
<code>\mu</code> , 11339	<code>\napapierkicore</code> , 12292,	12516, 13089
<code>\muexpr</code> , 2267	12300	<code>\newskip</code> , 2256, 3746, 9049,
<code>\multiply</code> , 10048, 10051,	<code>\napapierkistretch</code> ,	12625, 12834
12223, 12323, 12334,	12290, 12293	<code>\newtoks</code> , 1707, 1795, 2425,
12693, 12725	<code>\narrativett</code> , 9741, 9762,	5738, 5754, 5863, 7850
<code>\mw@getflags</code> , 10539	9821, 9829, 9836,	<code>\newwrite</code> , 11956
<code>\mw@HeadingBreakAfter</code> ,	9840, 9943	<code>\next</code> , 2264, 2265, 2266,
10385, 10406, 10421,	<code>\narra </code>	2267, 2269, 3562,
10425, 10455, 10540	<code>tivett@storedhyphenchar</code> ,	3563, 3568, 3570,
<code>\mw@HeadingBreakBefore</code> ,	9742	3571, 3577, 9881,
10382, 10454, 10541	<code>\nawj</code> , 12337	9883, 9884, 9885,
<code>\mw@HeadingLevel</code> , 10362,	<code>\nazwired</code> , 12519	9886, 9887, 9889,
10365	<code>\ne</code> , 11275	9893, 9908, 13391, 13393
<code>\mw@HeadingRunIn</code> , 10401,	<code>\neb</code> , 11469	<code>\NextBgroup</code> , 13392
10454	<code>\NeedsTeXFormat</code> , 121, 186	<code>\nfss@text</code> , 9627
<code>\mw@HeadingType</code> , 10381,	<code>\neg</code> , 11276, 11475	<code>\nieczer</code> , 12030
10515, 10547, 10548,	<code>\nego</code> , 11277	<code>\nobreak</code> , 11600, 12855
10561	<code>\neq</code> , 11274, 11275, 11468	<code>\nobreakspace</code> , 12189,
<code>\mw@HeadingWholeWidth</code> ,	<code>\neqb</code> , 11468, 11469	13064, 13074
10404, 10455	<code>\newbox</code> , 8759, 8778, 8779	<code>\nocite</code> , 10294
<code>\mw@normalheading</code> , 10408,	<code>\NewCommand</code> , 8295	<code>\noexpand</code> , 108, 835
10417, 10420, 10424,	<code>\newcommand</code> , 1023, 1359,	<code>\nofileparts</code> , 13411
10596	1364, 1368, 1373,	<code>\nohy</code> , 4148
<code>\mw@processflags</code> , 10456	2647, 3187, 3705,	<code>\noindent</code> , 11521, 12855
<code>\mw@runinheading</code> , 10402,	3711, 3719, 4144,	<code>\nolimits</code> , 11412, 11416
10597	4148, 4149, 10012,	<code>\nolinkurl</code> , 13456
<code>\mw@secdef</code> , 10461, 10462,	10237, 10452, 11941,	<code>\none</code> , 6761, 6779, 7122,
10463, 10469	11950, 12006, 12021,	7918, 7927, 8034,
<code>\mw@section</code> , 10460	12023, 12024, 12030,	8035, 8038, 8042
<code>\mw@sectionxx</code> , 10361	12036, 12161, 12239,	<code>NoNumSecs</code> , <u>10328</u>
<code>\mw@secundef</code> , 10465,	12320, 12337, 12511,	<code>\NOO</code> , 136, 313, 358
10477, 10480	12621, 12641, 12711,	<code>\nopagebreak</code> , 12610, 12827
<code>\mw@setflags</code> , 10466	13088, 13339	<code>\NoPredate</code> , 12619
<code>\n@melet</code> , 3540, 3584, 3585,	<code>\newcount</code> , 4361, 5759,	<code>\normalfont</code> , 9738, 12987,
7517, 8052, 8055,	5760, 6858, 8995,	13030
10491, 10493, 10752,	9491, 12318	<code>\normalsize</code> , 9546, 9564,
10828	<code>\newcounter</code> , 10328, 13436	9576, 10646, 10653,
<code>\Name</code> , 200, 249, 804, 805,	<code>\newdef</code> , 1023, 1034, 1035	10658
810, 844, 4794, 4963,	<code>\newdimen</code> , 8780, 8781	<code>\not@onlypreamble</code> , 10237,
4971, 5104, 5223,	<code>\NewEnvironment</code> , 8327	10241, 10242, 10243,
6299, 6302, 7204,	<code>\newenvironment</code> , 12328,	10244, 10245, 10249,
7206, 7339, 7370,	12333, 12516, 12658	10250, 10252, 11374
7374, 7696, 7713,	<code>\newgif</code> , 2287	<code>\NotDeclared</code> , 8566
7722, 7776, 7777,	<code>\newif</code> , 810, 1450, 2137,	<code>\NoValue</code> , 6283, 6329, 6332,
	2304, 3208, 3608,	6337, 6342, 7451,

7471, 7509, 7527,	12571, 12572, 12915,	<code>\parbox</code> , 12788, 12816
8435, 11005, 11075,	12923	<code>\parfillskip</code> , 2037, 3831,
11116, 11210, 11493,	<code>\outer</code> , 7680, 7681, 7682,	3839, 12680, 12681,
12947	7737, 7762, 9108, 9229	12701, 12739, 12754,
<code>\nu</code> , 11340	<code>\outFileName</code> , 216, 219	12818, 13046
<code>\null</code> , 4765, 10411, 11600,	<code>\output</code> , 2073	<code>\parg</code> , 9834
11604	<code>\outputpenalty</code> , 1983	<code>\parindent</code> , 1940, 3747,
<code>\nulldelimiterspace</code> , 1938	<code>\overfullrule</code> , 1936	3828, 3842, 12008,
<code>\number</code> , 3999, 4001, 4002,	<code>\P</code> , 6549	12761, 12874, 13021,
4009, 4011, 4012,	<code>\p@</code> , 10049, 10622, 10624,	13022
4021, 4023, 4024,	10628, 10630, 10632,	<code>\ParseRCSVersion</code> , 13744
4032, 4034, 4035,	10636, 10638, 10640,	<code>\parskip</code> , 2034
4040, 4042, 4043,	10643, 10645, 10647,	<code>\partial</code> , 11283
4049, 4051, 4052,	10652, 10654, 10657	<code>\partmark</code> , 10707
4060, 4062, 4063,	<code>\pack</code> , 317, 336, 337, 338,	<code>\partopsep</code> , 4294, 13243,
4071, 4073, 4074,	339, 340, 341, 342,	13261
9256, 9257, 9815,	343, 344, 345, 346,	<code>\PassOptionsToPackage</code> ,
13129, 13130	347, 348, 349	5158, 8357
<code>\numexpr</code> , 1909, 2264, 3624,	<code>\PackageE</code> , 3783	<code>\pausing</code> , 1985
4001, 4011, 4023,	<code>\PackageError</code> , 1029, 4920,	<code>\pauza</code> , 12360
4034, 4042, 4051,	6243, 6539, 8007,	<code>\pauza@skipcore</code> , 12346,
4062, 4073, 4085,	8041, 8153, 8190,	12419, 12420
4088, 4091, 4094,	8300, 8312, 8332,	<code>\pauzacore</code> , 12347, 12364,
4097, 4100, 4105,	8342, 9479, 10268,	12375, 12382, 12421,
4500, 4510, 4531,	12392, 12435, 13727,	12428, 12453, 12459,
4584, 4816, 4817,	13730	12465, 12468, 12670,
4855, 5976, 6218,	<code>\PackageInfo</code> , 4483, 7575,	12671
6219, 6225, 6232,	7628	<code>\pauzacore@long</code> , 12452,
6747, 7178, 7180,	<code>\PackageWarning</code> , 3344,	12453
7187, 7189, 7957,	3365, 4313, 7620,	<code>\pauzacore@short</code> , 12458,
8452, 8588, 8589,	9169, 9570, 9575, 13495	12459
8594, 8612, 8621,	<code>\pagebreak</code> , 10409, 10421,	<code>\pauzadial</code> , 12366, 12373,
8708, 8713, 9256,	10425	13148
9779, 9964, 11783,	<code>\pagedepth</code> , 1935	<code>\pdef</code> , 935, 981, 1016, 1027,
11787, 11788, 11884,	<code>\pagefilllstretch</code> , 1934	1277, 1405, 1432,
11885, 12512, 12513,	<code>\pagefillstretch</code> , 1934	1454, 1512, 1709,
12911, 13125, 13129	<code>\pagefilstretch</code> , 1934	1754, 1786, 1789,
<code>\nX</code> , 108	<code>\pagegoal</code> , 1933, 4837,	1797, 1804, 1856,
<code>\nxxa</code> , 6459	4863, 4865, 4874,	1878, 1893, 1947,
<code>\oarg</code> , 9827	4876, 4885, 4896,	1959, 2002, 2015,
<code>\obeylines</code> , 2596, 2687,	4898, 8475	2044, 2055, 2080,
2909, 3133, 9915,	<code>\pagegoalortextheight</code> , 4889	2090, 2111, 2121,
12837, 13092	<code>\pageshrink</code> , 1935	2141, 2171, 2209,
<code>\obeyspaces</code> , 2592, 2685	<code>\pagestretch</code> , 1933	2218, 2287, 2304,
<code>\of</code> , 1804, 11759	<code>\pagestyle</code> , 13089	2323, 2324, 2345,
<code>\oldLaTeX</code> , 9995	<code>\pagetotal</code> , 1933, 4838	2360, 2368, 2412,
<code>\oldLaTeXe</code> , 9996	<code>\par</code> , 2954, 3834, 4296,	2631, 2691, 2696,
<code>\Omega</code> , 11320	6443, 7498, 9025,	2717, 2723, 2729,
<code>\omega</code> , 11352	11521, 11920, 12012,	2766, 2769, 2820,
<code>\on@line</code> , 8301, 8313, 8333,	12298, 12302, 12328,	2843, 2868, 2889,
8343, 8483, 9383,	12335, 12516, 12594,	2898, 2905, 2911,
10258, 12050	12631, 12672, 12696,	2996, 3015, 3033,
<code>\onecolumn</code> , 10407	12844, 12848, 13180	3078, 3102, 3112,
<code>\openout</code> , 11957, 13318, 13374	<code>\paragraph</code> , 12642	3122, 3134, 3163,
<code>\or</code> , 9564, 9565, 12546,	<code>\paragraphmark</code> , 10708	3193, 3210, 3224,
12547, 12548, 12570,	<code>\ParanoidPostsec</code> , 10572	3234, 3395, 3441,
		3455, 3459, 3476,

3483, 3509, 3540,  
 3552, 3554, 3583,  
 3588, 3617, 3765,  
 3771, 3783, 3792,  
 3883, 3900, 3949,  
 3951, 3953, 4154,  
 4161, 4174, 4251,  
 4336, 4478, 4671,  
 4673, 4716, 4728,  
 4743, 4746, 4750,  
 4780, 4859, 4870,  
 4881, 4982, 4995,  
 5092, 6403, 6410,  
 7489, 7565, 7842,  
 8049, 8189, 8387,  
 8404, 8599, 8653,  
 8667, 8681, 8836,  
 8945, 9051, 9111,  
 9230, 9486, 9488,  
 9618, 9639, 9665,  
 9673, 9688, 9695,  
 9709, 9754, 9765,  
 9770, 9821, 9827,  
 9829, 9830, 9834,  
 9839, 9918, 9938,  
 9942, 9968, 9974,  
 9976, 9998, 10028,  
 10332, 10799, 10853,  
 10863, 10874, 10902,  
 11206, 11388, 11501,  
 11505, 11576, 11597,  
 11608, 11625, 11628,  
 11631, 11650, 11659,  
 11663, 11771, 11866,  
 11903, 11968, 11970,  
 11972, 12063, 12112,  
 12121, 12194, 12360,  
 12373, 12380, 12391,  
 12404, 12434, 12534,  
 12560, 12588, 12619,  
 12628, 12636, 12649,  
 12652, 12676, 12699,  
 12884, 12941, 12986,  
 13003, 13016, 13020,  
 13037, 13042, 13096,  
 13435, 13456, 13485,  
 13490, 13495, 13672,  
 13674, 13707  
 \pdfTeX, 10127  
 \pdfLaTeX, 10130  
 \pdfstrcmp, 830  
 \pdfTeX, 10129  
 \pedef, 962  
 \penalty, 2269, 2272, 4149,  
 4744, 9003, 9005,  
 9034, 9040, 9044,  
 9683, 9723, 9776,  
 9779, 10545, 10556,  
 10558, 10560, 10563,  
 10566, 11569, 11570,  
 11576, 12161, 12355,  
 12362, 12375, 12406,  
 12421, 12428, 12603,  
 12872, 13674, 13675,  
 13713, 13715, 13720,  
 13722  
 \perCent, 144  
 \perCents, 144  
 \phantom, 12868  
 \Phi, 11318  
 \phi, 11349  
 \Pi, 11343  
 \pi, 11342  
 \pk, 9709  
 \PlainTeX, 10112  
 \pldef, 939, 4455, 4463,  
 4465, 4469, 4471, 4488  
 \pm, 11284  
 \pnew, 1034  
 \polskadata, 12534, 12557  
 \possfil, 9776  
 \possvfil, 9778, 12597  
 \postdisplaypenalty, 1982  
 Pp!L\long\par, p. 80  
 \ppauza, 12391  
 \ppauza@dash, 12356,  
 12397, 12443, 12461,  
 12469  
 \ppauza@skipcore, 12355,  
 12440, 12441  
 \pprovide, 997, 3622, 12057  
 \preamBeginningLeaf, 282  
 \preamEndingLeaf, 291  
 \prec, 11556  
 \predisplaypenalty, 1982  
 \predisplaysize, 1939  
 \prependtomacro, 2416,  
 7012, 7997, 11692, 12985  
 \prependtotoks, 2450  
 \pretolerance, 1979  
 \prevdepth, 1933, 3767, 3773  
 \prevgraf, 1977  
 \printspaces, 9679, 9688  
 \ProcessOptions, 10787  
 \ProcessOptionsX, 13783  
 \protect, 5854, 5857  
 \protected, 935, 938, 962,  
 964, 997, 1034, 1035,  
 2299, 2318, 3779,  
 5833, 7730, 7762,  
 9108, 9229, 9278,  
 9542, 9584, 9585,  
 11201, 11572, 11574,  
 11757  
 \protected@edef, 2346, 10394  
 \protected@empty, 4996  
 \protected@nil, 4995, 4996  
 \protected@xdef, 2346  
 \provide, 981, 997  
 \ProvideCommand, 8318  
 \providecommand, 9570, 9575  
 \providecounter, 13429,  
 13435  
 \providesStatement, 292  
 \Provides gmFileKind, 187  
 \Psi, 11319  
 \psi, 11351  
 \putpenalty, 9029  
 \pxdef, 964  
 \quad, 12374, 12519  
 \quantifierhook, 11510,  
 11533  
 \quiet, 7649, 7650  
 quiet, 5157, 10785  
 \quotkern, 13149, 13150  
 \raggedright, 12632  
 \raise, 10109, 10171,  
 10182, 10187, 12136  
 \rangle, 9636, 11282, 13597  
 \RCSAuthor, 13752  
 \RCSDate, 13750  
 \RCSFile, 13748  
 \RCSFileExt, 13747  
 \RCSFileName, 13746  
 \RCSTime, 13751  
 \RCSVersion, 13749  
 \rdate, 12628  
 \real, 12217, 12219  
 \reflectbox, 4775, 10139,  
 10152, 10178  
 \refstepcounter, 2346,  
 10331, 10393  
 \relax, 139, 140, 148, 150,  
 151, 270, 271, 272,  
 273, 274, 275, 276,  
 277, 278, 320, 321,  
 784, 953, 983, 1349,  
 1368, 2162, 2194,  
 2248, 2502, 2525,  
 2692, 2693, 2694,  
 2963, 3055, 3189,  
 3196, 3198, 3356,  
 3481, 3485, 3513,  
 3540, 3600, 3624,  
 3722, 3723, 3767,  
 3773, 3780, 3800,  
 3806, 3807, 3809,  
 3810, 3828, 3829,  
 3830, 3831, 3839,

3840, 3841, 3843,  
4085, 4088, 4091,  
4094, 4097, 4100,  
4105, 4148, 4176,  
4192, 4233, 4235,  
4241, 4269, 4304,  
4520, 4542, 4586,  
4692, 4697, 4698,  
4722, 4723, 4816,  
4817, 4818, 4839,  
4865, 4876, 4900,  
4905, 4918, 4955,  
5174, 6226, 6747,  
7195, 7440, 7448,  
7763, 7899, 8018,  
8087, 8401, 8402,  
8612, 8644, 8793,  
8808, 8810, 8840,  
8861, 8888, 8945,  
8968, 8970, 8980,  
8982, 9445, 9559,  
9682, 9683, 9781,  
9782, 9815, 10006,  
10008, 10012, 10020,  
10048, 10052, 10054,  
10136, 10138, 10139,  
10149, 10151, 10152,  
10175, 10394, 10409,  
10421, 10425, 10453,  
10456, 10464, 10545,  
10546, 10547, 10562,  
10564, 10576, 10577,  
10880, 10881, 11043,  
11069, 11089, 11090,  
11104, 11147, 11154,  
11170, 11178, 11203,  
11358, 11361, 11362,  
11363, 11364, 11365,  
11366, 11367, 11368,  
11369, 11414, 11477,  
11502, 11554, 11569,  
11570, 11576, 11600,  
11605, 11738, 11788,  
11789, 11884, 11885,  
11886, 11910, 11936,  
11942, 11957, 11998,  
12038, 12077, 12161,  
12169, 12170, 12171,  
12182, 12183, 12185,  
12215, 12221, 12222,  
12242, 12243, 12244,  
12293, 12303, 12324,  
12334, 12335, 12337,  
12346, 12353, 12355,  
12363, 12364, 12458,  
12468, 12489, 12512,  
12513, 12545, 12546,  
12570, 12576, 12659,  
12660, 12664, 12681,  
12692, 12724, 12738,  
12740, 12741, 12753,  
12755, 12756, 12788,  
12818, 12835, 12849,  
12874, 12888, 12901,  
12910, 12954, 13022,  
13069, 13119, 13125,  
13180, 13295, 13319,  
13354, 13375, 13533,  
13536, 13564, 13565,  
13566, 13675, 13698  
\relaxen, 1369, 4928, 4931,  
7677, 7678, 8949,  
10451, 12492, 12893,  
13487, 13492  
\relpenalty, 1980, 13575  
\relsize, p. 138, 9542,  
9543, 9580, 9581,  
9582, 9583, 9584, 9585  
\RenewCommand, 8307  
\renewcommand, 12047  
\RenewEnvironment, 8337  
\repeat, 8724  
\RequirePackage, 781, 815,  
817, 3790, 4984,  
5180, 5182, 5220,  
8364, 9101, 9310,  
9602, 9989, 10222,  
10310, 10778, 11951,  
12041, 12201, 13281,  
13448  
\reserved@a, 2733, 2742,  
2777, 2787, 2798,  
3178, 13312, 13313  
\reserved@b, 2734, 2744,  
2778, 2800, 3179,  
13311, 13313  
\reserved@c, 2739, 2742,  
2744, 2747, 2784,  
2787, 2798, 2800, 2804  
\reserved@d, 2732, 2741,  
2776, 2786, 2794, 3177  
\Reset@Macro, 3399, 3402,  
3431  
\Reset@Macros, 3427, 3429  
\Reset@MacroSt, 3400, 3414  
\ResetMacro, 3395  
\ResetMacros, 3427  
\resizebox, 3797, 11725  
\resizegraphics, 3792  
\Restore@Macro, 3334,  
3337, 3387, 3442,  
6405, 6414, 6416,  
6770, 8373, 8380,  
8383, 10000, 11155,  
11211  
\Restore@Macros, 3382, 3385  
\Restore@MacroSt, 3335,  
3362, 3461, 8380,  
10718, 13391  
\RestoreCatM, 3779, 3783  
\RestoreCommand, 8676,  
8677, 8737  
\RestoreEnvironment,  
3459, 8374, 8375, 8667  
\RestoreMacro, 3331  
\RestoreMacros, 3377  
\restoreparindent, 12761  
\RestoringDo, 3497  
\rho, 11344  
\rightharpoonright, 11287, 11553  
\righthyphenmin, 1989  
\rightline, 12628, 12794  
\rightrightarrowskip, 2036, 3830,  
3840, 12660, 12664,  
12679, 12714, 12752,  
12789, 12797, 12805,  
12861, 12862, 13040,  
13044, 13065  
\rlap, 12671, 12856  
\rm, 9666, 11398, 11403,  
11420, 11421, 11438,  
11461, 11462, 11529,  
11628, 11640  
\rmfamily, 10103, 10109,  
11014, 11056  
\rmopname, 11206, 11361,  
11362, 11363, 11364,  
11365, 11366, 11367,  
11368, 11369  
\romannumeral, 4105, 7148,  
7177, 7186, 7203,  
7204, 7205, 7207,  
8180, 8209, 11677,  
13241, 13259  
\romorzero, 11676, 11709,  
11714  
\rotatebox, 10171, 10182,  
10187, 11410, 11416,  
11424, 11427, 11557  
\rrthis, 13037  
\rs@size@warning, 9561,  
9566, 9570  
\rs@unknown@warning,  
9556, 9575  
\rule, 11537, 11544  
\runindate, 12641  
\SameAs, 5935, 6013, 8225,  
8296, 8308, 8319,  
8328, 8338, 8551, 8976



<code>\sbox</code> , 10040, 10083, 10089	<code>\show</code> , 3707, 3708, 3722, 3723, 4329, 7782, 8566, 13392	<code>\spifletter</code> , 2766, 9811, 9904, 9964, 12118, 12119, 12960
<code>\scalefsize</code> , 8883	<code>\showbox</code> , 3672	<code>\splitdiscards</code> , 8848
<code>\scantnoline</code> , 3759	<code>\showboxbreadth</code> , 1992, 3667, 3672, 3678, 8970, 8982	<code>\splitmaxdepth</code> , 1937
<code>\scantokens</code> , 1279, 3760, 11699, 11707, 13535	<code>\showboxdepth</code> , 1993, 3667, 3672, 3677, 8968, 8980	<code>\splittopskip</code> , 2037
<code>\scantwo</code> , 1277	<code>\ShowCommand</code> , 7772	<code>\sqrtsign</code> , 11357
<code>\scanverb</code> , 9883	<code>\ShowFont</code> , 12006	<code>\square</code> , 11304
<code>\Scope</code> , 5935, 6121, 6319, 7455, 8531, 8574, 8737, 8762	<code>\showlists</code> , 3667, 8971	<code>\stepcounter</code> , 12643
<code>\scriptsize</code> , 9552, 9564, 11413, 11421, 11462, 11543, 13024	<code>\ShowOut</code> , 4327	<code>\stepnummacro</code> , 4497, 6228
<code>\scriptspace</code> , 1938	<code>\showthe</code> , 3713, 3714	<code>\Store@Macro</code> , 3231, 3238, 3245, 3296, 6401, 6768, 8360, 9999, 10064, 10125, 11091, 11491, 12027, 12670, 13409, 13455
<code>\scshape</code> , 10110, 11546	<code>\Sigma</code> , 11316	<code>\Store@Macros</code> , 3291, 3294, 10706
<code>\sdName</code> , 4660	<code>\sigma</code> , 11345	<code>\Store@MacroSt</code> , 3232, 3238, 3253, 3457, 10065
<code>\secondclass</code> , 3736	<code>\sim</code> , 11285	<code>\StoreCatM</code> , 3778, 3784
<code>\SecondClasstrue</code> , 3738	<code>\sin</code> , 11364	<code>\StoreCommand</code> , 8599
<code>\secondoftwo</code> , 197	<code>\skewchar</code> , 10992, 11025, 11038, 11050, 11062	<code>\Stored@Macro</code> , 3441, 3442
<code>\sectionmark</code> , 10707	<code>\skip</code> , 2031, 10546, 10547, 10549, 10550, 10553, 10557, 10559, 10576, 10577	<code>\storedcsname</code> , 3445, 10717, 11033, 11045, 12028, 13456
<code>\selectfont</code> , 8880, 8890, 10037, 10080, 10903	<code>\skiplines</code> , 139	<code>\StoredMacro</code> , 3441
<code>\set@smashbox</code> , 8761	<code>\SliTeX</code> , 10109	<code>\StoreEnvironment</code> , 3455, 8361, 8362, 8653
<code>\setbasefont</code> , 11868	<code>\small</code> , 9547, 9564, 12023	<code>\StoreMacro</code> , 3224
<code>\setbox</code> , 8764, 8805, 8840, 8843, 8848, 8850, 8857, 10035, 10073, 10078, 10162, 10170, 10181, 10186, 10191, 10522, 11413, 11439, 11443, 11536, 11547, 11548, 12749	<code>\smaller</code> , p. 138, 9581, 12112, 12144, 12987	<code>\StoreMacro@nocat</code> , 3234, 3357, 6408, 6931
<code>\setcounter</code> , 10329, 13432	<code>\smallerr</code> , p. 138, 9585, 12231	<code>\StoreMacros</code> , 3288
<code>\setlength</code> , 12212, 12213, 12217, 12219	<code>\smallskipamount</code> , 11929, 11930, 11931	<code>\StoringAndRelaxingDo</code> , 3476
<code>\SetMathAlphabet</code> , 11029, 11066	<code>\smartunder</code> , 2631	<code>\strcmp</code> , 114, 829, 830, 1151, 1189, 1204, 1233, 1393, 1425, 3912
<code>\setprevdepth</code> , 3767, 3773, 12799	<code>\smash</code> , 8767, 10166	<code>\StreamPut</code> , 243
<code>\SetSectionFormatting</code> , 10451, 10452, 10615, 10620, 10629, 10637, 10644, 10651, 10656	<code>\smash@box</code> , 8767	<code>\string</code> , 868, 875, 877, 1193, 1834, 1839, 2250, 2299, 2301, 2318, 2319, 3149, 3654, 3784, 3799, 4341, 7321, 7498, 7559, 7623, 7629, 7669, 7738, 7774, 7776, 7779, 8042, 8043, 9561, 9566, 9576, 9811, 10020, 10269, 11472, 11709, 13148, 13149, 13150, 13297, 13306, 13356, 13363, 13432, 13495, 13633, 13640, 13697
<code>\setspacekip</code> , 8394	<code>\SMglobal</code> , 3210	<code>\strip@bslash</code> , 862, 891, 929, 1233, 3572, 6066, 6281, 6282,
<code>\SetSymbolFont</code> , 11017, 11058	<code>\sName</code> , 4657	
<code>\settowidth</code> , 3851	<code>\space</code> , 143, 144, 2766, 2957, 4059, 4070, 4460, 4920, 6246, 7586, 7587, 7622, 7629, 9640, 9925, 9946, 10117, 10363, 11025, 11038, 11050, 11062, 11087, 11088, 11102, 11103, 11170, 11177, 11202, 11203, 11358, 11942, 12078, 12099, 12551, 12575, 13096, 13099, 13297, 13356, 13495	
<code>\SetTwoheadSkip</code> , 10599, 10628, 10636, 10643	<code>\spacefactor</code> , 1977	
<code>\sf@size</code> , 10036, 10079, 10905	<code>\spaces@and@Ms</code> , 2603	
<code>\sffamily</code> , 11942	<code>\spaceskip</code> , 2037, 8401, 8405	
<code>\sfname</code> , 9688, 9698	<code>\sphack</code> , 7654, 7656, 7769	
<code>\shortleftarrow</code> , 11552		
<code>\shortpauza</code> , 12457, 12467		
<code>\shortrightarrow</code> , 11554		
<code>\shortthousep</code> , 12945		

6299, 6302, 6534,	6232, 6246, 6351,	<code>\tolerance</code> , 1979, 12323,
6536, 6537, 6538,	6352, 6398, 6983,	12334, 12662, 12693,
6875, 6884, 7370,	6995, 7016, 7299,	12725
7374, 7548, 8191,	7306, 7714, 7723,	<code>\topsep</code> , 4292, 13243, 13261
8202, 8524	7735, 7853, 7872,	<code>\topskip</code> , 2036
<code>\strip@pt</code> , 4814, 11598,	7873, 7882, 7957,	<code>\totalheight</code> , 8792, 8807
11613, 11619, 11639,	7964, 7968, 8256,	<code>\tracingassigns</code> , 3688, 3697
11797, 11882	8588, 8589, 8594,	<code>\tracingcommands</code> , 1988,
<code>\strut</code> , 12821	8612, 8621, 8708,	3689, 3698
<code>\subs</code> , 2558, 2636	8713, 8825, 8878,	<code>\tracinglostchars</code> , 1987
<code>\subsectionmark</code> , 10707	8879, 8888, 8889,	<code>\tracingmacros</code> , 1986,
<code>\subsubsectionmark</code> , 10708	9350, 9370, 9498,	3687, 3696
<code>\sum</code> , 11409	9503, 9742, 9964,	<code>\tracingonline</code> , 1985,
<code>\supposedJobname</code> , 103, 114	10054, 10593, 10594,	3666, 3671, 3676,
<code>\sups</code> , 2562	10957, 10992, 11399,	3686, 3695, 8967,
<code>\symgmathroman</code> , 11358	11404, 11440, 11441,	8978, 9281
	11723, 11758, 11772,	<code>\tracingoutput</code> , 1987,
<code>\tabskip</code> , 2037	11783, 11787, 11793,	3675, 3699, 8979
<code>\tan</code> , 11368	11823, 11825, 11858,	<code>\tracingpages</code> , 1987
<code>\tau</code> , 11347	11866, 11877, 12012,	<code>\tracingparagraphs</code> , 1986
<code>\TB</code> , 10118	12171, 12225, 12228,	<code>\tracingrestores</code> , 1988
<code>\TeX</code> , 9998, 9999, 10000,	12512, 12514, 12545,	<code>\tracingstats</code> , 1986
10057, 10091, 10101,	12576, 12738, 12750,	<code>\tri@let</code> , 3554, 8568
10106, 10110, 10112,	12752, 12941, 13241,	<code>\truetextsuperscript</code> ,
10117, 10122, 10129,	13259, 13432, 13501,	10917, 10919
10140, 10195, 10198,	13529, 13558, 13559,	<code>\TrzaskaTilde</code> , 13061
10201	13560	<code>\ttfamily</code> , 9762, 12658
<code>\TeXbook</code> , 10117, 10118	<code>\thedate</code> , 12646	<code>\tw@</code> , 6210, 9850, 10040,
<code>\text</code> , 13218	<code>\Theta</code> , 11313	10041, 10050, 10051,
<code>\textbullet</code> , 3617, 3627	<code>\theta</code> , 11334	10052, 10053, 10054,
<code>\textcolor</code> , 12030	<code>\thickmuskip</code> , 11477,	10083, 10084, 10547,
<code>\textheight</code> , 4837, 4897	13560, 13566, 13574	10549, 10553, 10559,
<code>\textlarger</code> , 9582	<code>\thinmuskip</code> , 13558, 13564,	10587, 10594, 12693
<code>\textlit</code> , 11625	13574	<code>\two@Ms</code> , 2597, 2600, 2603,
<code>\textsc</code> , 10104, 10112,	<code>\thispagestyle</code> , 10383,	13180
10114, 10198, 12519	10386, 10387, 10412,	<code>\twocoltoc</code> , 11950, 11963
<code>\textsf</code> , 9688, 9701, 9709	12516, 13345	<code>\twocolumn</code> , 10415
<code>\textsl</code> , 10117	<code>\thous</code> , 12946	<code>\twopar</code> , 12716
<code>\textsmaller</code> , 9583	<code>\thousep</code> , 12884, 12941, 12964	<code>\twopar@atleast</code> , 12715,
<code>\textstyle</code> , 10062	<code>\thr@@</code> , 5025, 6210, 13237,	12720, 12734
<code>\textsuperscript</code> , 10907,	13255	<code>\twopar@default</code> , 12745
10911, 10920, 12987	<code>\time</code> , 1992, 12512, 12513	<code>\twopar@defts</code> , 12713, 12738
<code>\textsuperscript@@</code> ,	<code>\tiny</code> , 9553, 9561, 9564,	<code>\twoparcore</code> , 12727, 12733,
10902, 10907, 10912,	11421, 11462, 11543,	12744
10920	12199	<code>\twoparinit</code> , 12711
<code>\texttilde</code> , 11903	<code>\tinycae</code> , 12199	<code>\type@bslash</code> , 9723, 9751
<code>\textvisiblespace</code> , 2679	<code>\TO@resa</code> , 4323, 4324, 4325,	<code>\type@lbrace</code> , 9740, 9821
<code>\textwidth</code> , 12817	4329	<code>\TypeOut</code> , 4107, 4322, 4328
<code>\tg</code> , 11366	<code>\TODO</code> , 11941	<code>\typeout</code> , 3721, 3723, 4325,
<code>\the</code> , 1799, 2423, 2441,	<code>\toks</code> , 2070, 10586, 10587,	7203, 7205, 8483,
3189, 3767, 3773,	10593, 10594	8563, 11223, 12050
3780, 3850, 3856,	<code>\toks@</code> , 5834, 5874, 6064,	<code>\tytul</code> , 12652
4085, 4088, 4091,	6147, 6158, 6351,	<code>tytulowa</code> , <u>12516</u>
4094, 4097, 4100,	6352, 6373, 7735,	<code>\U</code> , 7918
4460, 4500, 4584,	7871, 7872	<code>\u</code> , 7918
4851, 4911, 5834,		<code>\ubound</code> , 7931, 7984
5859, 6218, 6219,		

<code>\uccode</code> , 11997, 12099	<code>\UrlFix</code> , 13485, 13487, 13490, 13492, 13495	<code>\vsplit</code> , 8843
<code>\uchyph</code> , 1988	<code>\UrlFont</code> , 13519, 13695	<code>\vss</code> , 10041, 10085
<code>\udigits</code> , 10799, 10802	<code>\UrlHyphenchar</code> , 13533, 13540	<code>\W</code> , 7918
<code>\UnDeclareCommand</code> , 8049	<code>\UrlLeft</code> , 13528	<code>\w</code> , 7918
<code>\undeksmallskip</code> , 11930	<code>\Urlmuskip</code> , 13574	<code>\wd</code> , 4233, 4235, 8792, 10039, 10042, 10077, 10082, 10090, 10091, 10165, 10192, 11414, 11536, 11537, 11549, 11550, 12750
<code>\unexpanded</code> , 847, 916, 1151, 1472, 1882, 1897, 2397, 2418, 2719, 2725, 3804, 3807, 3809, 3810, 3813, 3937, 4060, 4071, 4270, 4451, 4466, 4469, 4483, 5842, 6540, 6803, 6830, 6850, 6922, 6940, 7578, 7754, 7871, 7885, 7891, 8085, 8116, 8275, 9152, 9153, 9183, 9184, 9186, 9189, 9191, 9203, 9277, 9738, 10492, 11146, 12912, 13589, 13594, 13606, 13607	<code>\UrlNoBreaks</code> , 13512, 13580 <code>\UrlRight</code> , 13537 <code>\UrlSlashKern</code> , 13693 <code>\UrlSpecials</code> , 13513, 13582 <code>\urlstyle</code> , 13687, 13701 <code>\usc</code> , 12121, 12123 <code>\uscacro</code> , 12123 <code>\usecounter</code> , 13248 <code>\usepreamble</code> , 329 <code>\UseStored</code> , 8681, 8738	<code>\Web</code> , 10114 <code>\wedge</code> , 11308, 11427, 11475 <code>\whenonly</code> , 13339 <code>\whern</code> , 12826 <code>\whern@parbox</code> , 12796, 12804, 12861, 12862 <code>\wherncore</code> , 12779, 12828, 12829 <code>\whernfont</code> , 12787, 12795, 12821, 12832 <code>\whernskip</code> , 12829, 12834, 12835 <code>\whernup</code> , 12838 <code>\widetilde</code> , 11896 <code>\widowpenalty</code> , 1981, 9780 <code>\WPheadings</code> , 10614 <code>\write</code> , 4460, 4492, 9280, 9281, 13306, 13319, 13363, 13375, 13428 <code>\writefrom</code> , 210 <code>\WritePreamble</code> , 212 <code>\writeto</code> , 208, 334 <code>\Ws</code> , 11574 <code>\wyzejnizej</code> , 12239 <code>\Wz</code> , 11576
<code>\unless</code> , 114, 804, 809, 829, 1017, 1087, 2256, 2257, 3343, 3364, 3600, 3745, 4188, 4910, 5222, 5737, 8521, 8720, 8722, 9199, 10780, 11423, 11426, 11430, 12048, 12993, 13436	<code>\varepsilon</code> , 10062, 10122, 11331 <code>\varnothing</code> , 11271, 11547 <code>\varsigma</code> , 11346 <code>\vartheta</code> , 11335 <code>\vbadness</code> , 1979, 4910, 4911, 4912, 4918, 4920 <code>\vbadness@M</code> , 4909, 8842 <code>\vbadness@Restore</code> , 4916, 8844 <code>\vbox</code> , 8422, 8848, 8850, 10041, 10084, 12857 <code>\vee</code> , 11309, 11424, 11475 <code>\verb</code> , 8360, 8373 <code>\verbatim@mathhack</code> , 13523, 13550 <code>\verbatim@mathhack@</code> , 13552, 13556 <code>\verbatim@specials</code> , 9735, 13520, 13522 <code>\verbatim@specials@list</code> , 13551, 13552 <code>\vfil</code> , 9778 <code>\vfilneg</code> , 9783 <code>\vfuzz</code> , 1936 <code>\visiblespace</code> , 2677, 2681, 9673, 13514 <code>\voffset</code> , 1940 <code>\voidb@x</code> , 8857 <code>\vrule</code> , 4789 <code>\vrule@zero</code> , 4789 <code>\Vs</code> , 11572 <code>\vs</code> , 9673, 9679, 9683 <code>\vsize</code> , 1936 <code>\vskip</code> , 9052, 10550, 10553, 10557, 10559, 10561, 11929, 11930, 11931, 11932, 11933, 12604, 12829, 12849, 12867, 12876	<code>\whern@parbox</code> , 12796, 12804, 12861, 12862 <code>\wherncore</code> , 12779, 12828, 12829 <code>\whernfont</code> , 12787, 12795, 12821, 12832 <code>\whernskip</code> , 12829, 12834, 12835 <code>\whernup</code> , 12838 <code>\widetilde</code> , 11896 <code>\widowpenalty</code> , 1981, 9780 <code>\WPheadings</code> , 10614 <code>\write</code> , 4460, 4492, 9280, 9281, 13306, 13319, 13363, 13375, 13428 <code>\writefrom</code> , 210 <code>\WritePreamble</code> , 212 <code>\writeto</code> , 208, 334 <code>\Ws</code> , 11574 <code>\wyzejnizej</code> , 12239 <code>\Wz</code> , 11576  <code>\xA</code> , 107, 204, 213, 219, 223, 233, 251, 256, 261 <code>\xAE</code> , 6463 <code>\xAEacher</code> , 6463 <code>\xAeachr</code> , 6463 <code>\xanx</code> , 6455 <code>\xathousep</code> , 12941, 12958 <code>\xaxa</code> , 6455 <code>\xdef</code> , 964, 2725, 3767, 5104, 9111, 9213, 9230, 9348, 10239, 10515, 10953, 10977, 10978, 10992, 11638, 11858, 11881, 12225, 12750 <code>\xdefU</code> , 2723 <code>\Xedekfrac</code> , 10808 <code>\XeLaTeX</code> , 10149 <code>\XeTeX</code> , 10136 <code>\XeTeXcharglyph</code> , 10174
<code>\unsetspacekip</code> , 8404	<code>\vbadness@M</code> , 4909, 8842	<code>\whern@parbox</code> , 12796, 12804, 12861, 12862
<code>\unskip</code> , 2368, 9925, 9946, 12355, 12362, 12374, 12396, 12406, 12442, 12821, 13096, 13099	<code>\vbadness@Restore</code> , 4916, 8844	<code>\wherncore</code> , 12779, 12828, 12829
<code>\unvbox</code> , 8851, 8852	<code>\vbox</code> , 8422, 8848, 8850, 10041, 10084, 12857	<code>\whernfont</code> , 12787, 12795, 12821, 12832
<code>\uparrow</code> , 11291	<code>\vee</code> , 11309, 11424, 11475	<code>\whernskip</code> , 12829, 12834, 12835
<code>\uppercase</code> , 6298, 11998	<code>\verb</code> , 8360, 8373	<code>\whernup</code> , 12838
<code>\upshape</code> , 11033, 11056, 11654	<code>\verbatim@mathhack</code> , 13523, 13550	<code>\widetilde</code> , 11896
<code>\Upsilon</code> , 11317	<code>\verbatim@mathhack@</code> , 13552, 13556	<code>\widowpenalty</code> , 1981, 9780
<code>\upsilon</code> , 11348	<code>\verbatim@specials</code> , 9735, 13520, 13522	<code>\WPheadings</code> , 10614
<code>\Url</code> , 11482, 11483, 13707	<code>\verbatim@specials@list</code> , 13551, 13552	<code>\write</code> , 4460, 4492, 9280, 9281, 13306, 13319, 13363, 13375, 13428
<code>\url</code> , 13455, 13456	<code>\vfil</code> , 9778	<code>\writefrom</code> , 210
<code>\Url@Format</code> , 13518, 13684	<code>\vfilneg</code> , 9783	<code>\WritePreamble</code> , 212
<code>\Url@HyperHook</code> , 13681	<code>\vfuzz</code> , 1936	<code>\writeto</code> , 208, 334
<code>\Url@moving</code> , 13689	<code>\visiblespace</code> , 2677, 2681, 9673, 13514	<code>\Ws</code> , 11574
<code>\Url@z</code> , 13679	<code>\voffset</code> , 1940	<code>\wyzejnizej</code> , 12239
<code>\urladdstar</code> , 13452, 13459	<code>\voidb@x</code> , 8857	<code>\Wz</code> , 11576
<code>\UrlBigBreakPenalty</code> , 13575, 13578	<code>\vrule</code> , 4789	<code>\xA</code> , 107, 204, 213, 219, 223, 233, 251, 256, 261
<code>\UrlBigBreaks</code> , 13511, 13578	<code>\vrule@zero</code> , 4789	<code>\xAE</code> , 6463
<code>\UrlBreakPenalty</code> , 13575, 13576, 13581	<code>\Vs</code> , 11572	<code>\xAEacher</code> , 6463
<code>\UrlBreaks</code> , 13508, 13576	<code>\vs</code> , 9673, 9679, 9683	<code>\xAeachr</code> , 6463
	<code>\vsize</code> , 1936	<code>\xanx</code> , 6455
	<code>\vskip</code> , 9052, 10550, 10553, 10557, 10559, 10561, 11929, 11930, 11931, 11932, 11933, 12604, 12829, 12849, 12867, 12876	<code>\xathousep</code> , 12941, 12958
		<code>\xaxa</code> , 6455
		<code>\xdef</code> , 964, 2725, 3767, 5104, 9111, 9213, 9230, 9348, 10239, 10515, 10953, 10977, 10978, 10992, 11638, 11858, 11881, 12225, 12750
		<code>\xdefU</code> , 2723
		<code>\Xedekfrac</code> , 10808
		<code>\XeLaTeX</code> , 10149
		<code>\XeTeX</code> , 10136
		<code>\XeTeXcharglyph</code> , 10174

<code>\XeTeXdelcode</code> , 11172, 11176	<code>\xiitab</code> , 5139, 5145	10546, 10549, 10550,
<code>\XeTeXdelimiter</code> , 11202	<code>\xiiunder</code> , 2565, 2567,	10557, 10564, 10576,
<code>\XeTeXfonttype</code> , 10173	2634, 2637	10577, 10586, 10593,
<code>\XeTeXglyph</code> , 3624	<code>\xleq</code> , 11244, 11247, 11249	10616, 10659, 10905,
<code>\XeTeXglyphbounds</code> , 5038,	<code>\xspaceskip</code> , 2037	11207, 11651, 11653,
5066	<code>\xxt@visibleospace</code> , 2676,	11795, 11993, 11995,
<code>\XeTeXglyphindex</code> , 3624,	2677	11996, 11997, 12000,
5038, 5066	<code>\xxt@visibleospace@fallback</code> ,	12008, 12009, 12010,
<code>\XeTeXinputencoding</code> , 796	2678	12011, 12012, 12661,
<code>\XeTeXmathchardef</code> , 11093		12664, 12665, 12894,
<code>\XeTeXmathcode</code> , 11096, 11101	<code>\year</code> , 1992	13066, 13243, 13261,
<code>\XeTeXpar</code> , 10161	<code>\yeshy</code> , 4149, 9740	13332, 13383, 13531,
<code>\XeTeXradical</code> , 11358		13674, 13675
<code>\XeTeXthree</code> , 8352	<code>\z@</code> , 1155, 1191, 1206, 1233,	<code>\z@skip</code> , 4149, 4744, 8405,
<code>\XeTeXuseglyphmetrics</code> ,	1393, 1425, 1830,	9723, 12161, 12421,
10180	3827, 3838, 3912,	12633, 12701, 12874,
<code>\XeTeXversion</code> , 784, 785,	4586, 4671, 4722,	13046, 13713, 13716,
795, 2605, 3599,	4723, 4743, 4752,	13720, 13723
3600, 12401, 12475,	4788, 4789, 4989,	<code>\zeta</code> , 11332
12732, 13134	5129, 5870, 5871,	<code>\zf@default@options</code> , 8388
<code>\xgeq</code> , 11245, 11252, 11254	5889, 5895, 5914,	<code>\zf@scale</code> , 12207, 12210,
<code>\Xi</code> , 11315	5916, 6224, 6751,	12211
<code>\xi</code> , 11341	6755, 6900, 8481,	<code>\zwrobcy</code> , 12649
<code>\xiiand</code> , 2660	8814, 8849, 8927,	
<code>\xiibackslash</code> , 2647, 2651	9281, 9561, 10035,	<code>\cdot</code> , 11972
<code>\xiiEdetoke</code> , 3163, 4155	10036, 10039, 10041,	
<code>\xiihash</code> , 1894, 1898, 2666	10042, 10045, 10047,	<code>\U</code> , 7918
<code>\xiilbrace</code> , 2616, 9894	10048, 10049, 10051,	<code>\u</code> , 7918
<code>\xiiM</code> , 2577	10073, 10076, 10077,	
<code>\xiipercent</code> , 2656	10078, 10079, 10082,	<code>\_</code> , 12434, 12486, 12487,
<code>\xiirbrace</code> , 2617	10084, 10089, 10090,	12488, 12489
<code>\xiispace</code> , 2663, 2681,	10091, 10162, 10164,	<code>\_</code> , 12404, 12448, 12485,
3146, 3147, 3159	10165, 10191, 10192,	12487, 12488, 12489
<code>\xiistring</code> , 3145	10193, 10388, 10522,	<code>\</code> , 151