

Grzegorz Murzynowski

The gmutils Package*

Written by Grzegorz Murzynowski,
natror at o2 dot pl

© 2005, 2006, 2007, 2008, 2009, 2010 by Grzegorz Murzynowski.

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

See <http://www.ctan.org/tex-->

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

LPPL status: "author-maintained".

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

```
106 \NeedsTeXFormat {LaTeX2e}
107 \ProvidesPackage {gmutils}
108      [2010/03/04 v0.991 some rather TeXnical macros, some
      of them tricky (GM) ]
```

Contents

Intro	2	Storing and restoring the meanings of CSes	14
Installation	2	\DeclareCommand and \DeclareEnvironment	18
Contents of the gmutils.zip archive	2	Ampulex Compressa-like modifications of macros	36
Compiling of the documentation	2	Environments redefined	38
Options	3	Almost an environment or redefinition of <code>\begin</code>	38
A couple of abbreviations	3	<code>\@ifenvir</code> and improvement of <code>\end</code>	39
\ifs as a four-argument L^AT_EX command robust to unbalanced \ifs and \fis	4	Storing and restoring the catcodes of specials	40
\firstofone and the queer \catcodes	4	From relsize	41
\afterfi and pals	5	Meta-symbols	42
\foone	5	Macros for printing macros and filenames	43
\@ifempty, \IfAmong, \IfIntersect \@ifinmeaning	5	Typesetting arguments and commands	46
Global Boolean switches	6	Not only preamble!	48
\gm@ifundefined—a test that doesn't create any hash entry unlike \@ifundefined	9	Third person pronouns	49
Some 'other' and active stuff	9		
\@ifnextcat, \@ifnextac, catcode-independent \gm@ifstar, \@ifnextsingle, \@ifnextgroup	11		

* This file has version number v0.991 dated 2010/03/04.

Improvements to mwcls sectioning			
commands	50	Switching on and off parts of one file	87
An improvement of MW's		Fix of including when fontspec is used	88
<code>\SetSectionFormatting</code>	52	Faked small caps	88
Negative <code>\addvspace</code>	53	See above/see below	89
My heading setup for mwcls	55	<code>luzniej</code> and <code>napa-</code>	
Compatibilising standard and mwcls		<code>pierki</code> —environments	
sectionings	56	used in page breaking for money	90
enumerate* and itemize*	57	Typesetting dates in my memoirs	93
The logos	58	For dati under poems	97
Expandable turning stuff all into 'other'	60	Thousand separator	98
Brave New World of X_YTeX	61	<code>hyperref's \nolinkurl</code> into <code>\url*</code>	100
Fractions	61	Footnotes suggested by Andrzej	
<code>\resizegraphics</code>	63	Tomaszewski	100
Settings for mathematics in main font	64	A fix to the url package	101
Minion and Garamond Premier		Conditional tilde	105
kerning and ligature fixes	75	Storing the catcode of line end	106
A left-slanted font	75	A really empty page	106
Fake Old-style Numbers	77	Change History	107
Varia	81	Index	111
<code>\include not only .tex's</code>	86		

Intro

The `gmutils.sty` package provides some macros that are analogous to the standard L^AT_EX ones but extend their functionality, such as `\@ifnextcat`, `\addtomacro` or `\begin(*)`. The others are just conveniences I like to use in all my TeX works, such as `\afterfi`, `\pk` or `\cs`.

I wouldn't say they are only for the package writers but I assume some nonzero (L^A)T_EX-awareness of the user.

For details just read the code part.

Installation

Unpack the `gmutils.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_EX 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.sty
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 `.sty` file by running L^AT_EX on the `gmutils.sty` file twice (`xelatex gmutils.sty` in the

directory you wish the documentation to be in, you don't have copy the .sty file there, T_EX will find it), then MakeIndex on the gmutils.idx file, and then L^AT_EX on gmutils.sty once more.

MakeIndex shell command:

```
makeindex -r gmutilsDoc
```

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 gm-docc.cls), gmverb.sty, gmutils.sty, gmiflink.sty and also some standard packages: hyper-ref.sty, color.sty, geometry.sty, multicol.sty, lmodern.sty, fontenc.sty that should be installed on your computer by default.

If you had not installed the mwcls classes (available on CTAN and present in T_EX 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.

```
180 \ifx\XeTeXversion\relax
181   \let\XeTeXversion\@undefined% If someone earlier used
      % \@ifundefined{XeTeXversion} to test whether the engine is XYTEX,
      then \XeTeXversion is defined in the sense of  $\epsilon$ -TEX tests. In that case
      we \let it to something really undefined. Well, we might keep sticking
      to \@ifundefined, but it's a macro and it eats its arguments, freezing their
      catcodes, which is not what we want in line 6155.
188 \fi
190 \ifdefined\XeTeXversion
191 \XeTeXinputencoding utf-8% we use Unicode dashes later in this file.
192 \fi% and if we are not in XYTEX, we skip them thanks to XYTEX-test.
```

Options

```
\ifgmu@quiet 213 \newif\ifgmu@quiet
quiet         215 \DeclareOption{quiet}{\gmu@quiettrue}
217 \ProcessOptions
```

A couple of abbreviations

```
\@xa 222 \let\@xa\expandafter
\@nx 223 \let\@nx\noexpand
```

```
\@xau 225 \def\@xau#1{\unexpanded\@xa{#1}}
```

Note that there's only one `\expandafter`, after `\unexpanded`. It's because `\unexpanded` expands expandable tokens and gobbles `\relaxes` while looking for an opening brace or `\bgroup`. Note also that it *requires* a text in braces.

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

```
\pdef 240 \def\pdef{\protected\def}
```

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

```
\gobble 247 \long\def\gobble#1{}
```

```

\@gobble 249 \let \@gobble \gobble
\gobbletwo 250 \let \gobbletwo \@gobbletwo

\provide 254 \long \pdef \provide#1 {%
255   \ifdefined#1%
256     \ifx \relax#1 \afterfifi {\def#1}%
257     \else \afterfifi {\gmu@gobdef}%
258     \fi
259   \else \afterfi {\def#1}%
260   \fi}

\gmu@gobdef 263 \long \def \gmu@gobdef#1# {%
264   \def \gmu@tempa {}% it's a junk macro assignment to absorb possible prefixes.
265   \@gobble}

```

```

\pprovide 269 \def \pprovide {\protected \provide}

```

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

Note moreover that unlike L^AT_EX's `\providecommand`, our `\(p)provide` allow any parameters string just like `\def` (because they just *expand* to `\def`).

```

\@nameedef 282 \long \def \@nameedef#1#2 {%
283   \@xa \edef \csname#1 \endcsname {#2}}

```

`\ifs` as a four-argument L^AT_EX command robust to unbalanced `\ifs` and `\fis`

```

\gmu@if 287 \long \def \gmu@if#1#2 {%
288   \csname _if#1 \endcsname#2 \@xa \@firstoftwo \else \@xa%
289     \@secondoftwo \fi

```

`\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 `\cat` codes locally for a definition *outside* the changed `\catcodes`' group. Just see the below usage of this macro 'with T_EX's eyes', as my T_EX Guru taught me.

```

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

\scantwo 304 \long \pdef \scantwo#1#2 {
305   \begingroup \endlinechar \m@ne
306   \@xa \endgroup \scantokens {#1#2}%
307 }

\@firstofthree 309 \long \def \@firstofthree#1#2#3 {#1}
\@secondofthree 310 \long \def \@secondofthree#1#2#3 {#2}
\@thirdofthree 311 \long \def \@thirdofthree#1#2#3 {#3}
\@secondoffive 312 \long \def \@secondoffive#1#2#3#4#5 {#2}

```

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

```

\@firstofmany 319 \long\def\@firstofmany#1#2\@nil{#1}
\@allbutfirstof 322 \long\def\@allbutfirstof#1#2\@nil{#2}

```

\afterfi and pals

It happens from time to time that you have some sequence of macros in an `\if...` and you would like to expand `\fi` before expanding them (e.g., when the macros should take some tokens next to `\fi...` as their arguments. If you know how many macros are there, you may type a couple of `\expandafters` and not to care how terrible it looks. But if you don't know how many tokens will there be, you seem to be in a real trouble. There's the Knuthian trick with `\next`. And here another, revealed to me by my T_EX Guru.

I think the situations when the Knuthian (the former) trick is not available are rather seldom, but they are imaginable at least: the `\next` trick involves an assignment so it won't work e.g. in `\edef`.

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

```

\longafterfi 353 \def\longafterfi{%
  \afterfi 354 \long\def\afterfi##1##2\fi{\fi##1}
\longafterfi 355 \longafterfi

```

And two more of that family:

```

\afterfifi 357 \long\def\afterfifi#1#2\fi#3\fi{\fi\fi#1}
\afteriffifi 359 \long\def\afteriffifi#1#2\fi#3\fi{\fi#1}

```

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

```

\afteriffiffifi 363 \long\def\afteriffiffiffifi#1#2\fi#3\fi#4\fi{\fi#1}
\afteriffiffifi 364 \long\def\afteriffiffiffifi#1#2\fi#3\fi#4\fi{\fi\fi#1}
\afteriffifi 365 \long\def\afteriffiffifi#1#2\fi#3\fi#4\fi{\fi\fi\fi#1}

```

\foone

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

```

\foone 372 \long\def\foone#1{\begingroup#1\relax\egfirstofone}
374 \long\def\egfirstofone#1{\endgroup#1}
\foeatletter 376 \def\foeatletter{\foone\makeatletter}

```

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

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

```

\@ifempty 386 \long\pdef\@ifempty#1{%
  % #1 the token(s) we test, it may be just {},
  % #2 the stuff executed if #1 is empty (is {}),

```

```

% #3 the stuff executed if #1 consists of at least one token.
\gmu@reserveda 392 \def\gmu@reserveda{#1}%
393 \ifx\gmu@reserveda\@empty\@xa\@firstoftwo
394 \else\@xa\@secondoftwo
395 \fi}

\@ifnonempty 397 \long\def\@ifnonempty#1#2#3{\@ifempty{#1}{#3}{#2}}
\IfAmong 399 \long\pdef\IfAmong#1\among#2{%
% #1 the token(s) whose presence we check,
% #2 the list of tokens in which we search #1,
% #3 the 'if found' stuff,
% #4 the 'if not found' stuff.
\gmu@among@ 408 \long\def\gmu@among@##1#1##2\gmu@among@{%
409 \@ifempty{##2}\@secondoftwo\@firstoftwo}%
410 \gmu@among@#2#1\gmu@among@}

\IfIntersect 412 \long\def\IfIntersect#1#2{% this (not expandable) macro checks whether
the list of tokens #1 and #2 have nonempty intersection and if so executes
% \@firstoftwo or else \@secondoftwo, so it's a 4-argument com-
mand:
% #1 first list to match,
% #2 second list to match,
% #3 if match (nonempty intersection),
% #4 if not match (empty intersection).
422 \let\IfIntersect@next\@secondoftwo
423 \gmu@foreach#1\gmu@foreach{%
424 \@xa\IfAmong\gmu@forarg\among{#2}
425 {\let\IfIntersect@next\@firstoftwo}}}%
426 \IfIntersect@next}

We need a macro that iterates over every token on a list. LATEX's \@for iterates over a list
separated with commas so it's not the case. Our macro is much simpler.

\gmu@foreach 431 \long\def\gmu@foreach#1\gmu@foreach#2{%
\gmu@forer 432 \long\def\gmu@forer##1{%
433 \ifx\gmu@foreach##1\else
\gmu@forarg 434 \long\def\gmu@forarg{##1}%
435 #2%
436 \@xa\gmu@forer
437 \fi}%
438 \gmu@forer#1\gmu@foreach}

\@ifinmeaning 442 \long\pdef\@ifinmeaning#1\of#2{%
% #1 the token(s) whose presence we check,
% #2 the macro in whose meaning we search #1 (the first token of this
argument is expanded one level with \expandafter),
% #3 the 'if found' stuff,
% #4 the 'if not found' stuff.
\gmu@reserveda 458 \def\gmu@reserveda{\IfAmong#1\among}%
459 \@xa\gmu@reserveda\@xa{#2}}

```

Global Boolean switches

The `\newgiff` declaration's effect is used even in the L^AT_EX 2_ε source by redefining some particular user defined ifs (UD-ifs henceforth) step by step. The goal is to make the

UD-if's assignment global. I needed it at least twice during gmdoc writing so I make it a macro. It's an almost verbatim copy of L^AT_EX's `\newif` modulo the letter *g* and the `\global` prefix. (File `d:ltdefns.dtx` Date: 2004/02/20 Version v1.3g, lines 139–150)

```
\newgif 472 \pdef\newgif#1{%
         473   {\escapechar\m@ne
         474     \global\let#1\iffalse
         475     \@gif#1\iftrue
         476     \@gif#1\iffalse
         477   }}
```

'Almost' is also in the detail that in this case, which deals with `\global` assignments, we don't have to bother with storing and restoring the value of `\escapechar`: we can do all the work inside a group.

```
\@gif 483 \def\@gif#1#2{%
      484   \protected\@xa\gdef\csname\@xa\@gobbletwo\string#1%
      485   g% the letter g for '\global'.
      486   \@xa\@gobbletwo\string#2\endcsname
      487   {\global\let#1#2}}
```

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

```
496   \count@\escapechar\@escapechar\m@ne
497   \let#1\iffalse
498   \@if#1\iftrue
499   \@if#1\iffalse
500   \escapechar\count@}
```

```
\@if 502 \def\@if#1#2{%
      503   \protected\@xa\def\csname\@xa\@gobbletwo\string#1%
      504   \@xa\@gobbletwo\string#2\endcsname
      505   {\let#1#2}}
```

```
\hidden@iffalse 508 \pdef\hidden@iffalse{\iffalse}
\hidden@iftrue 509 \pdef\hidden@iftrue{\iftrue}
```

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

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

Note that it's just a shorthand. `\global\if<switch>(true|false)` *does* work as expected.

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

Warning. I use it because of very special reasons in gmdoc and in general it is probably not a good idea to make `\refstepcounter` global since it is contrary to the original L^AT_EX approach.

```
\grefstepcounter 530 \pdef\grefstepcounter#1{%
                 531   {\let\protected@edef=\protected@xdef\refstepcounter{#1}}}
```

Naïve first try `\globaldefs=\tw@` raised an error `unknown\command\reserved@e`. The matter was to globalize `\protected@edef` of `\@currentlabel`.

Thanks to using the true `\refstepcounter` inside, it observes the change made to `\refstepcounter` by `hyperref`.

2008/08/10 I spent all the night debugging `\penalty 10000` that was added after a `hypertarget` in vertical mode. I didn't dare to touch `hyperref`'s guts, so I worked it around with ensuring every `\grefstepcounter` to be in `hmode`:

```
\hgrepstepcounter 545 \pdef\hgrepstepcounter#1{%
546   \ifhmode\leavevmode\fi\grefstepcounter{#1}}
```

By the way I read some lines from *The T_EXbook* and was reminded that `\unskip` strips any last skip, whether horizontal or vertical. And I use `\unskip` mostly to replace a blank space with some fixed skip. Therefore define

```
\hunskip 553 \pdef\hunskip{\ifhmode\unskip\fi}
```

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

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

```
\glet 563 \def\glet{\global\let}
```

L^AT_EX provides a very useful `\g@addto@macro` macro that adds its second argument to the current definition of its first argument (works iff the first argument is a no argument macro). But I needed it some times in a document, where `@` is not a letter. So:

```
\gaddtomacro 571 \let\gaddtomacro=\g@addto@macro
```

The redefining of the first argument of the above macro(s) is `\global`. What if we want it local? Here we are:

```
\addto@macro 576 \long\def\addto@macro#1#2{%
580   \edef#1{\@xa\unexpanded\@xa{#1}\unexpanded{#2}}% \edef and \un!
           expanded to double the hashes.
582 }
```

And for use in the very document,

```
\addtomacro 586 \let\addtomacro=\addto@macro
```

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

```
\prependtomacro 589 \long\def\prependtomacro#1#2{%
591   \edef#1{\unexpanded{#2}\@xa\unexpanded\@xa{#1}}}
```

Note that `\prependtomacro` can be prefixed.

```
\addtotoks 595 \long\def\addtotoks#1#2{%
596   #1=\@xa{\the#1#2}}
```

```
\@emptify 599 \newcommand*\@emptify[1]{\let#1=\@empty}
\emptify 600 \@ifdefinable\emptify{\let\emptify\@emptify}
```

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

```
\g@emptify 604 \newcommand*\g@emptify{\global\@emptify}
\gemptify 605 \@ifdefinable\gemptify{\let\gemptify\g@emptify}
```

```
\@relaxen 608 \newcommand*\@relaxen[1]{\let#1=\relax}
\relaxen 609 \@ifdefinable\relaxen{\let\relaxen\@relaxen}
```

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

```
\g@relaxen 613 \newcommand*\g@relaxen{\global\@relaxen}
\grelaxen 614 \@ifdefinable\grelaxen{\let\grelaxen\g@relaxen}
```


`\gm@ifundefined`—a test that doesn't create any hash entry unlike `\@ifundefined`

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

```
\gm@ifundefined 623 \long\def\gm@ifundefined#1{% not \protected because expandable.
629   \ifcsname#1\endcsname% defined...
630   \@xa\ifx\csname_#1\endcsname\relax% but only as \relax—then
        2nd argument.
632     \afterfifi\@firstoftwo%
633     \else% defined and not \relax—then 3rd argument.
634     \afterfifi\@secondoftwo%
635     \fi
636   \else% not defined—then 3rd.
637     \afterfi\@firstoftwo%
638   \fi}

\gm@testdefined 641 \long\def\gm@testdefined#1\iftrue{% This is a macro that expands to \iftrue
        or \iffalse depending on whether its argument is defined in the LATEX
        sense. Its syntax requires an \iftrue to force balancing \ifs with \fis.
646   \csname
647   \ifdefined#1%
648   \ifx#1\relax
649     iffalse%
650   \else_ \iftrue%
651   \fi
652   \else_ iffalse%
653   \fi\endcsname
654 }

\gm@testundefined 656 \long\def\gm@testundefined#1\iftrue{% we expand the last macro two
        levels. We repeat the parameter string to force the proper syntax.
659   \@xa \@xa \@xa \unless\gm@testdefined#1\iftrue}
```

Some 'other' and active stuff

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

```
670 \foone{\catcode`\_ =8_}
\subs 671 {\let\subs=_}

674 \foone{\catcode`\^ =7_}
\sup 675 {\let\sups=^}

677 \foone{\@makeother\_}
\xiiunder 678 {\def\xiiunder{_}}

680 \let\all@unders\xiiunder
681 \foone{\catcode`\_ =8_}
682 {\addtomacro\all@unders{_}}
683 \foone{\catcode`\_ =11_}
684 {\addtomacro\all@unders{_}}
685 \foone{\catcode`\_ =13_}
```

686 `{\addtomacro\all@unders{_}}`
 Now `\all@unders` bears underscores of categories 8, 11, 12 and 13.

`\all@stars` 690 `\def\all@stars{★}`
 691 `\foone{\catcode\★=11}`
 692 `{\addtomacro\all@stars{★}}`
 693 `\foone{\catcode\★=13}`
 694 `{\addtomacro\all@stars{★}}`
 And `\all@stars` bears stars of categories 11, 12 and 13.

698 `\ifdefined\XeTeXversion`
 699 `\chardef_="005F`
 700 `\fi`

`\backquote` 702 `\foone{\@makeother\`}%`
 703 `{\def\backquote{`}}`
 706 `\foone{\catcode\ [=1\@makeother\{`
 707 `\catcode\]=2\@makeother\}}%`
 708 `[%`

`\xiilbrace` 709 `\def\xiilbrace[{}%]`
`\xiirbrace` 710 `\def\xiirbrace[{}%]`
 711 `]%` of `\firstofone`

Note that L^AT_EX's `\@charlb` and `\@charrb` are of catcode 11 ('letter'), cf. The L^AT_EX 2_ε Source file `k`, lines 129–130.

Now, let's define such a smart `_` (underscore) which will be usual `_8` in the math mode and `_12` ('other') outside math.

722 `\foone{\catcode_=\active}`
 723 `{%`
`\smartunder` 724 `\newcommand*\smartunder{%`
 725 `\catcode_=\active`
 726 `\def_{\ifmmode\subs\else_}\fi}}%` We define it as `_` not just as
 % `\xiiunder` because some font encodings don't have `_` at the `\char``
 % `_` position.

732 `\foone{\catcode\!=0`
 733 `\@makeother\!}`
`\xiibackslash` 734 `!\newcommand*!\xiibackslash{\!}`

`\bslash` 738 `\let\bslash=\xiibackslash`

742 `\foone{\@makeother\}%`
`\xiipercen` 743 `{\def\xiipercen{ %}}`

746 `\foone{\@makeother\&%}`
`\xiiand` 747 `{\def\xiand{&}}`

749 `\foone{\@makeother_%}`
`\xiispace` 750 `{\def\xiispace{ %}}`

752 `\foone{\@makeother\#}%`
`\xiihash` 753 `{\def\xiishash{#}}`

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

```

762 \AtBeginDocument {%
763   \ifdefined\xxt@visibleSPACE
764     \let \visibleSPACE\xxt@visibleSPACE
\visibleSPACE@fallback 765     \def \xxt@visibleSPACE@fallback {%
766       \fontspec {Latin_Modern_Mono} \textvisibleSPACE}}%
767   \else
768     \let \visibleSPACE\xiSPACE
769   \fi}

\gmu@activespace 772 \foone\obeyspaces {\def \gmu@activespace {_}}
\activeM 774 \foone\obeylines {\def \activeM {^M}}

```

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

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

```

\@ifnextcat 794 \long\pdef \@ifnextcat#1#2#3{%
798   \def \reserved@d{#1}%
799   \def \reserved@a{#2}%
800   \def \reserved@b{#3}%
801   \futurelet \@let@token \@ifncat}

\@ifncat 804 \def \@ifncat{%
805   \ifx \@let@token \@sptoken
806     \let \reserved@c \@xifncat
807   \else
808     \ifcat \reserved@d \@nx \@let@token
809       \let \reserved@c \reserved@a
810     \else
811       \let \reserved@c \reserved@b
812     \fi
813   \fi
814   \reserved@c}

816 {\def \: {\let \@sptoken=_} \global \:_% this makes \@sptoken a space
      token.

819 \def \: {\@xifncat} \xa\gdef \: {\futurelet \@let@token%
      \@ifncat}}

```

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

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

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

```

\spifletter 833 \def\spifletter{\@ifnextcat_a{\space}{}}
\@ifnextif 836 \long\pdef\@ifnextif#1#2#3{%
    (the future token in \noexpanded, unlike #1)
843   \def\reserved@d{#1}%
844   \def\reserved@a{#2}%
845   \def\reserved@b{#3}%
846   \futurelet\@let@token\@ifnif}

\@ifnif 849 \def\@ifnif{%
850   \ifx\@let@token\@sptoken
851     \let\reserved@c\@xifnif
852   \else
853     \if\reserved@d\@nx\@let@token
854       \let\reserved@c\reserved@a
855     \else% #1 of \@ifnextif is not \if-equivalent the future token. But this
        may be because the future token is active so it would be \if-equivalent
        if not passed through \futurelet. Let's manage this case.
859       \begingroup
860       \edef\gmu@tempa{%
861         \lccode`\@nx~=\reserved@d
862       }\gmu@tempa
863       \lowercase{\endgroup
864         \ifx~\@let@token
865           \let\reserved@c\reserved@a
866         \else
867           \let\reserved@c\reserved@b
868         \fi
869       \fi
870     \fi
871   \reserved@c}

874 {\def\:\let\@sptoken= }\:\% this makes \@sptoken a space token.
876 \def\:\{\@xifnif}\@xa\gdef\:\{\futurelet\@let@token\@ifnif}}

```

But how to peek at the next token to check whether it's an active char? First, we look with `\@ifnextcat` whether there stands a group opener. We do that to avoid taking a whole `{...}` as the argument of the next macro, that doesn't use `\futurelet` but takes the next token as an argument, tests it and puts back intact.

```

\@ifnextac 887 \long\pdef\@ifnextac#1#2{%
888   \@ifnextsingle
889   {\gm@ifnac{#1}{#2}}%
890   {#2}}

\gm@ifnac 892 \long\def\gm@ifnac#1#2#3{%
893   \ifcat\@nx~\@nx#3%
894     \@xa\@firstoftwo
895   \else\@xa\@secondoftwo
896   \fi{#1#3}{#2#3}}

```

Yes, it won't work for an active char `\let` to `{_}`, but it *will* work for an active char `\let` to a char of catcode $\neq 1$. (Is there anybody on Earth who'd make an active char working as `\bgroup` not just recatcode it to `_`?)

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

```
\gmu@lowstar 910 \pdef\gmu@lowstar{%
911   \iffontchar\font"22C6_\char"22C6_\else\gmu@lowstarfake\fi}% 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 `\activation`) are defined in line [7048](#) because they use `\DeclareCommand`.

```
\gmu@tempa 924 \def\gmu@tempa{*}
925 \foone{\catcode`\*=\active}
\gmu@tempb 926 {\def\gmu@tempb{*}% it's defined in line ?? to make * defined (when it was
      undefined, \newcommand's \gm@ifstar test turned true, the next unde-
      fined token was gobbled and raised an error).
930   \let*=\gmu@lowstar}
934 \edef\gmu@tempa{%
935   \long\pdef\@nx\gm@ifstar##1##2{% ]
938     \@nx\@ifnextif\gmu@tempa%
939     {\@nx\@firstoftwo{##1}}%
940     {%
941       \@nx\@ifnextchar\@xa\@nx\gmu@tempb
942       {\@nx\@firstoftwo{##1}}{##2}}%
943   }% of \gm@ifstar.
944 }\gmu@tempa
```

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

```
\@ifnextsingle 951 \long\pdef\@ifnextsingle#1#2{% This macro checks whether the next token
      is able to be picked or is it a braced list of tokens or is it a group closer so there's
      no token to be picked.
955   \@ifnextcat\bgroup{#2}{%
956   \@ifnextcat\egroup{#2}%
958   {#1}}}
```

```
\@ifnextgroup 960 \long\pdef\@ifnextgroup#1#2{% Note this macro turns true both before a group
      opener and before a group closer.
962   \@ifnextsingle{#2}{#1}}
```

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

```
\ignoreactiveM 967 \pdef\ignoreactiveM{%
968   \@ifnextgroup{}{\gmu@checkM}}
970 \foone\obeylines{% we know it's a single token since we use this macro only
      in \@ifnextgroup's 'else'.
```

```

\gmu@checkM 972 \long\pdef\gmu@checkM#1{%
973 \ifx
974 #1\@xa\ignoreactiveM%
975 \else\@xa#1\fi}}

```

Now, define a test that checks whether the next token is a genuine space, `_` that is. First define a CS `\let` such a space. The assignment needs a little trick (*The T_EXbook* appendix D) since `\let`'s syntax includes one optional space after `=`.

```

985 \let\gmu@reserveda\*%
\* 986 \def\*{%
987 \let\*\gmu@reserveda
988 \let\gm@letspace=\_}%
989 \*_\%

```

```

\@ifnextspace 992 \def\@ifnextspace#1#2{%
993 \let\gmu@reserveda\*%
\* 994 \def\*{%
995 \let\*\gmu@reserveda
996 \ifx\@let@token\gm@letspace\afterfi{#1}%
997 \else\afterfi{#2}%
998 \fi}%
999 \futurelet\@let@token\*}

```

First use of this macro is for an active `-` that expands to `---` if followed by a space. Another to make dot checking whether is followed by `~` without gobbling the space if it occurs instead.

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

```

1008 \foone\obeylines{%
\@ifnextMac 1009 \long\pdef\@ifnextMac#1#2{%
1010 \@ifnextchar^^M{#1}{#2}}

```

Storing and restoring the meanings of CSes

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

```

\ifgmu@SMglobal 1017 \newif\ifgmu@SMglobal
\SMglobal 1019 \pdef\SMglobal{\gmu@SMglobaltrue}

```

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

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

```

\StoreMacro 1030 \pdef\StoreMacro{%
1031 \begingroup\makeatletter\gm@ifstar\egStore@MacroSt%
\egStore@Macro}

```

The unstarred version takes a CS and the starred version a text, which is intended for special control sequences. For storing environments there is a special command in line [1177](#).

```

\egStore@Macro 1036 \long\def\egStore@Macro#1{\endgroup\Store@Macro{#1}}

```

```

\egStore@MacroSt 1037 \long\def\egStore@MacroSt#1{\endgroup\Store@MacroSt{#1}}
\Store@Macro 1039 \long\def\Store@Macro#1{%
1040 \escapechar92
1041 \ifgmu@SMglobal\afterfi\global\fi
1042 \@xa\let\csname_/gmu/store/string#1\endcsname#1%
1043 \global\gmu@SMglobalfalse}

\Store@MacroSt 1046 \long\def\Store@MacroSt#1{%
1047 \edef\gmu@smtempa{%
1048 \ifgmu@SMglobal\global\fi
1049 \@nx\let\@xa\@nx\csname/gmu/store/bslash#1\endcsname%
we add backslash because to ensure compatibility
between \ (Re)StoreMacro and \ (Re)StoreMacro*, that
is. to allow writing e.g. \StoreMacro\kitten and
then \RestoreMacro*\kitten} to restore the meaning of \kitten.
1055 \@xa\@nx\csname#1\endcsname}
1056 \gmu@smtempa
1057 \global\gmu@SMglobalfalse}% we wish the globality to be just once.

```

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

The starred version, `\StoreMacro*` works with csnames (without the backslash). It's first used to store the meanings of robust commands, when you may need to store not only `\foo`, but also `\csname_#1\endcsname`.

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

```

\StoreMacros 1073 \long\pdef\StoreMacros{\begingroup\makeatletter\Store@Macros}
\Store@Macros 1074 \long\def\Store@Macros#1{\endgroup
1075 \gmu@setsetSMglobal
1076 \let\gml@StoreCS\Store@Macro
1077 \gml@storemacros#1.}

\gmu@setsetSMglobal 1080 \def\gmu@setsetSMglobal{%
1081 \ifgmu@SMglobal
1082 \let\gmu@setSMglobal\gmu@SMglobaltrue
1083 \else
1084 \let\gmu@setSMglobal\gmu@SMglobalfalse
1085 \fi}

```

And the inner iterating macro:

```

\gml@storemacros 1088 \long\def\gml@storemacros#1{%
\gmu@reserveda 1089 \def\gmu@reserveda{\@nx#1}% My TEX Guru's trick to deal with \fi and
such, i.e., to hide #1 from TEX when it is processing a test's branch without
expanding.
1092 \if\gmu@reserveda.% a dot finishes storing.
1093 \global\gmu@SMglobalfalse
1094 \else
1095 \if\gmu@reserveda,% The list this macro is put before may contain com-
mas and that's O.K., we just continue the work.
1097 \afterfifi\gml@storemacros
1098 \else% what is else this shall be stored.
1099 \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.

```

```

1102     \afterfifi {\gmu@setSMglobal\gml@storemacros}%
1103     \fi
1104     \fi}

    And for the restoring

\RestoreMacro 1110 \pdef\RestoreMacro{%
1111     \begingroup\makeatletter\gm@ifstar\egRestore@MacroSt%
        \egRestore@Macro}

\egRestore@Macro 1113 \long\def\egRestore@Macro#1{\endgroup\Restore@Macro{#1}}
\egRestore@MacroSt 1114 \long\def\egRestore@MacroSt#1{\endgroup\Restore@MacroSt{#1}}

\Restore@Macro 1116 \long\def\Restore@Macro#1{%
1117     \escapechar2
1118     \gmu@ifstored#1{%
1119         \ifgmu@SMglobal\afterfi\global\fi
1120         \@xa\let\@xa#1\csname_/gmu/store/string#1\endcsname
1121         \global\gmu@SMglobalfalse}%
1122     {\unless\ifgmu@quiet
1123         \PackageWarning{gmutils}{\@nx#1 is not stored, I do
            nothing with
1124             it}%
1125         \fi
1126     }%
1127 }

\gmu@ifstored 1129 \long\def\gmu@ifstored#1#2#3{%
1130     \gm@ifundefined{/gmu/store%
1131         \if\relax\@nx#1\else\backslash\fi
1132         \string#1}{#3}{#2}%
1133 }

\gmu@storeifnotyet 1135 \long\pdef\gmu@storeifnotyet#1{%
1136     \if\relax\@nx#1% we check if it's a CS
1137     \gmu@ifstored{#1}{ }\StoreMacro#1}%
1138     \fi}

\Restore@MacroSt 1141 \long\def\Restore@MacroSt#1{%
1142     \gm@ifundefined{/gmu/store\backslash#1}%
1143     {\unless\ifgmu@quiet
1144         \PackageWarning{gmutils}{\backslash#1 is not stored. I~do
            nothing}%
1145         \fi}%
1146     {\edef\gmu@smtempa{%
1147         \ifgmu@SMglobal\global\fi
1148         \@nx\let\@xa\@nx\csname#1\endcsname
1149         \@xa\@nx\csname/gmu/store\backslash#1\endcsname}% cf. the com-
            mentary in line 1049.
1151         \gmu@smtempa}%
1152     \global\gmu@SMglobalfalse}

\RestoreMacros 1155 \long\pdef\RestoreMacros{\begingroup\makeatletter%
        \Restore@Macros}

\Restore@Macros 1157 \long\def\Restore@Macros#1{\endgroup
1158     \gmu@setsetSMglobal

```



```

1159 \let \gml@storeCS \Restore@Macro% we direct the core CS towards restor-
      ing and call the same iterating macro as in line 1077.
1162 \gml@storemacros#1.}

```

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

And to restore *and* use immediately:

```

\StoredMacro 1168 \def \StoredMacro {\begingroup \makeatletter \Stored@Macro}
\Stored@Macro 1169 \long \def \Stored@Macro#1 {\endgroup \Restore@Macro#1#1}

```

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

```

\storedcsname 1172 \def \storedcsname#1 {%
1173   \csname_/gmu/store\backslash#1 \endcsname}

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

```

```

\StoreEnvironment 1177 \pdef \StoreEnvironment#1 {%
1179   \StoreMacro*{#1} \StoreMacro*{end#1}}

```

```

\RestoreEnvironment 1181 \pdef \RestoreEnvironment#1 {%
1183   \RestoreMacro*{#1} \RestoreMacro*{end#1}}

```

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

```

\StoringAndRelaxingDo 1198 \def \StoringAndRelaxingDo {%
1199   \gmu@SMdo@setscope
1200   \long \def \do##1 {%
1201     \gmu@SMdo@scope
1202     \@xa \let \csname_/gmu/store/string##1 \endcsname##1%
1203     \gmu@SMdo@scope \let##1 \relax}}

```

```

\gmu@SMdo@setscope 1205 \def \gmu@SMdo@setscope {%
1206   \ifgmu@SMglobal \let \gmu@SMdo@scope \global
1207   \else \let \gmu@SMdo@scope \relax
1208   \fi
1209   \global \gmu@SMglobalfalse}

```

And here is the counter-definition for restore.

```

\RestoringDo 1218 \long \def \RestoringDo {%
1219   \gmu@SMdo@setscope
1220   \long \def \do##1 {%
1221     \gmu@SMdo@scope
1222     \@xa \let \@xa##1 \csname_/gmu/store/string##1 \endcsname}}

```

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

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

```

\n@melet 1230 \long \def \n@melet#1#2 {%
1231   \edef \gmu@nl@reserveda {%

```

```

1232     \let \@xa \@nx \csname#1 \endcsname
1233     \@xa \@nx \csname#2 \endcsname}%
1234     \gmu@nl@reserveda}

```

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

```

\gn@melet 1238 \long\def\gn@melet#1#2{%
1239     \edef\gmu@nl@reserveda{%
1240         \global\let \@xa \@nx \csname#1 \endcsname
1241         \@xa \@nx \csname#2 \endcsname}%
1242     \gmu@nl@reserveda}

\envirlet 1244 \long\pdef\envirlet#1#2{%
1245     \n@melet {#1} {#2}%
1246     \n@melet {end#1} {end#2}%
1247 }

```

`\DeclareCommand` and `\DeclareEnvironment`

The code of this section is based on the `xparse` package version 0.17 dated 1999/09/10, the version available in \TeX Live 2007-13, in Ubuntu packages at least. Originally considered a stub 'im Erwartung' (Schönberg) for the \LaTeX 3 bundle, it evolved to quite a nice tool I think.

I rewrote the ancient `xparse`'s code to use the usual catcodes (only with `@` a letter) and not to use the `l3dcat` package (which caused an error of undefined CS `\KV@def`). Then I added the `\protected` prefix to the first macro. And I added my concept of `S/T` arg. spec. And of `b` and `B` spec. And the `Q` argument type. And made `\DeclareEnvironment` makes the arguments passed also to the end macro.

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

In the args specification you can use:

- `m` for mandatory argument (braced or not),
- `O{<default>}` for optional argument (in square brackets) with default value `<default>` (which is passed as `#<n>` if the argument is missing),
- `o` equivalent of `O{\NoValue}`: optional argument (in square brackets) lack of which makes the `\NoValue` macro passed as `#<n>`. You can check it with `\If[No]Value(T|F|TF){\#<n>}`,
- `s` for optional star (if star is present, it becomes respective `#<n>`, or else `\NoValue` is at the `#<n>` (unlike in `xparse`!).
- `S{<list>}{<default>}` for a Single token, checks whether on the respective position there's an unbraced token one of `<list>` and returns it on `#<n>` if present or `{<default>}` if absent. The `{<default>}` is a braced optional. If absent, `\NoValue` is the default.
- `T` as 'Token', just an alias for `S`.

Like in `xparse`, `s` is a shorthand for `S{*}`, but *unlike* in `xparse`, in the parsed arguments you get `*` or `\NoValue` as `#<n>`. You can now declare

```

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

```

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

```
raised          untouched
      lowered
```

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

Q $\langle list \rangle$ as 'sequence', a sequence of symbols from $\langle list \rangle$; for the fundamental usage see line 2244. More clearly, an argument specified with **Q** $\langle tokens \rangle$ is a word over the alphabet $\langle tokens \rangle \cup _$ where $_$ is ignored and shall not be passed in the respective # $\langle n \rangle$.

`\DCcoordinate`

c for an optional argument in parentheses. Historically it comes from a 'coordinate' argument and may serve as such: if you declare `\DCcoordinate`, then its catcher will be redefined to check for presence of a comma and pass the argument as $\% \langle before comma \rangle \langle after comma \rangle$ if comma present. `\NoValue` is passed if absent. By default `\DCnocoordinate` is executed that defines the **c** type argument as just an optional in parentheses.

`\DCnocoordinate`

C $\langle default \rangle$ as **c**, with default value.

b for for an *optional* argument in curly braces. That's strange for anyone acquainted with L^AT_EX and contrary to its basic convention, but practised by Til Tantau in the beamer class. If missing, `\NoValue` is passed as # $\langle n \rangle$ and therefore all the tests `If[No]Value(T|F|TF)` apply.

B $\langle default \rangle$ for a braced *optional* argument with the default value $\langle default \rangle$.

K $\langle \#-string \rangle$ [$\langle replacement \rangle$] for a *mandatory* Knuthian delimited or undelimited macro parameters. This concept comes from Stephen Hicks' suggestion at BachoT_EX 2009 of implementing arguments delimited with # {. So, in the mandatory first argument to **K** you give an arbitrary parameters string as described in Chapter 20 of *The T_EXbook*. If you don't provide the replacement, only #1 of those parameters will be passed to the core macro of your command as # $\langle n \rangle$. In both arguments you use single # char.

G $\langle pair of tokens \rangle$ [default value] a **General** catcher of an optional. For instance, to declare an optional in angles (used e.g. in Till Tantau's beamer), declare `G { < > }`. Again, if second argument is absent, `\NoValue` is assumed.

A (for 'angles') a shorthand for `G { < > }` (accepts the second braced as the default value).

`\afterassignment { <register > }` a pseudo-argument that in fact *interrupts* parsing arguments to let an assignment to $\langle register \rangle$. The default $\langle register \rangle$ is `\@tempcnta`. For example,

```
\DeclareCommand\llf{\afterassignment
  {\@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, $\langle register \rangle$ may be empty `{}` and then each time our command is used the left side of the assignment has to be given explicitly and may even be different each time.

Doesn't add anything to the arguments' list.

This pseudo-argument type should be considered experimental.

Note that you could limit acceptable values of a mandatory or an optional-in-brackets argument to a list inside definition of the command, using `\IfAmong ... \among...` defined in line 399.

The **S/T**, **s** and **Q** arguments are always 'long', they allow `\par` as their value that

is. They have to be unbraced to be parsed so there's no danger of "runaway argument".
 By default, all the `c`, `C`, `o`, `O`, `m`, `b` and `B` are 'short', they don't allow `\par` in them that is. Note however that `\par` is allowed in the default values. If you wish to allow `\par` for all the arguments, you can say `\DeclareCommand\mycommand!...` — the optional `!` makes all the arguments 'long'. Instead of `!` you can use `L` or `l` for 'long' or just `\long` itself.

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

`Pp!lL \long \par` to make a particular argument 'long', allowing `\par` in 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:

`& 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 `\detokenizes` its argument.

`\global \outer` If you wish to define your command `\globally`, you can specify `\DeclareCommand% \mycommand\global`. If you wish to forbid usage of your command in arguments of macros, add the `\outer` prefix. As with original 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 L^AT_EX mechanism of `\@bsphack`—`\@esphack` and to use it I provide yet another 'prefix': if you type `W` or `w` (for 'white') or `I` or `i` (for 'invisible') or, if you prefer a prefix-like,

`iIW \sphack`

`\sphack`¹ 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`
`\DCMessagesfalse`
`\DCMessagestrue`

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 `\Decla|recommand` 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 ϵ -`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 `\DeclareCom|mand`.

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.

`\@temptokenb`

1582 `\unless\ifdefined\@temptokenb`
 1583 `\newtoks\@temptokenb`

¹ 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.

```

1584 \fi
\ifdc@alllong@ 1586 \newif\ifdc@alllong@_ % for an option of all arguments long.
\ifdc@quiet@ 1587 \newif\ifdc@quiet@_ % for suppressing the message of using of declared com-
mand.

\IfDCMessages 1590 \newif\IfDCMessages_ % a global switch to suppress the message about parsing.
\dc@long@yes 1594 \def\dc@long@yes {P}
\dc@arguments 1596 \newtoks \dc@arguments
\dc@catchernum 1598 \newcount \dc@catchernum
\dc@argnum 1599 \newcount \dc@argnum
\ifdc@ignore 1601 \newif\ifdc@ignore
1605 \long\pdef\DeclareCommand#1#2#3 {%
% #1 command to be defined,
% #2 arguments specification,
% #3 definition body.
\dc@catchernum 1616 \dc@catchernum\z@
1617 \dc@argnum\z@
1618 \toks@{}% in this register we will store the created sequence of argument catch-
ers.
1622 \@temptokena\toks@_ % in this register we will store the sequence of #1#2...
1623 \@temptokenb\toks@_ % in this register we will store the sequences of m's.
1627 \emptify\dc@long@
1628 \dc@ignorefalse
1629 \dc#2X% X is the sentinel of parsing.
1630 \protected\edef#1{%
1631 \nx\dc@_{}_{}{\the\toks@}%
1632 \xa\nx\csname\string#1\endcsname
1633 \nx_#1%
1634 }%
1635 \edef\gmu@reserveda{%
1636 \long\def\xa\nx\csname\string#1\endcsname
1637 \the\@temptokena{%
1639 \unexpanded{#3}}}%
1640 \gmu@reserveda}% of the 'draft' \DeclareCommand.

\dc@ 1643 \long\def\dc@
1644 #1% the argument catchers in a pair of braces (their arguments may contain \par
so the macro is long).
1646 #2% \thecommand
1647 #3% \thecommand
1648 {%
1649 \ifx\protect\@typeset@protect
1650 \xa\@firstofone
1651 \else
1652 \protect#3\xa\@gobble
1653 \fi
1654 {\dc@arguments{#2}#1\the\dc@arguments}}

\dc 1657 \def\dc#1{% The main parsing macro.
1658 \ifx#1X% if we meet the sentinel we stop. Note there may be some m's in
% \@temptokenb and we might not to care about them since they denote

```

mandatory arguments and as such they will be grabbed by
 % `\<declared command>`. However, if any of them contains `\par` (while it
 shouldn't), then it wouldn't be noticed since `\<declared command>`
 is always long. Therefore:

```

1666 \dc@add@ms
1667 \else
1668 \ifx#1>%
1669 \let\next@dc\grab@prefix
1670 \else% not X neither >
1671 \if@dc@alllong@\let\@dc@long@\@dc@long@yes\fi
1672 \ifx#1\afterassignment\@dc@ignoretrue\fi% if we parse the
    assigned pseudo-argument, we don't want to add anything to the ar-
    guments' toks register so we apply general mechanism of ignoring an
    argument.
1676 \ifx#1\gobblespace\@dc@ignoretrue\fi% again, as above for ig-
    noring spaces.
1678 \ifnum\ifx#1m\else1\fi\ifx\@dc@long@\@empty\else\
    2\fi
1679 \if@dc@ignore\else\fi=%\@m% we check whether #1 is
    % m and not prefixed.
1681 \addto@hook\@temptokenb@m% as you see, \@temptokenb serves
    as a storage of m's.
1683 \else% (not m) or prefixed
1684 \dc@add@ms
1685 \if@dc@ignore\addtotoks\toks@\@dc@ignorearg\fi
1686 \@xa\addtotoks\@xa\toks@\@xa{%
1687 \csname\ArgumentCatcher@\@dc@long\detokenize{#1}\endcsname}
    argument catcher's name is \ArgumentCatcher@[P >]<arg.\
    % type>.
1689 \@temptokenb{}%
1690 \fi% of \if short m or not
1691 \advance\@dc@catchernum\@ne
1692 \unless\if@dc@ignore
1693 \advance\@dc@argnum\@ne
1694 \@temptokena\@xa{%
1695 \the\@xa\@temptokena\@xa##\the\@dc@argnum}%
1696 \else
1697 \@dc@ignorefalse
1698 \fi
1699 \IfAmong#1\among{ABCO}%
1700 {\let\next@dc\grab@default}%
1701 {\IfAmong#1\among{GQST}%
1702 {\let\next@dc\grab@defaults}% G,Q,S and T have second argu-
    ment, braced optional, which is the default value (\NoValue by
    default).
1704 {\gmu@if_x_{K#1}%
1705 {\let\next@dc\@dc@define@K}%
1706 {\gmu@if_x{\afterassignment#1}%
1707 {\let\next@dc\dc@grab@afterass}%
1708 {\let\next@dc\@dc}%
1709 }%
1710 }%
1711 }%
  
```

```

1712     \unless\ifx\next@dc\@dc@define@K
1713     \emptify\@dc@long@
1714     \fi
1715 \fi_□% of \if#1>.
1716     \@xa\next@dc
1717 \fi
1718 }%

\@dc@add@ms 1720 \def\@dc@add@ms{%
1721   \@xa\@ifempty\@xa{\the\@temptokenb}%
1722   }%
1723   {\toks@\@xa{%
1724     \the\@xa\toks@
1725     \csname_□ArgumentCatcher@\the\@temptokenb\endcsname}}}}

\grab@default 1728 \long\def\grab@default#1{% the default value or the list of tokens for S may
              contain \par (why not?), that's why we make this macro \long.
1731   \toks@\@xa{\the\toks@{#1}}%
1732   \let\next@dc\@dc
1733   \@dc
1734 }

\grab@defaults 1736 \long\def\grab@defaults#1{% default (when default is absent)
1737   \toks@\@xa{\the\toks@{#1}}%
1738   \@ifnextchar\bgroup
1739   {\grab@default}%
1740   {\grab@default{\NoValue}}%
1741 }

\dc@grab@afterass 1743 \def\dc@grab@afterass{%
1744   \@ifnextchar\bgroup
1745   {\grab@default}%
1746   {\grab@default{\@tempcnta}}%
1747 }

\@dc@add@argum 1750 \long\def\@dc@add@argum{%
1751   \addtotoks\@dc@arguments}

1753 \StoreMacro\@dc@add@argum

\@dc@ignore@arg 1755 \pdef\@dc@ignore@arg{%
\@dc@add@argum 1756   \long\def\@dc@add@argum##1{%
1757     \RestoreMacro\@dc@add@argum}}

    Parsing of Knuthian parameters string is easier to implement with full-feature \De|
    clareCommand so we postpone it till line 2363

\grab@prefix 1763 \def\grab@prefix#1{%
1764   \IfIntersect{#1}{\long!LL\par_□Pp}%
\@dc@long@ 1765   {\def\@dc@long@{P}}}%
1766   \IfIntersect{#1}{Ii}%
1767   {\@dc@ignore@true}}%
1768   \@dc
1769 }

\ArgumentCatcher@ 1771 \long\def\ArgumentCatcher@o#1\@dc@arguments{% parsing of an optional
              with no default
1772   \@ifnextchar[

```



```

\@dc@parse@next 1773  {\def \@dc@parse@next {#1} \@dc@arguments}% the last macro has
                  to be short so we can't give it all the arguments' catchers since some de-
                  fault may contain \par. Therefore we hide all the argument catchers in
                  % \@dc@parse@next.
1778  {\@dc@addargum\NoValue#1 \@dc@arguments}}

\ArgumentCatcher@o@ 1780 \def \ArgumentCatcher@o@[#1] {%
1781  \@dc@addargum {{#1}} \@dc@parse@next \@dc@arguments}

\ArgumentCatcher@Po 1783 \long \def \ArgumentCatcher@Po#1 \@dc@arguments {% parsing of an optional
                  with no default.
1784  \@ifnextchar [%
1785  {\ArgumentCatcher@Po@{#1}}%
1786  {\@dc@addargum\NoValue#1 \@dc@arguments}}

\ArgumentCatcher@Po@ 1788 \long \def \ArgumentCatcher@Po@#1[#2] {%
1789  \@dc@addargum {{#2}} #1 \@dc@arguments}

\ArgumentCatcher@O 1791 \long \def \ArgumentCatcher@O#1#2 \@dc@arguments {% parsing of optional
                  with default value.
                  % #1 the default value,
                  % #2 the tail of args parser.
1797  \@ifnextchar [%
\@dc@parse@next 1798  {\def \@dc@parse@next {#2} \@dc@arguments}% note that even when
                  an optional is short, its default may contain \par.
1800  {\@dc@addargum {{#1}} #2 \@dc@arguments}}

\ArgumentCatcher@PO 1803 \long \def \ArgumentCatcher@PO#1#2 \@dc@arguments {% parsing of optional
                  with default value.
                  % #1 the default value,
                  % #2 the tail of args parser.
1809  \@ifnextchar [%
1810  {\ArgumentCatcher@Po@#2}% note that even when an optional is short, its de-
                  fault may contain \par.
1812  {\@dc@addargum {{#1}} #2 \@dc@arguments}}

\ArgumentCatcher@b 1815 \long \def \ArgumentCatcher@b#1 \@dc@arguments {% parsing of an optional
                  braced short argument.
1816  \@ifnextcat \bgroup
\@dc@parse@next 1817  {\def \@dc@parse@next {#1} \@dc@arguments}% the last macro has
                  to be short so we can't give it all the arguments' parsers since some de-
                  fault may contain \par. Therefore we hide all the argument catchers in
                  % \@dc@parse@next.
1821  {\@dc@addargum\NoValue#1 \@dc@arguments}}

\ArgumentCatcher@Pb 1824 \long \def \ArgumentCatcher@Pb#1 \@dc@arguments {% parsing of a braced
                  and long optional with \NoValue default.
1826  \@ifnextcat \bgroup
1827  {\ArgumentCatcher@Pm#1 \@dc@arguments}%
1828  {\@dc@addargum\NoValue#1 \@dc@arguments}%
1829  }

\ArgumentCatcher@B 1831 \long \def \ArgumentCatcher@B#1#2 \@dc@arguments {% parsing of a braced
                  optional with default value.
                  % #1 the default value,
                  % #2 the tail of args catchers.

```

```

1837 \ifnextcat\bgroup
\@dc@parse@next 1838 {\def\@dc@parse@next{#2}\ArgumentCatcher@m@}% note that even when
                an optional is short, its default may contain \par.
1840 {\@dc@addargum{#1}#2\@dc@arguments}
\ArgumentCatcher@PB 1842 \long\def\ArgumentCatcher@PB#1#2\@dc@arguments{% parsing of a braced
                and long optional with default value.
                #1 the default value,
                #2 the tail of args catchers.
1849 \ifnextcat\bgroup
1850 {\ArgumentCatcher@Pm#2\@dc@arguments}%
1851 {\@dc@addargum{#1}#2\@dc@arguments}
\ArgumentCatcher@SE 1854 \long\def\ArgumentCatcher@SE{% parsing of a Single continued:
1855 #1% the list to search #4 on,
1856 #2% the default value,
1857 #3% the tail of args parsers,
1858 #4% the token we search in #1 (it's an unbraced single token as we checked in line
                1873).
1860 {%
1861 \IfAmong#4\among{#1}%
1862 {\@dc@addargum{#4}#3\@dc@arguments}%
1863 {\@dc@addargum{#2}#3\@dc@arguments#4}%
1864 }
\ArgumentCatcher@S 1866 \long\def\ArgumentCatcher@S_{}% parsing of a Single token. It's always long
                since we look for a single unbraced token so there is no danger of "runaway
                argument".
1869 #1% the list we search optional token on,
1870 #2% the default value,
1871 #3% the tail of args parser.
1872 \@dc@arguments{%
1873 \ifnextsingle
1875 {\ArgumentCatcher@SE{#1}{#2}{#3}}%
1876 {\@dc@addargum{#2}#3\@dc@arguments}% if we parse an opening or
                closing brace, then we are sure it's not any expected Single.
1879 }
1882 \let\ArgumentCatcher@PS\ArgumentCatcher@S_{}% as above: we always act
                'long' with single tokens.
1883 \let\ArgumentCatcher@T\ArgumentCatcher@S_{}% we allow T (for Token) as
                an alias of S.
1887 \edef\ArgumentCatcher@s{%
1888 \@nx\ArgumentCatcher@S{\@xa\@nx\all@stars}{\@nx\NoValue}}% the
                s arg spec is a shorthand for S{*}. Except the star may be of any category
                from {11, 12, 13}.
1891 \let\ArgumentCatcher@Ps\ArgumentCatcher@s
\ArgumentCatcher@Q 1893 \long\def\ArgumentCatcher@Q% parsing of a seQquence of tokens from the list.
                It's always long since we look for unbraced tokens so there is no danger of
                "runaway argument".
1896 #1% the list we search optional tokens on,
1897 #2% the default value,
1898 #3% the tail of args parser.

```

```

1899 \@dc@arguments_␣% delimiter
1900 {%
\@dc@parse@next 1901 \def \@dc@parse@next {#3}%
1902 \emptify \@dc@seSequence
1903 \ArgumentCatcher@Q@ {#1} {#2} }

\ArgumentCatcher@PQ 1906 \let \ArgumentCatcher@PQ \ArgumentCatcher@Q

\ArgumentCatcher@Q@ 1908 \long \def \ArgumentCatcher@Q@#1#2 {% list, default
1910 \@ifnextsingle
1911 {\ArgumentCatcher@Q@@ {#1} {#2} }%
1912 {\@dc@seSequence@finish {#2} }%
1913 }

\ArgumentCatcher@Q@@ 1915 \long \def \ArgumentCatcher@Q@@#1#2#3 {% list, default, token to be checked
we only launch this macro when not before opening curly brace. #1 is the list
of acceptable values, #2 the default value, #3 is the token to be sought on the
list.
1920 \IfAmong_␣#3 \among {#1} %
1921 {\addtomacro \@dc@seSequence#3%
1922 \ArgumentCatcher@Q@ {#1} {#2} }%
1923 {\@dc@seSequence@finish {#2} #3} %
1924 }

\@dc@seSequence@finish 1926 \long \def \@dc@seSequence@finish#1 {%
\@dc@seSequence 1928 \ifx \@dc@seSequence \@empty \def \@dc@seSequence {#1} \fi
1929 \@xa \@dc@addargum \@xa {\@xa {\@dc@seSequence} }%
1931 \@dc@parse@next \@dc@arguments_␣}

\ArgumentCatcher@c 1934 \long \def \ArgumentCatcher@c#1 \@dc@arguments {%
1935 \@ifnextchar ( %
\@dc@parse@next 1936 {\def \@dc@parse@next {#1} \ArgumentCatcher@c@}%
1937 {%
1940 \@dc@addargum {\NoValue} #1 \@dc@arguments }%
1941 }

\DCcoordinate 1943 \def \DCcoordinate {%
\ArgumentCatcher@c@ 1944 \def \ArgumentCatcher@c@ (##1) {%
1945 \IfAmong, \among {##1} %
1946 {\@xa \@dc@addargum \@dc@commasep##1 \@dc@commasep}%
1947 {\@dc@addargum {##1} } \@dc@parse@next \@dc@arguments }%

\@dc@commasep 1949 \def \@dc@commasep##1, ##2 \@dc@commasep { { {##1} {##2} } } %

\ArgumentCatcher@Pc@ 1951 \long \def \ArgumentCatcher@Pc@ (##1) {%
1952 \IfAmong, \among {##1} %
1953 {\@xa \@dc@addargum \@dc@commasep##1 \@dc@commasep}%
1954 {\@dc@addargum {##1} } \@dc@parse@next \@dc@arguments }%
1955 }

\DCnocoordinate 1957 \def \DCnocoordinate {%
\ArgumentCatcher@c@ 1958 \def \ArgumentCatcher@c@ (##1) {%
1959 \@dc@addargum {##1} } \@dc@parse@next \@dc@arguments }%

\ArgumentCatcher@Pc@ 1961 \long \def \ArgumentCatcher@Pc@ (##1) {%
1962 \@dc@addargum {##1} } \@dc@parse@next \@dc@arguments }%
1963 }

1965 \DCnocoordinate

```

```

\ArgumentCatcher@C 1967 \long\def\ArgumentCatcher@C#1#2\@dc@arguments{%
1968   \@ifnextchar(%
\dc@parse@next 1969   {\def\dc@parse@next{#2}\ArgumentCatcher@c@}%
1970   {\@dc@addargum{{#1}}#2\@dc@arguments}}

\ArgumentCatcher@Pc 1973 \long\def\ArgumentCatcher@Pc#1\@dc@arguments{%
1974   \@ifnextchar(%
\dc@parse@next 1975   {\def\dc@parse@next{#1}\ArgumentCatcher@Pc@}%
1976   {\@dc@addargum{{\NoValue}}#1\@dc@arguments}%
1977 }

\ArgumentCatcher@PC 1979 \long\def\ArgumentCatcher@PC#1#2\@dc@arguments{%
1980   \@ifnextchar(%
\dc@parse@next 1981   {\def\dc@parse@next{#2}\ArgumentCatcher@Pc@}%
1982   {\@dc@addargum{{#1}}#2\@dc@arguments}}

\ArgumentCatcher@Pm 1985 \long\def\ArgumentCatcher@Pm#1\@dc@arguments#2{%
1986   \@dc@addargum{{#2}}#1\@dc@arguments}

\gmu@hashes 1988 \def\gmu@hashes#1#2{% this is a fully expandable loop analogous to that of The
    \epsilon-TeX Manual p. 9.
1990   \ifnum#1<#2%
1991   ####\number#1
1992   \expandafter\gmu@hashes
1993   \expandafter{\number\numexpr#1+1\expandafter}%
1994   \expandafter{\number#2\expandafter}%
1995   \fi}% of \gmu@hashes.

\gmu@hashesbraced 1997 \def\gmu@hashesbraced#1#2{%
1998   \ifnum#1<#2%
1999   ####\number#1}%
2000   \expandafter\gmu@hashesbraced
2001   \expandafter{\number\numexpr#1+1\expandafter}%
2002   \expandafter{\number#2\expandafter}%
2003   \fi}% of \gmu@hashesbraced.

2005 \@tempcnta=1
2006 \@whilenum\@tempcnta<9\do{%
2007   \edef\gmu@ms{\romannumeral\numexpr\@tempcnta*1000\@}%
2011   \edef\gmu@tempa{%
2012     \long\def\@xa\@nx\csname
2013       ArgumentCatcher@\gmu@ms
2014     \endcsname
2015     #1\@nx\@dc@arguments{\def\@nx\dc@parse@next{##1}%
2016     \@xa\@nx\csname
2017       ArgumentCatcher@\gmu@ms\@%
2018     \endcsname}%
2020   \def\@xa\@nx\csname
2021   ArgumentCatcher@\gmu@ms\@%
2022   \endcsname
2023   \gmu@hashes1{\numexpr\@tempcnta+1}%
2024   {\@nx\@dc@addargum{%
2025     \gmu@hashesbraced1{\numexpr\@tempcnta+1}}%
2026     \@nx\dc@parse@next\@nx\dc@arguments
2027   }% of \ArgumentCatcher@<m's>@.
2028   }% of \edef.

```

```

2029 \gmu@tempa
2030 \advance\@tempcnta1\relax
2031 }% of the loop.

```

The loop above defines 8 pairs of macros as we see thanks to the code below:

```

2035 \if_1_1
2036 \@tempcnta\@ne
2037 \@whilenum\@tempcnta<9\do{%
2038 \edef\gmu@ms{\romannumeral\numexpr\@tempcnta*1000}%
2039 \typeout{\bslash_ArgumentCatcher@\gmu@ms:%
2040 ^^J\@xa\meaning\csname_ArgumentCatcher@\gmu@ms%
\endcsname}%
2041 \typeout{\bslash_ArgumentCatcher@\gmu@ms_@:%
2042 ^^J\@xa\meaning\csname_ArgumentCatcher@\gmu@ms_@%
\endcsname}%
2043 \advance\@tempcnta\@ne
2044 }
2045 \fi

```

The code above produces on the terminal:

```

\ArgumentCatcher@m:
macro:#1\@dc@arguments ->\def \@dc@parse@next
  {#1}\ArgumentCatcher@m@
\ArgumentCatcher@m@:
macro:#1->\addto@hook \@dc@arguments {{#1}}\@dc@parse@next
  \@dc@arguments
\ArgumentCatcher@mm:
macro:#1\@dc@arguments ->\def \@dc@parse@next
  {#1}\ArgumentCatcher@mm@
\ArgumentCatcher@mm@:
macro:#1#2->\addto@hook \@dc@arguments
  {{#1}{#2}}\@dc@parse@next \@dc@arguments
\ArgumentCatcher@mmm:
macro:#1\@dc@arguments ->\def \@dc@parse@next
  {#1}\ArgumentCatcher@mmm@
\ArgumentCatcher@mmm@:
macro:#1#2#3->\addto@hook \@dc@arguments
  {{#1}{#2}{#3}}\@dc@parse@next \@dc@arguments
\ArgumentCatcher@mmmm:
macro:#1\@dc@arguments ->\def \@dc@parse@next
  {#1}\ArgumentCatcher@mmmm@
\ArgumentCatcher@mmmm@:
macro:#1#2#3#4->\addto@hook \@dc@arguments
  {{#1}{#2}{#3}{#4}}\@dc@parse@next \@dc@arguments
\ArgumentCatcher@mmmmm:
macro:#1\@dc@arguments ->\def \@dc@parse@next
  {#1}\ArgumentCatcher@mmmmm@
\ArgumentCatcher@mmmmm@:
macro:#1#2#3#4#5->\addto@hook \@dc@arguments
  {{#1}{#2}{#3}{#4}{#5}}\@dc@parse@next \to
ks@
\ArgumentCatcher@mmmmmm:

```

```

macro:#1\@dc@arguments ->\def \@dc@parse@next
  {#1}\ArgumentCatcher@mmmmmmmm@
\ArgumentCatcher@mmmmmmmm@:
macro:#1#2#3#4#5#6->\addto@hook \@dc@arguments
  {{#1}{#2}{#3}{#4}{#5}{#6}}\@dc@parse@ne
xt \@dc@arguments
\ArgumentCatcher@mmmmmmmm@:
macro:#1\@dc@arguments ->\def \@dc@parse@next
  {#1}\ArgumentCatcher@mmmmmmmmmm@
\ArgumentCatcher@mmmmmmmmmm@:
macro:#1#2#3#4#5#6#7->\addto@hook \@dc@arguments
  {{#1}{#2}{#3}{#4}{#5}{#6}{#7}}\@dc@pa
rse@next \@dc@arguments
\ArgumentCatcher@mmmmmmmmmm@:
macro:#1\@dc@arguments ->\def \@dc@parse@next
  {#1}\ArgumentCatcher@mmmmmmmmmmmm@
\ArgumentCatcher@mmmmmmmmmmmm@:
macro:#1#2#3#4#5#6#7#8->\addto@hook \@dc@arguments
  {{#1}{#2}{#3}{#4}{#5}{#6}{#7}{#8}}\@
dc@parse@next \@dc@arguments

```

ArgumentCatcher@mmmmmmmmmm

```

2087 \def\ArgumentCatcher@mmmmmmmmmm\the\@dc@arguments
2088 #1#2#3#4#5#6#7#8#9{% yes, it has to be delimited (well, actually: started) by
  % \the\@dc@arguments not only \@dc@arguments. And it doesn't need
  to be long and launch inner short macro because there's nothing more to catch.

```

```

2092 \@dc@addargum{%
2093   {#1}{#2}{#3}{#4}{#5}{#6}{#7}{#8}{#9}}\the\@dc@arguments}

```

\ArgumentCatcher@Pm

```

2096 \long\def\ArgumentCatcher@Pm#1\@dc@arguments#2{% catcher of a long
  mandatory.
2097   \@dc@addargum{{#2}}#1\@dc@arguments}

```

\ArgumentCatcher@K

```

2100 \long\def\ArgumentCatcher@K_ % a Knuthian delimited or undelimited argu-
  ments parsed and passed to the command's body in a digest (Knuthian re-
  placement)

```

```

2103 #1% the particular Knuthian macro

```

```

2104 #2% the tail of args parser.

```

```

2105 \@dc@arguments_ % tail delimiter

```

\@dc@parse@next

```

2106 {\def\@dc@parse@next{#2}#1}

```

```

2108 \let\ArgumentCatcher@PK\ArgumentCatcher@K

```

\gmu@twostring

```

2110 \long\def\gmu@twostring#1#2{\string#1\string#2}

```

\ArgumentCatcher@G

```

2112 \long\def\ArgumentCatcher@G_ % general catcher

```

```

2113 #1% left and right delimiter

```

```

2114 #2% default value

```

```

2115 #3% the tail of args parser.

```

```

2116 \@dc@arguments_ % tail delimiter

```

```

2117 {%

```

```

2118   \edef\@dc@Gname{ArgumentCatcher@G\gmu@twostring#1}%

```

```

2119   \unless\ifcsname_\@dc@Gname_\endcsname

```

```

2120     \@xa\def\csname_\@dc@Gname_\@xa\endcsname

```

```

2121     \@dc@Gparams#1{\@dc@addargum{{##2}}##1\@dc@arguments}%

```

```

2122   \fi

```

```

2123   \@xa\@ifnextchar\@firstoftwo#1%

```

```

\dc@parse@next 2124 {\def\dc@parse@next{#3}% we hide possible \pars.
2125 \csname\@dc@Gname\endcsname\dc@parse@next}%
2126 {\@dc@addargum{#2}}#3\@dc@arguments}%
2127 }

\ArgumentCatcher@PG 2130 \long\def\ArgumentCatcher@PG#1#2#3#4#5% general catcher
2131 #1% left and right delimiter
2132 #2% default value
2133 #3% the tail of args parser.
2134 \@dc@arguments#4#5% tail delimiter
2135 {%
2136 \edef\@dc@Gname{ArgumentCatcher@PG\gmu@twostring#1}%
2137 \unless\ifcsname\@dc@Gname\endcsname
2138 \long\@xa\def\csname\@dc@Gname\@xa\endcsname
2139 \@dc@Gparams#1{\@dc@addargum{##2}}##1\@dc@arguments}%
2140 \fi
2141 \@xa\@ifnextchar\@firstoftwo#1%
2142 {\csname\@dc@Gname\endcsname{#3}}%
2143 {\@dc@addargum{#2}}#3\@dc@arguments}%
2144 }

\@dc@Gparams 2147 \long\def\@dc@Gparams#1#2{##1##2}% 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.

\ArgumentCatcher@A 2153 \long\def\ArgumentCatcher@A{\ArgumentCatcher@G{<>}}
\ArgumentCatcher@PA 2154 \long\def\ArgumentCatcher@PA{\ArgumentCatcher@PG{<>}}

\ArgumentCatcher@a 2156 \long\def\ArgumentCatcher@a{\ArgumentCatcher@G{<>}}{\NoValue}}
\ArgumentCatcher@Pa 2157 \long\def\ArgumentCatcher@Pa{\ArgumentCatcher@PG{<>}}{\NoValue}}

\NoValue 2160 \def\NoValue{-NoValue-}

\IfNoValueTF 2162 \long\def\IfNoValueTF#1{%
2168 \@xa\ifx\@firstofmany#1\@empty\@nil\NoValue\afterfi%
\@firstoftwo
2169 \else\afterfi\@secondoftwo
2170 \fi}

\IfNoValueT 2172 \long\def\IfNoValueT#1#2{\IfNoValueTF{#1}{#2}\@empty}
\IfNoValueF 2174 \long\def\IfNoValueF#1#2{\IfNoValueTF{#1}\@empty{#2}}
\IfValueTF 2176 \long\def\IfValueTF#1#2#3{\IfNoValueTF{#1}{#3}{#2}}
\PutIfValue 2178 \long\def\PutIfValue#1{\IfValueT{#1}{#1}}
\IfValueT 2181 \let\IfValueT\IfNoValueF
\IfValueF 2184 \let\IfValueF\IfNoValueT
\IfLong 2186 \long\pdef\IfLong#1{%
% #1 the argument to check for \par,
% #2 what if \par found,
% #3 what if \par not found.

```

```

2193 \edef\gmu@reserveda{\detokenize{#1}}%
2194 \edef\gmu@reserveda{%
2195   \IfAmong\string\par\@nx\among{\gmu@reserveda}}%
2196 \gmu@reserveda}

\afterassignment pseudo-argument

2201 \long\@namedef{ArgumentCatcher@\detokenize{%
\afterassignment}}%
2202 #1% register used in the assignment
2203 #2% the tail of args parser.
2204 \@dc@arguments{%
\@dc@parse@next 2205 \def\@dc@parse@next{%
2206   \@dc@addargum\NoValue\@% to make ignoring mechanism work
2207   #2\@dc@arguments}%
2208 \afterassignment\@dc@parse@next
2209 #1}% optional space and = is left at user's discretion. In fact, #1 may be empty
and then each time our command is used the left side of the assignment has
to be given explicitly and may even be different each time.

2214 \n@melet{ArgumentCatcher@P\detokenize{\afterassignment}}
2215 {ArgumentCatcher@\detokenize{\afterassignment}}

\gobblespace pseudo-argument: useful between optional and Knuthian-type ar-
guments and after all parsing. Equivalent >iT{\somedummymacro}. At first I wanted
mark it \ignorespaces but the gobbling doesn't expand macros: it doesn't use \ig|
norespaces but \@ifnextchar's space trick.

2223 \long\@namedef{ArgumentCatcher@\detokenize{\gobblespace}}%
2224 {\ArgumentCatcher@S{\dc@gobblespace@dummy}{\NoValue}}

2226 \n@melet{ArgumentCatcher@P\detokenize{\gobblespace}}
2227 {ArgumentCatcher@\detokenize{\gobblespace}}

\dc@gobblespace@dummy 2229 \def\dc@gobblespace@dummy{\dc@gobblespace@dummy}

2231 \let\gobblespace\dc@gobblespace@dummy

end of catchers' definitions

\DeclareCommand 2237 \DeclareCommand\DeclareCommand\@% This is the final version by now. Note
we introduce the 'prefixes' at the #2 position only at this point so we have yet
to prefix every argument specifier separately. Note also we only here 'teach'
\DeclareCommand to pass the name of its main argument in the \dc@bsname
macro.

2242 >Pm\@% (1) command to be defined (may be \par if you really
wish),
2244 Q{\global\protected\outer\long!lL.qQiIwW\sphack\envhack}% (2)
an optional seQuence of ε-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...
% \@esphack2,
2257 >Pm\@% (3) arguments specification (may contain \par),

```

² To deal with an 'invisible' environment, there is one more acceptable value of this argument: `\envhack`, which suppresses placing `\@bsphack`—`\@esphack` and places `\@ignoretrue` instead in the end macro.


```

2258 >Pm_ (4) the definition body; may contain nearly anything including \par and
      tests for presence/absence of arguments
      % \If[No]Value(T|F|TF) and for their longness \IfLong; you refer to the
      arguments with #<n>.
2262 } {%
2270 \edef\dc@bsname{\string#1}% we'll use it in K-type arg. catcher and to
      allow #1 == \par (\PackageWarning is short and \typeout too!).
2273 \gm@testdefined#1\iftrue
2274 \IfDCMessages
2275 \PackageWarning{gmutils_ (gmparse)}{Redefining_ command_
      \dc@bsname\space
2276 with_\string\DeclareCommand}%
2277 \fi
2278 \else
2279 \IfDCMessages
\dc@bsname 2280 \typeout{^^J\string\DeclareCommand\dc@bsname\space_
      (new)^^J}%
2281 \fi
2282 \fi

```

First we parse the declaration options to know whether all the arguments are to be long and whether we should suppress the message “Parsing arguments for...”.

```

2287 \@dc@alllong@false
2288 \@dc@quiet@false
2290 \IfIntersect{#2}{\long!LL}{%
2292 \@dc@alllong@true}{}%
2293 \IfIntersect{#2}{.qQ}{\@dc@quiet@true}{}%
2294 \IfIntersect{#2}{iIwW\sphack}%
2295 {% if ‘invisibility’ is specified:
2296 \def\@dc@bsphack@{\@nx\@bsphack}%
2297 \def\@dc@esphack@{\@nx\@esphack}%
2298 }%
2299 {\emptify\@dc@bsphack@\emptify\@dc@esphack@}% if ‘invisibility’ is not
      specified.
2300 \IfAmong\envhack\among{#2}%
2301 {% but if we define an environment:
2302 \emptify\@dc@bsphack@\emptify\@dc@esphack@}%
2303 {% and if we don't define an environment:
2304 \emptify\@dc@env@argstoend}%
2305 \relaxen\@dc@global@\relaxen\@dc@outer@
2306 \IfAmong\outer\among{#2}{\let\@dc@outer@\outer}{}%
2307 \IfAmong\global\among{#2}{\let\@dc@global@\global}{}%

```

We initialise the scratch count and toks registers before parsing of the arguments specifiers.

```

2310 \@dc@catchernum\z@
2311 \@dc@argnum\z@
2312 \toks@{}%
2313 \@temptokena\toks@
2314 \@temptokenb\toks@

```

We parse the arguments specifiers:

```

2316 \@dc#3X% X is the sentinel of parsing.

```

Now we define the basic macro:

```

2318 \@dc@outer@\@dc@global@_ % possibly the prefixes,
2319 \protected\edef#1{% always \protected ;-}
2320 \@dc@bsphack@_ % perhaps \bsphack
2321 \unless\if@dc@quiet@
2322 \unexpanded{%
2323 \IfDCMessages
2324 \typeout}{Parsing arguments for \@dc@bsname.}%
2325 \@nx\fi
2326 \fi
2327 \@nx\@dc@{\the\toks@}% in the braces appear the argument catchers.
2328 \@xa\@nx\csname\dc@bsname\endcsname
2329 \ifx\@dc@outer@\outer
2330 \@xa\@nx\csname_\@xa\gobble\string#1_\endcsname_ % note the
        blank space after #1! If the command is to be \outer, we can't put
        it itself, so we put another, whose name is the same plus a space.
        Anyway, since #1 becomes \protected, this case will never be ex-
        ecuted.
2335 \else_\@nx_#1%
2336 \fi
2337 }%
2338 \edef\gmu@reserveda{%
2339 \@dc@global@_ % perhaps \global
2340 \long\def\@xa\@nx\csname\string#1\endcsname% for
        a command \command, define \command
2342 \the\@temptokena% it's #1#2#3...
2343 {% the definition body:
2344 \@dc@env@argstoend_ % if we define an environment, this macro will
        provide passing the arguments to the end macro (see line 2409).
2349 \unexpanded{#4}%
2350 \@dc@esphack@}%
2351 }% of \edef.
2352 \gmu@reserveda
2353 }% of \DeclareCommand.

```

Now we have a convenient tool to implement parsing of a Knuthian argument(s).

Knuthian argument's default replacement

```

\@dc@K@defaultrepl 2359 \def\@dc@K@defaultrepl{##1}
        2009/11/22 inner pair of braces removed (as leading to additional group causing
        errors).

```

```

\@dc@define@K 2363 \DeclareCommand\@dc@define@K\long{mb}{% first we add the particular CS
        to the toks register:
2365 \edef\gmu@tempa{%
2366 \unexpanded{\toks@\@xa}%
2367 {\@nx\the\toks@{%
2368 \@xa\@nx\csname\dc@bsname_@K\the\@dc@catchernum%
        \endcsname}}}%
2369 }\gmu@tempa

```

and define this macro:

```

2371 \edef\gmu@tempa{%
2372 \def\@xa\@nx\csname\dc@bsname_@K\the\@dc@catchernum%
        \endcsname

```

```

2373     \unexpanded{#1} {%
2374     \@nx \@dc@addargum {%
2375         \IfValueTF{#2}{\unexpanded{#2}}{\@xa {%
                \dc@K@defaultrepl}}%
2376     }}% we add the Knuthian replacement to the toks register as #n.
2378     \@nx \@dc@parse@next \@dc@arguments}% and continue parsing.
2379 }% of \edef. Now we execute this temporary macro, which means we \def \<command>@K<n
2380 \@dc@global@_ % set properly before \@dc#3X.
2381 \if_P \@dc@long@ \relax \long \fi
2382 \gmu@tempa

```

Now we clean up and continue construction of arguments parser.

```

2385 \emptify \@dc@long@
2386 \@dc
2387 }

```

\DeclareEnvironment

```

2390 \DeclareCommand\DeclareEnvironment \long
2391 {m% Pm the environment's name; you may wonder why it allows \par. Once I met
        an environment with a name containing
        % \string\par.
2394 Q{\outer\global\protected\long!lL.qQiIwW\sphack\envhack}% a se-
        quence of  $\epsilon$ -TeX's prefixes and/or specifiers to make all the arguments
        'long' ( \long, !, l or L) and/or not to place the message issuer in the
        command ( ., q, Q) and/or to make the environment ignore spaces fol-
        lowing its end (i, I, w, W or \sphack),
2402 m_ _ _ % the arguments specification (both for the begin and the end definitions),
2404 m_ % the begin definition,
2405 m_ % the end definition; it can contain any #<n>—the arguments of the environ-
        ment are passed to it, too.
2408 } {%

```

\@dc@env@argstoend

```

2409 \def \@dc@env@argstoend {%
2410     \edef
2411     \@xa \@nx \csname_ \bslash#1 (arguments) \endcsname
2412     {\unexpanded {%
2413         \@xa \@xa \@xa \unexpanded \@xa \@xa \@xa {\@xa \@gobble%
                \the \@dc@arguments}%
2414     }}% of outer \unexpanded,
2415     }% of \edef's body.

```

Note that \@dc@env@argstoend will be put in an \edef and the arguments bearer it sets will be defined as custom for this particular environment (its name contains the name of the environment) and inside the environment's group and will bear a list of tokens {<arg.1>} {<arg.2>} ... not the CS \@dc@arguments so it's robust to nested environments and even to \DeclareEnvironment's occurrences inside the environment.

```

2424 }% of \@dc@env@argstoend.
2426 \def \gmu@reserveda {\envhack}% we add the information we define an en-
        vironment.
2428 \IfValueT{#2}%
2429 {\addtomacro \gmu@reserveda {#2}}% we pass all the 'prefixes' to \Decl |
        % areCommand
2432 \@xa \DeclareCommand \csname#1 \@xa \endcsname
2433 \gmu@reserveda {#3} {#4}%

```

Now the begin definition is done. Let's get down to the end definition.

```

2438 \@dc@global@_ % note this CS was for sure redefined by the last \Declare |
      % Command (and by nothing else) and that's exactly what we want.
2441 \@nameedef{end#1}% It's \edef inside.
2442 {\@nx\@xa\@xa\@nx\csname\backslash_ end#1\endcsname
2443 \@xa\@nx\csname_\backslash#1 (arguments)\endcsname}%
      We \edef the \end<env. name> macro to be
      \expandafter\end<env. name>\<env. name>(arguments)
      (it's \expandafter and two control sequences, the latter with (arguments) in its
      name).
2449 \IfIntersect{#2}{iIwW\sphack}%
2450 {\@dc@global@\@xa\addtomacro
2451 \csname_ end#1\endcsname{\@ignoretrue}}{ }%
2452 \edef\gmu@reserveda{%
2453 \@dc@global@\long\def_ % the inner end macro is always \long and
      perhaps defined \globally.
2455 \@xa\@nx\csname\backslash_ end#1\endcsname
2456 \the\@temptokena% it's #1#2#3...—the inner end macro takes the same
      number of parameters as the begin macro.
2458 { % the definition body
2461 \unexpanded{#5}}%
2462 }% of \edef.
2463 \gmu@reserveda
2464 }

```

Ampulex Compressa-like modifications of macros

Ampulex Compressa is a wasp that performs brain surgery on its victim cockroach to lead it to its lair and keep alive for its larva. Well, all we do here with the internal L^AT_EX macros resembles Ampulex's actions but here is a tool for a replacement of part of macro's definition.

The `\ampulexdef` command takes its #2 which has to be a macro and replaces part of its definition delimited with #5 and #6 with the replacement #7. The redefinition may be prefixed with #1. #2 may have parameters and for such a macro you have to set the parameters string and arguments string (the stuff to be taken by the one-step expansion of the macro) as the optional [#3] and [#4]. . If `\ampulexdef` doesn't find the start and end tokens in the meaning of the macro, it does nothing to it. You have to write #### instead of # or you can use `\ampulexhash` as well. For an example use see line 3366.

```

\ampulexdef 2510 \DeclareCommand\ampulexdef\long
2511 {Q{\outer\long\global\protected}_ % (1) (optional) prefix(es); allowed is
      any sequence of them in any order, just like for the original TEX's \def.
2514 S{\def\edef\gdef\xdef\pdef}_ % (2) (optional) kind of definition; if not
      specified, \def will be used.
2516 m_ % (3) macro to be redefined,
2517 O{}_ % (4) \def's parameters string; empty by default,
2518 O{}_ % (5) definition body's parameters to be taken in a one-step expansion of
      the redefined macro; empty by default,
2520 m_ % (6) start token(s),
2521 m_ % (7) end token(s)
2522 m_ % (8) the replacement

```

```

2523 } {% For the example of usage see 3366.
2530 \def \gmu@tempa {#6}%
2531 \def \gmu@tempb {#7}%
2532 \def \gmu@tempc {#8}% we wrap the start, end and replacement tokens in
      macros to avoid unbalanced \ifs.
2534 \edef \gmu@tempd {%
2535   \long\def \@nx \gmu@tempd
2536   #####1 \detoken@xa \gmu@tempa
2537   #####2 \detoken@xa \gmu@tempb
2538   #####3 \@nx \gmu@tempd {%
2539     \unexpanded {%
2541       \@ifempty {##3} {\hidden@iffalse} {\hidden@iftrue}}}%
2543 \gmu@tempd% it defines \gmu@tempc to produce an open \if depending on
      whether the start and end token(s) are found in the meaning of #3.

2547 \edef \gmu@tempe {%
2548   \@nx \gmu@tempd \all@other#3% note that #3 may have parameters so we
      have to look at its meaning not at its one-level expansion, which could
      produce an 'extra \}' error.
2551   \detoken@xa \gmu@tempa
2552   \detoken@xa \gmu@tempb \@nx \gmu@tempd
2553 }%
2555 \gmu@tempe% we apply the checker and it produces an open \if.
2557 \edef \gmu@tempd {%
2558   \long\def \@nx \gmu@tempd
2559   #####1 \@xa \unexpanded \@xa {\gmu@tempa}%
2560   #####2 \@xa \unexpanded \@xa {\gmu@tempb}%
2561   #####3 \@nx \ampulexdef {% we define a temporary macro with the param-
      eters delimited with the 'start' and 'end' parameters of \ampulexdef.

2564     \@nx \unexpanded {#####1}%
2565     \@nx \@xa \@nx \unexpanded
2566     \@nx \@xa {\@nx \gmu@tempc}% we replace the part of the redefined macro's
      meaning with the replacement text.
2568     \@nx \unexpanded {#####3}}}%
2570 \gmu@tempd
2573 \edef \gmu@tempf {#5}%
2574 \edef \gmu@tempe {%
2575   \IfValueT {#1} {#1} \IfValueTF {#2} {#2} {\def}%
2576   \@nx#3#4 {%
2577     \@xa \@xa \@xa \gmu@tempd \@xa#3 \gmu@tempf \ampulexdef}}%
2578 \gmu@tempe
2579 \fi}

```

`\ampulexhash` 2581 `\def \ampulexhash {####}%` for your convenience (not to count the hashes).

For the heavy debugs I was doing while preparing `gmdoc`, as a last resort I used `\showlists`. But this command alone was usually too little: usually it needed setting `\showboxdepth` and `\showboxbreadth` to some positive values. So,

```

\gmshowlists 2588 \def \gmshowlists {\showboxdepth=1000 \showboxbreadth=1000 \showlists}

```

```

\namesthe 2593 \newcommand \namesthe [1] {\@xa \show \csname#1 \endcsname}
\namesthe 2594 \newcommand \namesthe [1] {\@xa \showthe \csname#1 \endcsname}

```

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

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

```
\xiistring 2604 \long\def\xiistring#1{%
2605   \if\@nx#1\xiispace
2606     \xiispace
2607   \else
2608     \afterfi{\string#1}% to make the same error as bare \string would
           cause in case of empty #1.
2610   \fi}
```

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

```
\@xiispaces 2614 \def\@xiispaces#1_#2\@@nil{%
2615   #1%
2616   \ifx\@xiispaces#2\@xiispaces
2617   \else
2618   \xiispace
2619   \afterfi{\@xiispaces#2\@@nil}%
2620   \fi}
```

Environments redefined

Almost an environment or redefinition of `\begin`

We’ll extend the functionality of `\begin`: the non-starred instances shall act as usual and we’ll add the starred version. The difference of the latter will be that it won’t check whether the ‘environment’ has been defined so any name will be allowed.

This is intended to structure the source with named groups that don’t have to be especially defined and probably don’t take any particular action except the scoping.

(If the `\begin*`’s argument is a (defined) environment’s name, `\begin*` will act just like `\begin`.)

Original L^AT_EX’s `\begin`:

```
\def\begin#1{%
  \@ifundefined{#1}%
    {\def\reserved@a{\@latex@error{Environment #1
  undefined}\@eha}}%
    {\def\reserved@a{\def\@currenvir{#1}%
      \edef\@currenvline{\on@line}%
      \csname #1\endcsname}}%
  \@ignorefalse
  \begingroup\@endpefalse\reserved@a}
```

```
\@begnamedgroup 2652 \long\def\@begnamedgroup#1{%
2653   \edef\@prevgrouplevel{\the\currentgrouplevel}% added 2009/03/24
           to handle special pseudo-environments that don’t increase \current!
           grouplevel(such as document). Note it’s \edefed outside the environ-
           ment’s group.
2657   \@ignorefalse% not to ignore blanks after group
2658   \begingroup\@endpefalse
```

```

2659 \edef\@prevenvir{\@currentenv}% Note we \edef it inside the group (for
      obvious reason), unlike the 'previous' grouplevel.
2661 \edef\@currentenv{#1}% We could do recatcoding through \string or
      % \detokenize but all the name 'other' and 10 could affect a thousand
      packages so we don't do that and we'll recatcode in a testing macro, see line
      2748.
2666 \edef\@currentvline{\on@line}%
2667 \csname_#1\endcsname}% if the argument is a command's name (an environ-
      ment's e.g.), this command will now be executed. (If the corresponding
      control sequence hasn't been known to TEX, this line will act as \relax.)

```

Let us make it the starred version of \begin.

```

\begin* 2676 \def\begin{\gm@ifstar{\@begnamedgroup}}{%
\begin 2677   \@begnamedgroup@ifcs}
\@begnamedgroup@ifcs 2680 \def\@begnamedgroup@ifcs#1{%
2681   \ifcsname#1\endcsname\afterfi{\@begnamedgroup{#1}}%
2682   \else\afterfi{\@latex@error{Environment_#1_undefined}}%
      \@eha}%
2683   \fi}%

```

\@ifenvir and improvement of \end

It's very clever and useful that \end checks whether its argument is \ifx-equivalent \@currentenv. However, in standard L^AT_EX it works not quite as I would expect: Since the idea of environment is to open a group and launch the CS named in the \begin's argument. That last thing is done with \csname...\endcsname so the catcodes of chars are irrelevant (until they are \active, _{1,2} etc.). Thus should be also in the \end's test and therefore we ensure the compared texts are both expanded and made all 'other'.

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

```

\@ifenvir 2707 \long\def\@ifenvir#1{%
      % #1 enquired environment name which will be confronted with \@cur|
      renvir
      % #2 what if true (if the names are equivalent3)
      % #3 what if false
2719 \@ifdetokens{\@currentenv}{#1}}
\@ifdetokens 2722 \long\pdef\@ifdetokens#1#2{%
      % #1 first list of tokens to be expanded and detokenized
      % #2 second list
      % #3 if agree
      % #4 else
2736   \edef\gmu@tempa{#1}% to get #1 fully expanded.
2737   \edef\gmu@reserveda{\@xa\detokenize\@xa{\gmu@tempa}}% with our
      brave new \begin, \@currentenv is fully expanded, remember?
2740   \edef\gmu@tempa{#2}% to get #2 fully expanded.
2741   \edef\gmu@reservedb{\@xa\detokenize\@xa{\gmu@tempa}}%
2742   \ifx\gmu@reserveda\gmu@reservedb\@xa\@firstoftwo

```

³ The names are checked whether they produce the same \csname. They don't have to have the same catcodes.

```

2743 \else \@xa \@secondoftwo
2744 \fi}
\ifjobname 2746 \def \@ifjobname#1 {\@ifedetokens {\jobname} {#1}}
\ifprevenvir 2748 \long \def \@ifprevenvir#1 {%
    % #1 enquired environment name which will be confronted with \@pre!
    % #2 what if true (if the names are equivalent4)
    % #3 what if false
2760 \@ifedetokens {\@prevenvir} {#1}}

```

Note that \@ifjobname and \@ifenvir are expandable and in an \edef they expand to

```
\@ifedetokens {<current jobname>} {<arg.1>}
```

and

```
\@ifedetokens {<current envir>} {<arg.1>}
```

resp. which may be useful for some T_EXvert.

```
\@checkend 2771 \def \@checkend#1 {\@ifenvir {#1} {} {\@badend {#1}}}
```

Thanks to it you may write \begin {macrocode*} with *₁₂ and end it with \end {% macrocode*} with *₁₁ (that was the problem that led me to this solution). The error messages looked really funny:

```
! LaTeX Error: \begin{macrocode*} on input line 1844 ended
by \end{macrocode*}.
```

You might also write also \end {macrocode \star} where \star is defined as ‘other’ star or letter star.

Storing and restoring the catcodes of specials

```

\gmu@storespecials 2787 \DeclareCommand \gmu@storespecials {o} {% we provide a possibility of adding
    stuff. For usage see line ??
2789 \def \do##1 {\catcode ` \@nx##1=\the \catcode ` ##1 \relax}%
\gmu@restorespecials 2790 \edef \gmu@restorespecials {%
2791 \dospecials \do ^^M} \IfValueT {#1} {#1}}
\gmu@septify 2793 \pdef \gmu@septify {% restoring the standard catcodes of specials. The name is
    the opposite of ‘sanitize’ :-). It restores also the original catcode of ^^M.
2796 \def \do {\relax \catcode `}%
2797 \do \_10 \do \_0 \do \_1 \do \_2 \do \_3 \do \_4%
2798 \do \_6 \do \_7 \do \_8 \do \_14 \do \_13 \do \_5 \relax
    %% \let \do \@makeother
    %% \do \_0 \do \_1 \do \_2 \do \_3 \do \_4 \do \_5 \do \_6 \do \_7 \do \_8 \do \_9 \relax
2801 }

```

⁴ The names are checked whether they produce the same \csname. They don’t have to have the same catcodes.

From relsize

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

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

```

\relsize      You declare the font size with \relsize {<n>} where <n> gives the number of steps
               ("mag-step" = factor of 1.2) to change the size by. E.g., n = 3 changes from \normal|
\smaller     size to \LARGE size. Negative n selects smaller fonts. \smaller == \relsize{%
\larger      -1}; \larger == \relsize{1}. \smallerr(my addition) == \relsize{-2};
\smallerr    \largerr guess 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 2831 \pdef\relsize#1{%
2832   \ifmmode\@nomath\relsize\else
2833     \begingroup
2834       \@tempcnta\% assign number representing current font size
2835         \ifx\@currsize\normalsize\4\else\% funny order is to have
               most ...
2836         \ifx\@currsize\small\3\else\% ...likely sizes checked
               first
2837         \ifx\@currsize\footnotesize\2\else
2838         \ifx\@currsize\large\5\else
2839         \ifx\@currsize\Large\6\else
2840         \ifx\@currsize\LARGE\7\else
2841         \ifx\@currsize\scriptsize\1\else
2842         \ifx\@currsize\tiny\0\else
2843         \ifx\@currsize\huge\8\else
2844         \ifx\@currsize\Huge\9\else
2845         4\rs@unknown@warning\% unknown state: \normal|
               size as starting point
2846     \fi\fi\fi\fi\fi\fi\fi\fi\fi\fi

               Change the number by the given increment:
2848     \advance\@tempcnta#1\relax

               watch out for size underflow:
2850     \ifnum\@tempcnta<\z@\rs@size@warning{small}{\string%
               \tiny}\@tempcnta\z@\fi
2851     \@xa\endgroup
2852     \ifcase\@tempcnta\% set new size based on altered number
2853     \tiny\or\scriptsize\or\footnotesize\or\
               \small\or\normalsize\or
2854     \large\or\Large\or\LARGE\or\huge\or\Huge\
               \else
2855     \rs@size@warning{large}{\string\Huge}\Huge
2856 \fi\fi}\% end of \relsize.

\rs@size@warning 2858 \providecommand*\rs@size@warning[2]{\PackageWarning{gmutils\
               (relsize)}{\%
2859 Size\requested\is\too\#1.\MessageBreak\Using\#2\instead}}

\rs@unknown@warning 2862 \providecommand*\rs@unknown@warning{\PackageWarning{gmutils\

```

```

                (relsize)}{Current_font_size
2863 is_unknown! (Why?!?) \MessageBreak Assuming \string%
                \normalsize}}

```

And a handful of shorthands:

```

\larger 2867 \DeclareRobustCommand*\larger[1][\@ne]{\relsize{+#1}}
\smaller 2868 \DeclareRobustCommand*\smaller[1][\@ne]{\relsize{-#1}}
\textlarger 2869 \DeclareRobustCommand*\textlarger[2][\@ne]{\relsize{+#1}#2}}
\textsmaller 2870 \DeclareRobustCommand*\textsmaller[2][\@ne]{\relsize{-#1}#2}}
\largerr 2871 \pdef\largerr{\relsize{+2}}
\smallerr 2872 \pdef\smallerr{\relsize{-2}}

```

Meta-symbols

I fancy also another Knuthian trick for typesetting *<meta-symbols>* in *The T_EXbook*. So I repeat it here. The inner `\meta` macro is copied verbatim from doc's v2.1b documentation dated 2004/02/09 because it's so beautifully crafted I couldn't resist. I only don't make it `\long`.

"The new implementation fixes this problem by defining `\meta` in a radically different way: we prevent hyphenation by defining a `\language` which has no patterns associated with it and use this to typeset the words within the angle brackets."

```
\meta 2894 \pdef\meta#1{%
```

"Since the old implementation of `\meta` could be used in math we better ensure that this is possible with the new one as well. So we use `\ensuremath` around `\langle` and `\rangle`. However this is not enough: if `\meta@font@select` below expands to `\itshape` it will fail if used in math mode. For this reason we hide the whole thing inside an `\nfss@text` box in that case."

```

2902   {\meta@fontsetting\ensuremath\langle}%
2903   \ifmmode\@xa\nfss@text\fi
2904   {% this has to be a begin-group because \nfss@text becomes \hbox in math
        mode.
2906   \gmu@activespaceblank
2907   \meta@font@select

```

Need to keep track of what we changed just in case the user changes font inside the argument so we store the font explicitly.

```

2915     #1 \/%
2917   }%
2918   {\meta@fontsetting\ensuremath\rangle}%
2919 }% of \meta.

```

```

\gmu@activespaceblank 2921 \pdef\gmu@activespaceblank{%
\gmu@activespace 2922   \@xa\def\gmu@activespace{\space\ignorespaces}% note the subtle per-
                    versity 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.
2929 }
\meta@fontsetting 2931 \def\meta@fontsetting{\color{red!50!black}}
\meta@font@select 2933 \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`⁵ is a version of *The T_EXbook's* one.

```
\<...> 2945 \def\<#1>{\meta{#1}}
\metachar 2947 \pdef\metachar#1{\begingroup\metacharfont\char#1\endgroup}
\metacharfont 2948 \def\metacharfont{\meta@fontsetting\rm}
```

Macros for printing macros and filenames

First let's define three auxiliary macros analogous to `\dywiz` from `polski.sty`: a 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_EX primitive has three arguments: #1 'before break', #2 'after break', #3 'without break', remember?

```
\discre 2961 \pdef\discre#1#2#3{\leavevmode\kernosp%
2962 \discretionary{#1}{#2}{#3}\penalty10000\hskiposp\relax}
\discret 2964 \pdef\discret#1{\discre{#1}{#1}{#1}}
```

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

```
2968 \def\:{%
2969 \ifmmode\afterfi{\mskip\medmuskip}%
2970 \else\afterfi{\discre{\null}{}}}% \null to get \hyphenpenalty
not \exhyphenpenalty.
2972 \fi}
2976 \let\gmu@discretionaryhyphen\-\char"2D% for the cases when we don't redefine \
but use \
```

```
\bihyphen 2980 \DeclareCommand\bihyphen{O{*}}{% a redefinition of \- that makes it optional-
argument to allow further hyphenation
\gmu@discretionaryhyphen 2982 \DeclareCommand\gmu@discretionaryhyphen{T{#1}ca}{%
2983 \IfValueT{##2}{%
2984 \@ifempty{##2}{}{%
\gmu@bihyphen@char 2985 \def\gmu@bihyphen@char{##2}}}%
2986 }%
2987 \IfValueT{##3}{%
2988 \@ifempty{##3}{}{%
\gmu@bihyphen@corr 2989 \def\gmu@bihyphen@corr{##3}}}%
2990 }%
2991 \IfValueTF{##1}\discre\discretionary
2992 {% before break
2993 \IfValueTF{##2}{%
2994 \@ifempty{##2}{\gmu@bihyphen@char}{##2}}%
2995 }{%
2996 \ifnum\hyphenchar\font>\z@
2997 \char\hyphenchar\font
2998 \fi}}% end of before break
```

⁵ Think of the drags that transform a very nice but rather standard 'auntie' ('Tante' in Deutsch) into a most adorable Queen ;-).

```

2999     {% after break
3000     \IfValueT{##3}{%
3001     \@ifempty{##3}{\gmu@bihyphen@corr}{##3}%
3002     }%
3003     }%
3004     {% without break
3005     }% almost as in The TEXbook: unlike The TEXbook, we allow hyphenchars ≥ 255
        as we are XYTEX.
3007 }% of \DeclareCommand\–
3008 \gmu@storeifnotyet \–% original \– is a TEX's primitive, therefore we should
        store it.
3010 \let \– \gmu@discretionaryhyphen
3011 \let \@dischiph \gmu@discretionaryhyphen_□% to override framed.sty
3012 }% of \bihyphen

```

```

3015 \relaxen \gmu@bihyphen@corr

```

```

\vs 3017 \pdef \vs {\discre {\visiblespace} {} {\visiblespace}}

```

Then we define a macro that makes the spaces visible even if used in an argument (i.e., in a situation where `re\catcodeing` has no effect).

```

\printspaces 3023 \def \printspaces#1 {\let ~ = \vs_□ \let \_ = \vs_□ \gm@pswords#1_□%
        \@@nil}}

```

```

\gm@pswords 3025 \def \gm@pswords#1_□#2 \@@nil {%
3026     \ifx \relax#1 \relax \else#1 \fi
3027     \ifx \relax#2 \relax \else \vs \penalty \hyphenpenalty%
        \gm@pswords#2 \@@nil \fi}% note that in the recursive call
        of \gm@pswords the argument string is not extended with a sentinel
        space: it has been already by \printspaces.

```

```

\sfname 3032 \pdef \sfname#1 {\textsf {\printspaces {#1}}}

```

```

\gm@discretionaryslash 3034 \def \gm@discretionaryslash {\discre {/} {\hbox {/}} {/}}% the
        second pseudo-argument nonempty to get \hyphenpenalty
        not \exhyphenpenalty.

```

```

\file 3039 \pdef \file#1 {\gmu@printslashes#1/\gmu@printslashes}

```

```

\gm@printslashes 3041 \def \gm@printslashes#1/#2 \gm@printslashes {%
3042     \sfname {#1}%
3043     \ifx \gm@printslashes#2 \gm@printslashes
3044     \else
3045     \textsf {\gm@discretionaryslash}%
3046     \afterfi {\gm@printslashes#2 \gm@printslashes} \fi}

```

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

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

```

\pk 3053 \pdef \pk#1 {\textsf {#1}}

```

Some (if not all) of the below macros are copied from `doc` and/or `ltxdoc`.

A macro for printing control sequences in arguments of a macro. Robust to avoid writing an explicit `\` into a file. It calls `\ttfamily` not `\tt` to be usable in headings which are boldface sometimes.

```

\cs 3067 \DeclareCommand \cs {O {\type@bslash \penalty \@M \hskip \z@skip}} {%
        % [#1] O {\bslash} the control sequence's prefix, by default it's \ allowing
        hyphenation of subsequent word,

```

```

        % #2 m the control sequence or anything to be typeset in typewriter font.
3080 \begingroup
3081 \ifdefined\verbatim@specials\verbatim@specials\fi
3082 \edef\-\{\discretionary{%
3083     \ifdefined\gmv@hyphen\gmv@hyphen
3084     \else\unexpanded{\normalfont-}}%
3085     \fi}{}{}}%
3086 \def\{{\type@lbrace\yeshy}\def\}{\char`\\}}%
3087 \narrativett
3088 \edef\narrativett@storedhyphenchar{\the\hyphenchar\font}%
3089 \hyphenchar\font=\ifdefined\gmv@hyphenchar\gmv@hyphenchar
3090 \else␣"A6␣\fi
3091 \cs@inner{#1}}

\cs@inner 3093 \pdef\cs@inner#1#2{%
3094 #1#2%
3095 \hyphenchar\font=\narrativett@storedhyphenchar\relax
3096 \endgroup}

\narrativett 3098 \def\narrativett{\ttfamily}% such name because I introduce it to distin-
        guish the narrative verbatims from the code in gmdoc.

\env 3101 \long\pdef\env{\cs[]}

        And for the special sequences like ^^A:
3105 \foone{\@makeother\^}
\hathat 3106 {\pdef\hathat{\cs[^^]}}

3108 \AtBeginDocument{%
\hash 3109 \@ifpackageloaded{gmdoc}{\def\hash{\cs[#]}}{}}

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

\possfil 3112 \def\possfil{\hfil\penalty1000\hfilneg}

\possvfil 3114 \def\possvfil{\vfil
3115 \penalty\numexpr
3116 \gmu@minnum{\clubpenalty+\widowpenalty}{9999}%
3126 \relax␣% eaten by \gmu@maxnum
3127 \relax␣% eaten by \numexpr
3128 \vfilneg}

\gmu@extremnum 3130 \long\def\gmu@extremnum#1#2#3{%
        % #1 inequality sign
        % #2 left side of comparison
        % #3 right side of comparison
3136 \@xa\ifx\@firstofmany#3\@@nil\relax␣% this complicated test is to al-
        low arguments beginning with some \if<...> as in \possvfil e.g.
3139 \@xa\@firstoftwo
3140 \else\@xa\@secondoftwo
3141 \fi
3142 {#2}%
3143 {%
3144 \ifnum\numexpr#2\relax#1\numexpr#3\relax
3145 \@xa\@firstoftwo
3146 \else\@xa\@secondoftwo
3147 \fi

```

```

3148     {\gmu@extremnum#1 {#2}}%
3149     {\gmu@extremnum#1 {#3}}%
3150     }%
3151 }
\gmu@maxnum 3153 \def\gmu@maxnum{\gmu@extremnum>}
\gmu@minnum 3154 \def\gmu@minnum{\gmu@extremnum<}

```

Typesetting arguments and commands

`\arg` We define a conditional and iterating command `\arg` that in math mode does what it used to do was in math and outside math it typesets mandatory, optional and picture (parenthesed) and angled arguments and optional stars. You can write

```
\arg[gefülte]*<fisch>(mit){baigele}
```

to get

```
[<gefülte>][*]{<fisz>}(<mit>){<bajgele>}
```

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`)."

```
\cmd 3182 \long\def\cmd#1{\@xa\cs\@xa{\@xa\cmd@to@cs\string#1}}% it has to
      be un \protected! It has so many \expandafters to allow \cmd\par and
      still keep the \cs command 'short'.
```

```
\cmd@to@cs 3186 \def\cmd@to@cs#1#2{\char\number`#2\relax}
```

It can be short since it never gets actual control sequence as an argument only a string of 'other' tokens (and maybe spaces).

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

```
\marg 3192 \pdef\marg#1{\narrativett\type@lbrace}\meta{#1}{%
      \narrativett\char`\\}}
```

`\oarg{text}` prints `[<text>]`, 'optional argument'. Also `\oarg[text]` does that.

```
\oarg 3198 \pdef\oarg{\@ifnextchar[\@oargsq\@oarg}
3200 \pdef\@oarg#1{\narrativett[]\meta{#1}{\narrativett}}
3201 \pdef\@oargsq[#1]{\@oarg{#1}}
```

```
\parg{te,xt} prints (<te,xt>), 'picture mode argument'.
```

```
\parg 3205 \pdef\parg{\@ifnextchar(\@pargp\@parg}
3207 \def\@parg#1{\narrativett()\meta{#1}{\narrativett}}
```

```

3208 \def\@pargp(#1){\@parg{#1}}
3210 \pdef\@arg{\@ifnextchar<\@arga\@arg}
3211 \def\@arg#1{\narrativett<\meta{#1}\narrativett>}
3212 \def\@arga<#1>{\@arg{#1}}
3214 \def\@verbaarga#1#2>{\@arg{#2}\arg@dc}
3216 \foone{\catcode`>\active}{%
3217   \def\@verbaargact#1#2>{\@arg{#2}\arg@dc}%
3218 }
3220 \foone{\@makeother\{\@makeother\}}%
3221   \catcode`[=\@ne\catcode`]=\tw@}
3222 [%
3223   \def\@verbmargm#1#2][% for an argument in curly braces in a verbatim,
      where the braces are not groupers and not necessary ‘other’. We’ll know
      by \@ifnextif that the future token is an opening brace. Note this macro
      has 2nd parameter delimited with ‘other’ closing brace (so may not act
      correctly when braces are nested (then hide them with special verbatim
      groupers)).
3229     \marg[#2]%
3230     \arg@dc
3231   ]%
3232 ]% of \foone
\arg@dc 3235 \DeclareCommand\arg@dc!{%
3238   s□% (1)
3239   o□% (2)
3240   c□% (3)
3241   b□% (4)
3242   a□% (5)
3243   S{\arg}□% (6) just gobbled (for backwards compatibility)
3244 }{% This command iterates while it has arguments and typesets them in brackets,
      parentheses or curly braces. Note it gobbles subsequent \args and just iterates.
3247   \def\next{0}%
3248   \IfValueT{#1}%
3249   {\metachar[\scanverb{*}\metachar]\def\next{1}}%
3250   \IfValueT{#2}{\@oarg{#2}\def\next{1}}%
3251   \IfValueT{#3}{\@parg{#3}\def\next{1}}%
3252   \IfValueT{#4}{\marg{#4}\def\next{1}}%
3253   \IfValueT{#5}{\@arg{#5}\def\next{1}}%
3254   \@ifnextchar\egroup{\endgroup}{%
3255     \if1\next\@xa\arg@dc
3256     \else□% it’s crucial that we look for verbatim braces after we checked there
      were no #4, otherwise there would be an error.
3259     \def\next{%
3260       \@ifnextif\xiilbrace{\@verbmargm}%
3261       {% not active or other lbrace
3262         \@ifnextif<{% then we look for angles
3263           \ifnum\catcode`>=\active
3264             \@xa\@verbaargact
3265             \else\@xa\@verbaarga
3266             \fi}%

```

```

3267         {% and if not angles neither verbatim braces, then
3268         \endgroup□□% if we have no more arguments to typeset, we
                close the group opened in line 3286
3270         }%
3271     }%
3272 }%
3273 \@xa□\next
3274 \fi
3275 }% of not egroup
3276 }% of \arg@dc

```

Now define the front-end macro of the `\arg` command:

```

3280 \foone{\obeylines}{%
3281 \AtBeginDocument{%
3282 \let\math@arg\arg□%
\arg 3283 \pdef\arg{\ifmmode\math@arg□%
3285 \else\afterfi{%
3286 \begingroup□%
3288 \ifdefined\@ifQueerEOL\@ifQueerEOL{%
3289 \def^^M{\unskip\space}% in the 'queer' EOLs scope we keep
                line end active in case we have \arg {<arg.>} ending a line: the
                next char peeper touches line end or, if the line end was 5, gob-
                bles the space it turns into so the comment layer would 'leak'
                to the code layer.
3295     }}\fi□%
3296 \arg@dc}%
3297 \fi}% of \arg,
3298 }% of \AtBeginDocument,
3299 }% of \foone.

```

Now you can write

```

\arg{mand.\_arg}□[opt.\_arg]□(pict.\_arg)
to get {<mand. arg>} [ <opt. arg>] ( <pict. arg>). (Yes, with only one \arg!)
And $\arg(1+i)□=□\pi/4$ for  $\arg(1+i) = \pi/4$ .

```

```

\cat 3311 \DeclareCommand\cat{Q{"0123456789ABCDEF}}{%
3312 ${}_{\the\numexpr#1}\m@th$\@ifnextcat□a\space{}}

```

Not only preamble!

Let's remove some commands from the list to erase at begin document! Primarily that list was intended to save memory not to forbid anything. Nowadays, when memory is cheap, the list of only-preamble commands should be rethought IMHO.

```

\not@onlypreamble 3328 \newcommand\not@onlypreamble[1]{%
3329 \def\do##1{\ifx#1##1\else\@nx\do\@nx##1\fi}%
3330 \xdef\@preamblecmds{\@preamblecmds}}
3332 \not@onlypreamble\@preamblecmds
3333 \not@onlypreamble\@ifpackageloaded
3334 \not@onlypreamble\@ifclassloaded
3335 \not@onlypreamble\@ifl@aded
3336 \not@onlypreamble\@pkgextension

```


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

```
\gm@notprerr 3341 \def\gm@notprerr {\can be used only in preamble (\on@line) }
3343 \AtBeginDocument {%
3344   \def\do#1 {\@nx\do\@nx#1}%
3345   \edef\@preamblecmds {%
3346     \def\@nx\do##1 {%
3347       \def##1 {\@nx\PackageError {gmutils/LaTeX}%
3348         {\@nx\string##1 \@nx\gm@notprerr} \@nx\@eha}}%
3349   \@preamblecmds }
```

A subtle error raises: the L^AT_EX standard \onlypreamble and what \document does with \@preamblecmds makes any two of 'only preamble' CS's \ifx-identical inside document. And my change makes any two CS's \ifx-different. The first it causes a problem with is standard L^AT_EX's \nocite that checks \ifx\onlypreamble\document. So hoping this is a rare problem, we circumvent it. 2008/08/29 a bug is reported by Edd Barrett that with natbib an 'extra' error occurs so we wrap the fix in a conditional.

```
\gmu@nocite@ampulex 3366 \def\gmu@nocite@ampulex {% we wrap the stuff in a macro to hide an open \if.
                       And not to make the begin-input hook too large. the first optional argument
                       is the parameters string and the second the argument for one-level expansion
                       of \nocite. Both hash strings are doubled to pass the first \def.
3372   \ampulexdef\nocite [###1] [ { {###1} } ] % note the double brace around
                       % #3.
3374   \ifx
3375     {\@onlypreamble\document}%
3376     \iftrue}
3379 \AtBeginDocument {\gmu@nocite@ampulex}%
```

Third person pronouns

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

Previous versions of this documentation were consequently alternating 'he' and 'she' and provided specific macros for that purpose. Now I'm not that queer and gender so I take what is normally used, 'they' that is.

(The issue of human sexes and genders (certainly much more numerous than 2) is complex and delicate and a T_EX macro package is probably not the best place to discuss it.)

```
\heshe 3428 \def\heshe {they}
\hisher 3429 \def\hisher {their}
\himher 3430 \def\himher {them}
\hishers 3431 \def\hishers {theirs}

\HeShe 3433 \def\HeShe {They}
\HisHer 3434 \def\HisHer {Their}
\HimHer 3435 \def\HimHer {Them}
\HisHers 3436 \def\HisHers {Theirs}
```

Improvements to mwcls sectioning commands

That is, ‘Expe-ri-mente’⁶ mit MW sectioning & `\refstepcounter` to improve mwcls’s cooperation with hyperref. They shouldn’t make any harm if another class (non-mwcls) is loaded.

We `\refstep` sectioning counters even if the sectionings are not numbered, because otherwise

1. pdfTeX cried of multiply defined `\labels`,
2. e.g. in a table of contents the hyperlink `<rozdzia\l\Kwiaty\polskie>` linked not to the chapter’s heading but to the last-before-it change of `\ref`.

```

3454 \AtBeginDocument {% because we don't know when exactly hyperref is loaded and
      maybe after this package.
NoNumSecs 3456   \ifpackageloaded{hyperref} {\newcounter{NoNumSecs}%
3457     \setcounter{NoNumSecs}{617}% to make \refing to an unnumbered
      section visible (and funny?).
gm@hyperrefstepcounter 3459   \def\gm@hyperrefstepcounter {\refstepcounter{NoNumSecs}}%
\gm@targetheading 3460   \pdef\gm@targetheading#1 {%
3461     \hypertarget{#1}{#1}}% end of then
gm@hyperrefstepcounter 3462   {\def\gm@hyperrefstepcounter {}%
\gm@targetheading 3463   \def\gm@targetheading#1{#1}}% end of else
3464 }% of \AtBeginDocument

```

Auxiliary macros for the kernel sectioning macro:

```

sectionsoutofmainmatter 3467 \def\gm@dontnumbersectionsoutofmainmatter {%
3468   \if@mainmatter\else\HeadingNumberedfalse\fi
earpagesduetoopenright 3469 \def\gm@clearpagesduetoopenright {%
3470   \if@openright\cleardoublepage\else\clearpage\fi

```

To avoid `\defing` of `\mw@sectionxx` if it’s undefined, we redefine `\def` to gobble the definition and restore the original meaning of itself.

Why shouldn’t we change the ontological status of `\mw@sectionxx` (not define if undefined)? Because some macros (in `gmdocc` e.g.) check it to learn whether they are in an mwcls or not.

But let’s make a shorthand for this test since we’ll use it three times in this package and maybe also somewhere else.

```

\@ifnotmw 3483 \long\def\@ifnotmw#1#2 {\gm@ifundefined{mw@sectionxx}{#1}{#2}}

```

The kernel of MW’s sectioning commands:

```

3508 \@ifnotmw {} {%
\mw@sectionxx 3509 \def\mw@sectionxx#1#2[#3]#4 {%
3510   \edef\mw@HeadingLevel {\csname #1@level\endcsname
3511     \space}% space delimits level number!
3512   \ifHeadingNumbered
3513     \ifnum #1 > \mw@HeadingLevel > \c@secnumdepth
3514       \HeadingNumberedfalse\fi
      line below is in \gm@ifundefined to make it work in classes other than mwbc
3516     \gm@ifundefined{if@mainmatter} {} {%
3517       \gm@dontnumbersectionsoutofmainmatter}
3517   \fi
      % \ifHeadingNumbered

```

⁶ A. Berg, *Wozzeck*.

```

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

\HeadingRHeadText 3526 \def\HeadingRHeadText{#2}%
\HeadingTOCText   3527 \def\HeadingTOCText{#3}%
\HeadingText      3528 \def\HeadingText{#4}%
\mw@HeadingType   3529 \def\mw@HeadingType{#1}%
3530 \if\mw@HeadingBreakBefore
3531 \if@specialpage\else\thispagestyle{closing}\fi
3532 \gm@ifundefined{if@openright}{\}%
\gm@clearpagesduetoopenright}%
3533 \if\mw@HeadingBreakAfter
3534 \thispagestyle{blank}\else
3535 \thispagestyle{opening}\fi
3536 \global\@topnum\z@
3537 \fi% of \if\mw@HeadingBreakBefore

placement of \refstep suggested by me (GM):
3540 \ifHeadingNumbered
3541 \refstepcounter{#1}%
3542 \protected@edef\HeadingNumber{\csname\the#1\endcsname%
\relax}%

3543 \else
3544 \let\HeadingNumber\@empty
3546 \fi% of \ifHeadingNumbered

3548 \if\mw@HeadingRunIn
3549 \mw@runinheading
3550 \else
3551 \if\mw@HeadingWholeWidth
3552 \if@twocolumn
3553 \if\mw@HeadingBreakAfter
3554 \onecolumn
3555 \mw@normalheading
3556 \pagebreak\relax
3557 \if@twoside
3558 \null
3559 \thispagestyle{blank}%
3560 \newpage
3561 \fi% of \if@twoside
3562 \twocolumn
3563 \else
3564 \@topnewpage[\mw@normalheading]%
3565 \fi% of \if\mw@HeadingBreakAfter
3566 \else
3567 \mw@normalheading
3568 \if\mw@HeadingBreakAfter\pagebreak\relax\fi
3569 \fi% of \if@twocolumn
3570 \else
3571 \mw@normalheading

```

```

3572     \if\mw@HeadingBreakAfter\pagebreak\relax\fi
3573     \fi% of \if\mw@HeadingWholeWidth
3574 \fi% of \if\mw@HeadingRunIn
3575 }

```

An improvement of MW's \SetSectionFormatting

A version of MW's \SetSectionFormatting that lets to leave some settings unchanged by leaving the respective argument empty ({} or []).

Notice: If we adjust this command for new version of MWCLS, we should name it \SetSectionFormatting and add issuing errors if the inner macros are undefined.

```

[#1] the flags, e.g. breakbefore, breakafter;
#2  the sectioning name, e.g. chapter, part;
#3  preskip;
#4  heading type;
#5  postskip

```

```

\SetSectionFormatting 3599 \relaxen\SetSectionFormatting
3600 \newcommand*\SetSectionFormatting[5][\empty]{%
3601   \ifx\empty#1\relax\else% empty (not \empty!) #1 also launches \else.
3602   \def\mw@HeadingRunIn{10}\def\mw@HeadingBreakBefore{10}%
3603   \def\mw@HeadingBreakAfter{10}\def\mw@HeadingWholeWidth{%
3604     10}%
3605   \@ifempty{#1}{}{\mw@processflags#1,\relax}% If #1 is omitted, the
3606     flags are left unchanged. If #1 is given, even as [], the flags are first
3607     cleared and then processed again.
3608   \fi
3609   \gm@ifundefined{#2}{\@namedef{#2}{\mw@section{#2}}}{}%
3610   \mw@secdef{#2}{@preskip}{#3}{2\oblig.}%
3611   \mw@secdef{#2}{@head}{#4}{3\oblig.}%
3612   \mw@secdef{#2}{@postskip}{#5}{4\oblig.}%
3613   \ifx\empty#1\relax
3614     \mw@secundef{#2@flags}{1\optional)}%
3615   \else\mw@setflags{#2}%
3616   \fi}
\mw@secdef 3617 \def\mw@secdef#1#2#3#4{%
3618   % #1 the heading name,
3619   % #2 the command distincter,
3620   % #3 the meaning,
3621   % #4 the number of argument to error message.
3622   \@ifempty{#3}
3623   {\mw@secundef{#1#2}{#4}}
3624   {\@namedef{#1#2}{#3}}
\mw@secundef 3625 \def\mw@secundef#1#2{%
3626   \gm@ifundefined{#1}{%
3627     \ClassError{mwcls/gm}{%
3628       command\backslash#1\undefined\MessageBreak
3629       after\backslashSetSectionFormatting!!!\MessageBreak}{%
3630       Provide the #2 argument of\backslash
3631       SetSectionFormatting.}}{}}

```

First argument is a sectioning command (wo. the backslash) and second the stuff to be added at the beginning of the heading declarations.

```

\addtoheading 3638 \def\addtoheading#1#2{%
3639     \n@melet {gmu@reserveda} {#1@head}%
3640     \edef\gmu@reserveda {\unexpanded {#2} \@xa \unexpanded {%
        \gmu@reserveda} }%
3641     \n@melet {#1@head} {gmu@reserveda}%
3643 }
3645 }% of \@ifnotmw's else.

```

Negative \addvspace

When two sectioning commands appear one after another (we may assume that this occurs only when a lower section appears immediately after higher), we prefer to put the *smaller* vertical space not the larger, that is, the preskip of the lower sectioning not the postskip of the higher.

For that purpose we modify the very inner macros of MWCLS to introduce a check whether the previous vertical space equals the postskip of the section one level higher.

```
3657 \@ifnotmw {} {% We proceed only in MWCLS.
```

The information that we are just after a heading will be stored in the `\gmu@prevsec` macro: any heading will define it as the section name and `\everypar` (any normal text) will clear it.

```

\@afterheading 3662 \def\@afterheading{%
3663     \@nbreaktrue
3664     \xdef\gmu@prevsec {\mw@HeadingType}% added now
3665     \everypar{%
3666         \grelaxen\gmu@prevsec% added now. All the rest is original LATEX.
3667         \if@nbreak
3668         \@nbreakfalse
3669         \clubpenalty_\@M
3670         \if@afterindent_\else
3671         {\setbox\z@\lastbox}%
3672         \fi
3673         \else
3674         \clubpenalty_\@clubpenalty
3675         \everypar {}%
3676         \fi}}

```

If we are (with the current heading) just after another heading (one level lower I suppose), then we add the less of the higher header's post-skip and the lower header pre-skip or, if defined, the two-header-skip. (We put the macro defined below just before `\addvspace` in `mwcls` inner macros.)

```

\gmu@checkaftersec 3683 \def\gmu@checkaftersec{%
3684     \gm@ifundefined{gmu@prevsec} {} {%
3685         \ifgmu@postsec% an additional switch that is true by default but may be
            turned into an \ifdim in special cases, see line 3721.
3688         {\@xa \mw@getflags \@xa {\gmu@prevsec}%
3689         \glet\gmu@reserveda \mw@HeadingBreakAfter}%
3690         \if \mw@HeadingBreakBefore \def\gmu@reserveda {11} \fi% if the cur-
            rent heading inserts page break before itself, all the play with vskips is
            irrelevant.
3693         \if\gmu@reserveda\else
3694         \penalty10000\relax

```

```

3695     \skip\z@=\csname\gmu@prevsec_@postskip\endcsname\relax
3696     \skip\tw@=\csname\mw@HeadingType_@preskip\endcsname\relax
3697     \gmu@ifundefined{\mw@HeadingType_@twoheadskip}{%
3698         \ifdim\skip\z@>\skip\tw@
3699         \vskip-\skip\z@% we strip off the post-skip of previous header if it's
            bigger than current pre-skip
3701     \else
3702         \vskip-\skip\tw@% we strip off the current pre-skip otherwise
3703         \fi}{% But if the two-header-skip is defined, we put it
3705         \penalty10000
3706         \vskip-\skip\z@
3707         \penalty10000
3708         \vskip-\skip\tw@
3709         \penalty10000
3710         \vskip\csname\mw@HeadingType_@twoheadskip\endcsname
3711         \relax}%
3712     \penalty10000
3713     \hrule_@height\z@\relax% to hide the last (un)skip before
            subsequent \advspaces.
3715     \penalty10000
3716     \fi
3717     \fi
3718 }% of \gmu@ifundefined{gmu@prevsec} 'else'.
3719 }% of \def\gmu@checkaftersec.

\ParanoidPostsec 3721 \def\ParanoidPostsec{% this version of \ifgmu@postsec is intended for the
            special case of sections may contain no normal text, as while gmddocing.
\ifgmu@postsec 3724 \def\ifgmu@postsec{% note this macro expands to an open \if.
3725     \skip\z@=\csname\gmu@prevsec_@postskip\endcsname\relax
3726     \ifdim\lastskip=\skip\z@\relax% we play with the vskips only if the
            last skip is the previous heading's postskip (a counter-example I met
            while gmddocing).
3730 }}
3732 \let\ifgmu@postsec\iftrue

\gmu@getadvvs 3734 \def\gmu@getadvvs#1\advspace#2\gmu@getadvvs{%
3735     \toks\z@={#1}%
3736     \toks\tw@={#2}}

            And the modification of the inner macros at last:

\gmu@setheading 3739 \def\gmu@setheading#1{%
3740     \@xa\gmu@getadvvs#1\gmu@getadvvs
3741     \edef#1{%
3742         \the\toks\z@\@nx\gmu@checkaftersec
3743         \@nx\advspace\the\toks\tw@}}
3745 \gmu@setheading\mw@normalheading
3746 \gmu@setheading\mw@runinheading

\SetTwoheadSkip 3748 \def\SetTwoheadSkip#1#2{\@namedef{#1@twoheadskip}{#2}}
3750 }% of \@ifnotmw's else.

```

My heading setup for mwcls

The setup of heading skips was tested in ‘real’ typesetting, for money that is. The skips are designed for 11/13 pt leading and together with my version of mw11.clo option file for mwcls make the headings (except paragraph and subparagraph) consist of an integer number of lines. The name of the declaration comes from my employer, “Wiedza Powszechna” Editions.

```
3762 \@ifnotmw {} {% We define this declaration only when in mwcls.
\WPheadings 3763 \DeclareCommand\WPheadings {T{\chapter}} {%
3764   \SetSectionFormatting[breakbefore,wholewidth]
3765     {part} {\z@\@plus1fill} {} {\z@\@plus3fill}%
3767   \IfValueF{#1} {%
3768     \gm@ifundefined{chapter} {} {%
3769       \SetSectionFormatting[breakbefore,wholewidth]
3770         {chapter}
3771         {66\p@} {%67\p@} for Adventor/Schola 0,95.
3772         {\FormatHangHeading{\LARGE}}
3773         {27\p@\@plus0,2\p@\@minus1\p@}%
3774     }%
3775   }% of unless #1

3777   \SetTwoheadSkip{section} {27\p@\@plus0,5\p@}%
3778   \SetSectionFormatting{section}
3779     {24\p@\@plus0,5\p@\@minus5\p@}%
3780     {\FormatHangHeading_{\Large}}
3781     {10\p@\@plus0,5\p@}% ed. Krajewska of “Wiedza Powszechna”, as we
      understand her, wants the skip between a heading and text to be rigid.

3785   \SetTwoheadSkip{subsection} {11\p@\@plus0,5\p@\@minus1\p@}%
3786   \SetSectionFormatting{subsection}
3787     {19\p@\@plus0,4\p@\@minus6\p@}
3788     {\FormatHangHeading_{\large}}% 12/14 pt
3789     {6\p@\@plus0,3\p@}% after-skip 6pt due to p.12, not to squeeze the
      before-skip too much.

3792   \SetTwoheadSkip{subsubsection} {10\p@\@plus1,75\p@\@minus1%
      \p@}%
3793   \SetSectionFormatting{subsubsection}
3794     {10\p@\@plus0,2\p@\@minus1\p@}
3795     {\FormatHangHeading_{\normalsize}}
3796     {3\p@\@plus0,1\p@}% those little skips should be smaller than you cal-
      culate out of a geometric progression, because the interline skip en-
      larges them.

3800   \SetSectionFormatting[runin]{paragraph}
3801     {7\p@\@plus0,15\p@\@minus1\p@}
3802     {\FormatRunInHeading{\normalsize}}
3803     {2\p@}%
3805   \SetSectionFormatting[runin]{subparagraph}
3806     {4\p@\@plus1\p@\@minus0,5\p@}
3807     {\FormatRunInHeading{\normalsize}}
3808     {\z@}%
3809 }% of \WPheadings
3810 }% of \@ifnotmw
```

Compatibilising standard and mwcls sectionings

If you use Marcin Woliński's document classes (mwcls), you might have met their little queerness: the sectioning commands take two optional arguments instead of standard one. It's reasonable since one may wish one text to be put into the running head, another to the toc and yet else to the page. But the order of optionalities causes an incompatibility with the standard classes: MW section's first optional argument goes to the running head not to toc and if you've got a source file written with the standard classes in mind and use the first (and only) optional argument, the effect with mwcls would be different if not error.

Therefore I counter-assign the commands and arguments to reverse the order of optional arguments for sectioning commands when mwcls are in use and reverse, to make mwcls-like sectioning optionals usable in the standard classes.

With the following in force, you may both in the standard classes and in mwcls give a sectioning command one or two optional arguments (and mandatory the last, of course). If you give just one optional, it goes to the running head and to toc as in scls (which is unlike in mwcls). If you give two optionals, the first goes to the running head and the other to toc (like in mwcls and unlike in scls).

(In both cases the mandatory last argument goes only to the page.)

What more is unlike in scls, it's that even with them the starred versions of sectioning commands allow optionals (but they still send them to the Gobbled Tokens' Paradise).

(In mwcls, the only difference between starred and non-starred sec commands is (not) numbering the titles, both versions make a contents line and a mark and that's not changed with my redefinitions.)

```
3851 \@ifnotmw{% we are not in mwcls and want to handle mwcls-like sectionings i.e.,
      those written with two optionals.
\gm@secini 3854 \def\gm@secini{gm@la}%
3855 \StoreMacros{\partmark\chaptermark\sectionmark%
      \subsectionmark\subsubsectionmark\paragraphmark}%
\gm@secxx 3857 \def\gm@secxx#1#2[#3]#4{%
3858 \ifx\gm@secstar\@empty
      a little trick to allow a special version of the heading just to the running head.
3861 \namedef{#1mark}##1{% we redefine \<sec>mark to gobble its argu-
      ment and to launch the stored true marking command on the appro-
      priate argument.
3864 \storedcstype{#1mark}{#2}%
3865 \RestoreMacro*{#1mark}% after we've done what we wanted we
      restore original \#1mark.
3867 }%
\gm@secstar 3868 \def\gm@secstar{[#3]}% if \gm@secstar is empty, which means the
      sectioning command was written starless, we pass the 'true' section-
      ing command #3 as the optional argument. Otherwise the sectioning
      command was written with star so the 'true' s.c. takes no optional.
3873 \fi
3874 \@xa\@xa\cstype\gm@secini#1\endcstype
3875 \gm@secstar{#4}}%
3877 }{% we are in mwcls and want to reverse MW's optionals order i.e., if there's just one
      optional, it should go both to toc and to running head.
\gm@secini 3880 \def\gm@secini{gm@mw}%
3882 \let\gm@secmarkh\@gobble% in mwcls there's no need to make tricks for spe-
      cial version to running headings.
```



```

\gm@secxx 3885 \def\gm@secxx#1#2[#3]#4{%
3886 \@xa\@xa\csname\gm@secini#1\endcsname
3887 \gm@secstar[#2][#3]{#4}}%
3888 }

\gm@sec 3890 \def\gm@sec#1{\@dblarg{\gm@secx{#1}}}
\gm@secx 3891 \def\gm@secx#1[#2]{%
3892 \@ifnextchar[{\gm@secxx{#1}{#2}}{\gm@secxx{#1}{#2}[#2]}}% if there's
only one optional, we double it not the mandatory argument.

\gm@straightensec 3896 \def\gm@straightensec#1{% the parameter is for the command's name.
3897 \gm@ifundefined{#1}{}{% we don't change the ontological status of the
command because someone may test it.
3899 \n@melet{\gm@secini#1}{#1}%
3900 \@namedef{#1}{%
\gm@secstar 3901 \gm@ifstar{\def\gm@secstar*}\gm@sec{#1}}{%
\gm@secstar 3902 \def\gm@secstar{}\gm@sec{#1}}}%
3903 }%

3905 \let\do\gm@straightensec
3906 \do{part}\do{chapter}\do{section}\do{subsection}\do{%
subsection}
3907 \@ifnotmw{}{\do{paragraph}}% this 'straightening' of \paragraph with the
standard article caused the 'TEX capacity exceeded' error. Anyway, who on
Earth wants paragraph titles in toc or running head?

```

enumerate* and itemize*

We wish the starred version of `enumerate` to be just numbered paragraphs. But `hyperref` redefines `\item` so we should do it a smart way, to set the L^AT_EX'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.)

```

enumeratex 3923 \@namedef{enumerate*}{%
3924 \ifnum\@enumdepth>\thr@@
3925 \toodeep
3926 \else
3927 \advance\@enumdepth\@ne
3928 \edef\@enumctr{enum\romannumeral\the\@enumdepth}%
3929 \@xa\list\csname\label\@enumctr\endcsname{%
3930 \partopsep\topsep\topsep\z@\leftmargin\z@
3931 \itemindent\@parindent\advance\itemindent\labelsep
3932 \labelwidth\@parindent
3933 \advance\labelwidth-\labelsep
3934 \listparindent\@parindent
3935 \usecounter\@enumctr
3936 \def\makelabel##1{##1\hfil}}%
3937 \fi}
3938 \@namedef{endenumerate*}{\endlist}

itemizex 3941 \@namedef{itemize*}{%
3942 \ifnum\@itemdepth>\thr@@
3943 \toodeep

```

```

3944 \else
3945 \advance\@itemdepth\@ne
3946 \edef\@itemitem{labelitem\romannumeral\the\@itemdepth}%
3947 \@xa\list\csname\@itemitem\endcsname{%
3948 \partopsep\topsep_\topsep\z@\_leftmargin\z@
3949 \itemindent\@parindent
3950 \labelwidth\@parindent
3951 \advance\labelwidth-\labelsep
3952 \listparindent\@parindent
3953 \def\makelabel##1{##1\hfil_\}}%
3954 \fi}
3955 \@namedef{enditemize*}{\endlist}

```

The logos

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

```

3964 \let\oldLaTeX\LaTeX
3965 \let\oldLaTeXe\LaTeXe
3967 \pdef\TeX{T\kern-.1667em\lower.5ex\hbox{E}\kern-.125emX\@}
3968 \StoreMacro\TeX
3969 \AtBeginDocument{\RestoreMacro\TeX}
\DeclareLogo 3971 \newcommand*\DeclareLogo[3][\relax]{%
% [#1] is for non-LATEX spelling and will be used in the PD1 encoding (to
% make pdf bookmarks);
% #2 is the command, its name will be the PD1 spelling by default,
% #3 is the definition for all the font encodings except PD1.
\gmu@reserveda 3979 \ifx\relax#1\def\gmu@reserveda{\@xa\@gobble\string#2}%
3980 \else
\gmu@reserveda 3981 \def\gmu@reserveda{#1}%
3982 \fi
3983 \edef\gmu@reserveda{%
3984 \@nx\DeclareTextCommand\@nx#2{PD1}{\gmu@reserveda}}
\gmu@reserveda
3985 \gmu@reserveda
3986 \DeclareTextCommandDefault#2{#3}%
\pdef 3987 \pdef#2{#3}% added for XYLATEX.
3988 }
\DeclareLogo 3991 \DeclareLogo\LaTeX{%
3992 {%
3993 L%
3994 \setbox\z@\hbox{\check@mathfonts
3995 \fontsize\sf@size\z@
3996 \math@fontsfalse\selectfont
3997 A}%
3998 \kern-.57\wd\z@
4000 \sbox\tw@_T%
4001 \vbox_\to\ht\tw@{\copy\z@\_vss}%
4002 \kern-.2\wd\z@% originally -,15em for T.
4003 }%
4004 {%
4005 \ifdim\fontdimen1\font=\z@

```

```

4006     \else
4007         \count \z@=\fontdimen5\font
4008         \multiply\count \z@_by_64\relax
4009         \divide\count \z@_by_p@
4010         \count \tw@=\fontdimen1\font
4011         \multiply\count \tw@_by\count \z@
4012         \divide\count \tw@_by_64\relax
4013         \divide\count \tw@_by\tw@
4014         \kern-\the\count \tw@_sp\relax
4015     \fi}%
4016 \TeX}

\LaTeXe 4018 \DeclareLogo\LaTeXe{\mbox{\m@th_\if
4019     b\expandafter \@car \f@series \@nil \boldmath \fi
4020     \LaTeX\kern.15em2$_{\textstyle \varepsilon}$}}

4022 \StoreMacro\LaTeX
4023 \StoreMacro*{LaTeX_}

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

```

```

\LaTeXpar 4029 \DeclareLogo[(La)TeX]{\LaTeXpar}{%
4030     {%
4031         \setbox \z@ \hbox{()% }
4032         \copy \z@
4033         \kern-.2\wd \z@_L%
4034         \setbox \z@ \hbox{\check@mathfonts
4035             \fontsize \sf@size \z@
4036             \math@fontsfalse \selectfont
4037             A}%
4038         \kern-.57\wd \z@
4039         \sbox \tw@_T%
4040         \vbox_to \ht \tw@ {\box \z@%
4041             \vss}%
4042     }%
4043     \kern-.07em% originally -,15 em for T.
4044     {% (
4045         \sbox \z@)%
4046         \kern-.2\wd \z@ \copy \z@
4047         \kern-.2\wd \z@} \TeX
4048 }

```

“Here are a few definitions which can usefully be employed when documenting package files: now we can readily refer to $\mathcal{A}\mathcal{M}\mathcal{S}\text{-}\text{T}\text{E}\text{X}$, $\text{B}\mathbb{B}\text{T}\text{E}\text{X}$ and $\text{S}\text{L}\text{T}\text{E}\text{X}$, as well as the usual TEX and $\text{L}\text{T}\text{E}\text{X}$. There’s even a $\text{PLAIN}\ \text{T}\text{E}\text{X}$ and a WEB .”

```

4055 \gm@ifundefined{AmSTeX}
\AmSTeX 4056 {\def\AmSTeX{\leavevmode\hbox{\mathcal_A\kern-.2em%
         \lower.376ex%
4057         \hbox{\mathcal_M$}\kern-.2em\mathcal_SS-\TeX}}}{ }

\BibTeX 4059 \DeclareLogo\BibTeX{\rmfamily_B\kern-.05em%
4060     \textsc{i{\kern-.025em}b}\kern-.08em% the kern is wrapped in braces
         for my \fakescaps' sake.
4062     \TeX}}

\SlitEX 4065 \DeclareLogo\SlitEX{\rmfamily_S\kern-.06emL\kern-.18em%

```

```

\raise.32ex\hbox
4066      {\scshape\i}\kern-.03em\TeX}}
\PlainTeX 4068 \DeclareLogo\PlainTeX{\textsc{Plain}\kern2pt\TeX}
\Web       4070 \DeclareLogo\Web{\textsc{Web}}

There's also the (L)TeX logo got with the \LaTeXpar macro provided by gmutils.
And here The TeXbook's logo:

\TeXbook  4073 \DeclareLogo[The\TeX\book]\TeXbook{\textsl{The\TeX\book}}
4074 \let\TB\TeXbook% TUG Boat uses this.

\TeX       4076 \DeclareLogo[e-TeX]\eTeX{%
4077   \iffontchar\font"03B5{\itshape\epsilon}\else
4078   \ensuremath{\varepsilon}\fi-\kern-.125em\TeX}% definition sent by
      Karl Berry from TUG Boat itself.

4081 \StoreMacro\eTeX

\pdfTeX    4083 \DeclareLogo[pdfe-TeX]\pdfTeX{pdf\eTeX}
\pdfTeX    4085 \DeclareLogo\pdfTeX{pdf\TeX}
\pdfLaTeX  4086 \DeclareLogo\pdfLaTeX{pdf\LaTeX}

4089 \gm@ifundefined{XeTeX}{%
\XeTeX     4090   \DeclareLogo\XeTeX{X\kern-.125em\relax
4091     \gm@ifundefined{reflectbox}{%
4092       \lower.5ex\hbox{E}\kern-.1667em\relax}{%
4093       \lower.5ex\hbox{\reflectbox{E}}\kern-.1667em\relax}%
4094     \TeX}}{}

4096 \gm@ifundefined{XeLaTeX}{%
\XeLaTeX   4097   \DeclareLogo\XeLaTeX{X\kern-.125em\relax
4098     \gm@ifundefined{reflectbox}{%
4099       \lower.5ex\hbox{E}\kern-.1667em\relax}{%
4100       \lower.5ex\hbox{\reflectbox{E}}\kern-.1667em\relax}%
4101     \LaTeX}}

As you see, if TeX doesn't recognise \reflectbox (graphics isn't loaded), the first E
will not be reversed. This version of the command is intended for non-XeTeX usage. With
XeTeX, 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.

\LuaTeX    4109 \DeclareLogo[LuaTeX]\LuaTeX{\textsc{Lua}\TeX}

```

Expandable turning stuff all into 'other'

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

```
\detoken@xa 4128 \long\def\detoken@xa#1{\detokenize\@xa{#1}}
```

The next macro originates from the ancient era when I didn't know about ϵ -TeX's `\detokenize`. A try to redefine it to `\detokenize\@xa{#1}` resulted in error so (vo.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 4138 \long\def\all@other#1{\@xa\gm@gobmacro\meaning#1}
```

The `\gm@gobmacro` macro above is applied to gobble the `\meaning's` beginning, long `\macro:->` all 'other' that is.

```

4143 \edef\gmu@tempa{%
\gm@gobmacro 4144 \def\@nx\gm@gobmacro##1\@xa\@gobble\string\macro:##2->{}}
4145 \gmu@tempa

```

Brave New World of XeTeX

```

\@ifXeTeX 4150 \def\@ifXeTeX{% two-argument command
4151 \ifdefined\XeTeXversion
4152 \unless\ifx\XeTeXversion\relax\afterfifi\@firstoftwo\else%
\afterfifi\@secondoftwo\fi
4153 \else\afterfi\@secondoftwo\fi}

XeTeXifprefix 4155 \def\XeTeXifprefix{% to be used as prefix to an \if... test.
4156 \@ifXeTeX{}{\unless}}

XeTeXthree 4158 \DeclareCommand\XeTeXthree{o}{%
4162 \@ifXeTeX{%
4163 \IfValueT{#1}{\PassOptionsToPackage{#1}{fontspec}}%
4164 \@ifpackageloaded{gmverb}{\StoreMacro\verb}{}%
4165 \RequirePackage{xltextra}% since v 0.4 (2008/07/29) this package re-
defines \verb and verbatim*, and quite elegantly provides an option
to suppress the redefinitions, but unfortunately that option excludes also
a nice definition of \xxt@visibleSPACE which I fancy.
4172 \@ifpackageloaded{gmverb}{\RestoreMacro\verb}{}%
4173 \AtBeginDocument{%
4174 \RestoreMacro\LaTeX\RestoreMacro*{LaTeX}% my version of
the LATEX logo has been stored just after defining, in line 4023.
4177 \RestoreMacro\eTeX}%

adddefaultfontfeatures 4179 \pdef\adddefaultfontfeatures##1{%
4180 \addtomacro\zf@default@options{#1,}}
4181 }% of \@ifXeTeX's first argument,
4182 }% \@ifXeTeX's second argument,
4183 }% of \XeTeXthree's body.

```

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

```

4189 \AtBeginDocument{%
4190 \@ifpackageloaded{fontspec}{%
\udigits 4191 \pdef\udigits{%
4192 \addfontfeature{Numbers=Uppercase}}%
4193 }{%
4194 \emptify\udigits}}

```

Fractions

```

\Xedekfracc 4199 \def\Xedekfracc{\gm@ifstar\gmu@xedekfraccstar%
\gmu@xedekfraccplain}

```

(plain) The starless version turns the font feature `frac` on.

(*) But nor my modification of Minion Pro neither TeX Gyre Pagella doesn't feature the `frac` font feature properly so, with the starred version of the declaration we use the

characters from the font where available (see the \@namedefs below) and the numr and dnom features with the fractional slash otherwise (via \gmu@dekfracc).

(**) But Latin Modern Sans Serif Quotation doesn't support the numerator and denominator positions so we provide the double star version for it, which takes the char from font if it exist and typesets with lowers and kerns otherwise.

```

\gmu@xedekfraccstar 4214 \def\gmu@xedekfraccstar{%
\gmu@xefracccdef    4215 \def\gmu@xefracccdef##1##2{%
                    4216 \iffontchar\font_##2
                    4217 \@namedef{gmu@xefraccc##1}{\char##2_}%
                    4218 \else
                    4219 \n@melet{gmu@xefraccc##1}{relax}%
                    4220 \fi}%
\gmu@dekfracc       4222 \def\gmu@dekfracc##1/##2{%
                    4223 {\addfontfeature{VerticalPosition=Numerator}##1}%
                    \gmu@numeratorkern
                    4224 \char"2044_\gmu@denominatorkern
                    4225 {\addfontfeature{VerticalPosition=Denominator}##2}}%

```

We define the fractional macros. Since Adobe Minion Pro doesn't contain $\frac{n}{5}$ nor $\frac{n}{6}$, we don't provide them here.

```

4229 \gmu@xefracccdef{1/4}{"BC}%
4230 \gmu@xefracccdef{1/2}{"BD}%
4231 \gmu@xefracccdef{3/4}{"BE}%
4232 \gmu@xefracccdef{1/3}{"2153}%
4233 \gmu@xefracccdef{2/3}{"2154}%
4234 \gmu@xefracccdef{1/5}{"2155}%
4235 \gmu@xefracccdef{2/5}{"2156}%
4236 \gmu@xefracccdef{3/5}{"2157}%
4237 \gmu@xefracccdef{4/5}{"2158}%
4238 \gmu@xefracccdef{1/6}{"2159}%
4239 \gmu@xefracccdef{5/6}{"215A}%
4240 \gmu@xefracccdef{1/8}{"215B}%
4241 \gmu@xefracccdef{3/8}{"215C}%
4242 \gmu@xefracccdef{5/8}{"215D}%
4243 \gmu@xefracccdef{7/8}{"215E}%
\dekfracc@args     4244 \pdef\dekfracc@args##1/##2{%
                    4245 \gm@ifundefined{gmu@xefraccc\detokenize{##1/##2}}{%
                    4246 \gmu@dekfracc{##1}/{##2}}{%
                    4247 \csname_\gmu@xefraccc\detokenize{##1/##2}\endcsname}%
                    4248 \if@gmu@mmhbox\egroup\fi
                    4249 }% of \dekfracc@args.
                    4250 \gm@ifstar{\let\gmu@dekfracc\gmu@dekfracccsimple}{}%
                    4251 }
\gmu@xedekfraccplain 4253 \def\gmu@xedekfraccplain{% 'else' of the main \gm@ifstar
\dekfracc@args       4254 \pdef\dekfracc@args##1/##2{%
                    4255 \ifmmode\hbox\fi{%
                    4256 \addfontfeature{Fractions=On}%
                    4257 ##1/##2}%
                    4258 \if@gmu@mmhbox\egroup\fi
                    4259 }% of \dekfracc@args
                    4260 }

```

```
\if@gmu@mmhbox 4262 \newif\if@gmu@mmhbox% we'll use this switch for \dekfracc and also for \thous
(hacky thousand separator).
```

```
\dekfracc 4265 \pdef\dekfracc{%
4267 \iffmode\hbox\bgroup\@gmu@mmhboxtrue\fi
4268 \dekfracc@args}
```

```
\gmu@numeratorkern 4271 \def\gmu@numeratorkern{\kern-.055em\relax}
\gmu@denominatorkern 4272 \def\gmu@denominatorkern{\kern-.05em\relax}
```

What have we just done? We defined two versions of the `\XeFractions` declaration. The starred version is intended to make use only of the built-in fractions such as $\frac{1}{2}$ or $\frac{7}{8}$. To achieve that, a handful of macros is defined that expand to the Unicodes of built-in fractions and `\dekfracc` command is defined to use them.

The unstarred version makes use of the Fraction font feature and therefore is much simpler.

Note that in the first argument of `\gm@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 `\gm@ifstar` so maybe it's subject to change.)

A simpler version of `\dekfracc` is provided in line [5722](#).

`\resizegraphics`

```
\resizegraphics 4294 \pdef\resizegraphics#1#2#3{% 2009/11/17 works bad with a file whose
name contains spaces so I return \XeTeXpicfile
4299 \resizebox{#1}{#2}{%
4300 \edef\gmu@tempa{\@nx\csname\XeTeX\@nx\@ifendswithpdf{%
4301 \@xa\string\csname#3\endcsname}{pdf}{pic}file\@nx%
\endcsname}%
4302 \gmu@tempa\@#3"\relax}}
4304 \edef\gmu@tempa{%
4305 \def\@nx\@ifendswithpdf##1{%
4306 \unexpanded{%
4307 \ifnum
4308 \if\relax\gmu@pdfdetector}##1%
4309 \detokenize{pdf}\unexpanded{\relaxo\else1\fi}% we expand to
1 if #1 ends with lowercase 'pdf' of cat. 12
4311 \unexpanded{\if\relax\gmu@PDFdetector}##1%
4312 \detokenize{PDF}\unexpanded{\relaxo\else1\fi}% we expand to
1 if #1 ends with uppercase 'PDF' of cat. 12
4314 >0
4315 \unexpanded{\@xa\@firstoftwo\else\@xa\@secondoftwo\fi}%
4316 }% of \@ifendswithpdf
4320 \def\@nx\gmu@pdfdetector##1\detokenize{pdf}{}%
4321 \def\@nx\gmu@PDFdetector##1\detokenize{PDF}{}%
4322 }\gmu@tempa
\textsuperscript@@ 4325 \pdef\textsuperscript@@#1{%
4326 \@textsuperscript{\selectfont#1}}%
\@textsuperscript 4327 \def\@textsuperscript#1{%
4328 {\m@th\ensuremath{\^{\mbox{\fontsize\sf@size\z@#1}}}}}
4330 \let\textsuperscript@@\textsuperscript
```

```

\GMtextsuperscript 4332 \def\GMtextsuperscript{%
4333   \@ifXeTeX{%
\textsuperscript 4334     \DeclareCommand\textsuperscript{sm}{%
4335       \IfValueTF{##1}{\textsuperscript@@{##2}}%
4336       {%
4337         \begingroup
4338         \addfontfeature{VerticalPosition=Numerator}##2%
4339         \endgroup}}%
4340   }{\truetextsuperscript}}
\truetextsuperscript 4342 \def\truetextsuperscript{%
4343   \let\textsuperscript\textsuperscript@@
4344 }

```

Settings for mathematics in main font

`\gmath` and `\garamath` 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(spec-ification)>)
```

Note that `\gmath` without first optional argument has always to come first because otherwise it overwrite settings of your math version.

```

\gmu@getfontstring 4378 \def\gmu@getfontstring{%
4379   \xdef\gmu@fontstring{%
4380     \gmu@fontstring@}}
\gmu@fontstring@ 4382 \def\gmu@fontstring@{%
4383   \@xa \@xa \@xa \gmu@fontstring@@ \@xa \meaning\the\font \@@nil}
\gmu@fontstring@@ 4391 \def\gmu@fontstring@@#1"#2"#3\@@nil{"#2"}
\gmu@getfontscale 4393 \def\gmu@getfontscale#1Scale#2=#3,{%
4394   \ifx\gmu@getfontscale#3\else
\gmu@tempa 4395     \def\gmu@tempa{MatchLowercase}%
\gmu@tempb 4396     \def\gmu@tempb{#3}%
4397     \ifx\gmu@tempa\gmu@tempb
4398       \gmu@calc@scale{5}%
4399       \@xa \@firstoftwo
4400     \else
\gmu@tempa 4401     \def\gmu@tempa{MatchUppercase}%
4402     \ifx\gmu@tempa\gmu@tempb

```



```

4403         \gmu@calc@scale{8}%
4404         \afterfifi \@firstoftwo
4405         \else\afterfifi \@secondoftwo
4406     \fi
4407 \fi
4408 {\xdef\gmu@fontscalebr{[\gmu@fontscale]_}}%
4409 {\xdef\gmu@fontscalebr{[#3]_}}%
4410 \afterfi\gmu@getfontscale
4411 \fi
4412 }

\gmu@getfontdata 4415 \def\gmu@getfontdata#1{%
4416   \global\emptify\gmu@fontscalebr
4417   \begingroup
4418   #1%
4420   \@xa \@xa \@xa \gmu@getfontscale
4421   \csname_#1zf@family@options\ffamily\endcsname
4422   ,Scale=\gmu@getfontscale,%
4423   \gmu@getfontstring
4424   \xdef\gmu@theskewchar{\the\skewchar\font}%
4425   \endgroup}

\gmu@stripchar 4428 \def\gmu@stripchar#1{"}

\gmath@getfamnum 4430 \DeclareCommand\gmath@getfamnum{C{\gmath@fam}}{%
4431   \edef\gmath@famnum{\@xa\gmu@stripchar\meaning#1}%
4432 }

\XeTeXmathcode<char slot> [ <=> ] <type> <family> <char slot>

\gmathFams 4436 \DeclareCommand\gmathFams{o_#1% the name of math version
4437   C{\NoValue}_#1% 'rescue' font. Will be accessible via \symgmathRoman/ <version>
         (math family) and \gmath@fontt/ <version> (font).
4440 }{%
4442   \IfValueT{#1}{%
4443     \DeclareMathVersion{#1}% this sets the defaults so no need to define
         them explicitly
4449     }% of if #1 given

4451   \gmu@getfontdata{\rmfamily\itshape}%
4452   \edef\gmu@tempa{%
4453     \IfValueTF{#1}{\@nx\SetSymbolFont{letters}{#1}}%
4454     {\@nx\DeclareSymbolFont{letters}}%
4455     {\encodingdefault}{gmathit\PutIfValue{#1}}{m}{it}%
4456     \IfValueT{#1}{%
4457       \@nx\DeclareSymbolFont{letters#1}%
4458       {\encodingdefault}{gmathit\PutIfValue{#1}}{m}{it}%
4459       }%
4460     }%
4461     \@nx\DeclareFontFamily{\encodingdefault}{gmathit%
         \PutIfValue{#1}}{%
4462       \skewchar\font\gmu@theskewchar\space}%
4463     \@nx\DeclareFontShape{\encodingdefault}{gmathit%
         \PutIfValue{#1}}{m}{it}{%
4464       <->_#1\gmu@fontscalebr_#1\gmu@fontstring}{}%
4465     \IfValueT{#1}{%

```

```

4466     \@nx\SetMathAlphabet \@nx\mathit {#1} {%
          \encodingdefault} {gmathit#1} {m} {it}}%
4467 } \gmu@tempa
4469 \IfValueT {#2} {%
4470   \gmu@getfontdata {#2 \gmu@ifstored {\upshape} {%
          \storedcsname {upshape}} {\upshape}}%
4471   \edef \gmu@tempa {%
4472     \@nx\DeclareSymbolFont {gmathRoman \PutIfValue {#1}}%
4473     {\encodingdefault} {gmathRm \PutIfValue {#1}} {m} {n}%
4474     \@nx\DeclareFontFamily {\encodingdefault} {gmathRm%
          \PutIfValue {#1}} {%
4475       \skewchar \font \gmu@theskewchar \space}%
4476     \@nx\DeclareFontShape {\encodingdefault} {gmathRm%
          \PutIfValue {#1}} {m} {n} {%
4477       <-> \gmu@fontscalebr \gmu@fontstring} {}%
4478   } \gmu@tempa
4479   \@xa \font \csname \gmu@fontt \PutIfValue {#1} \endcsname
4480   = \gmu@fontstring \relax

4482   \gmu@getfontdata {#2 \gmu@ifstored {\itshape} {%
          \storedcsname {itshape}} {\itshape}}%
4483   \edef \gmu@tempa {%
4484     \@nx\DeclareSymbolFont {gmathItalic \PutIfValue {#1}}%
4485     {\encodingdefault} {gmathIt \PutIfValue {#1}} {m} {n}%
4486     \@nx\DeclareFontFamily {\encodingdefault} {gmathIt%
          \PutIfValue {#1}} {%
4487       \skewchar \font \gmu@theskewchar \space}%
4488     \@nx\DeclareFontShape {\encodingdefault} {gmathIt%
          \PutIfValue {#1}} {m} {n} {%
4489       <-> \gmu@fontscalebr \gmu@fontstring} {}%
4490   } \gmu@tempa
4491 }% of if #2 given

4493 \gmu@getfontdata {\rmfamily \upshape}%
4494 \edef \gmu@tempa {%
4495   \IfValueTF {#1} {\@nx\SetSymbolFont {gmathroman} {#1}}%
4496   {\@nx\DeclareSymbolFont {gmathroman}}%
4497   {\encodingdefault} {gmathrm \PutIfValue {#1}} {m} {n}%
4498   \@nx\DeclareFontFamily {\encodingdefault} {gmathrm%
          \PutIfValue {#1}} {%
4499     \skewchar \font \gmu@theskewchar \space}%
4500   \@nx\DeclareFontShape {\encodingdefault} {gmathrm%
          \PutIfValue {#1}} {m} {n} {%
4501     <-> \gmu@fontscalebr \gmu@fontstring} {}%
4502   \IfValueT {#1} {%
4503     \@nx\SetMathAlphabet \@nx\mathrm {#1} {%
          \encodingdefault} {gmathrm#1} {m} {n}}%
4504   } \gmu@tempa
4507   \@xa \font \csname \gmu@font \PutIfValue {#1} \endcsname
4508   = \gmu@fontstring \relax
4509   \gmathfamshook
4510 }
4512 \emptify \gmathfamshook

```

```

\gmathbase 4514 \DeclareCommand\gmathbase {oC {\NoValue}} {%
4515   \gmathFams [#1] (#2) %
\gmath@do 4516   \DeclareCommand\gmath@do {%
4517     m_ % (1) the character or CS to be declared,
4518     o_ % (2) the Unicode to be assigned,
4519     m_ % (3) math type (CS like \mathord etc.)
4520     C {\gmath@fam}_ % font family
4521   } {%
4522     \gmath@getfamnum (##4) %
4523     \IfValueTF {##2} {%
4524       \edef\gmu@tempa {%
4525         =_ \mathchar@type##3 \space
4526         \gmath@famnum \space
4527         "##2 \relax}%
4528       \if\relax \@nx##1 %
4529         \gmu@ifstored {##1} {} {\StoreMacro##1}%
4530         \edef\gmu@tempa {%
4531           \XeTeXmathchardef_ \@nx##1 \gmu@tempa}%
4532         \else
4533         \edef\gmu@tempa {%
4534           \XeTeXmathcode_ `##1_ \gmu@tempa}
4535         \fi%
4536       }%
4537     }%
4538     {% no value of ##2
4539       \edef\gmu@tempa {%
4540         \XeTeXmathcode_ `##1_ =
4541         \mathchar@type##3 \space
4542         \gmath@famnum \space
4543         `##1 \relax}%
4544       }%
4545     \gmu@tempa
4546   }% of \gmath@do
\gmath@doif 4552 \DeclareCommand\gmath@doif {%
4553   m_ % (1) the Unicode hex of char enquired,
4554   m_ % (2) the char or CS to be declared,
4555   m_ % (3) math type CS(\mathord etc.),
4556   o_ % (4) second-choice Unicode (taken if first-choice is absent),
4557   o_ % (5) third-choice Unicode (as above if second-choice is absent
4558   from font).
4559   B {\gmath@fam}_ % (6) never used in this package. Why?
4560   C {\NoValue}
4561 } {%
4562   \gmu@storeifnotyet {##2}%
4563   \@xa \let \@xa \gmath@ft \csname
4564     gmath@font%
4565     \ifx\gmath@fam##6 \else_ t \fi
4566     \gmath@version
4567   \endcsname
4568   \iffontchar \gmath@ft "##1_ \gmath@do##2 [##1] ##3 (##6) %
4569   \else
4570     \IfValueTF {##4} {%
4571       \iffontchar \gmath@ft "##4_ %

```

```

\gmath@do##2[##4]##3(##6)%
4571 \else
4572 \IfValueTF{##5}{%
4573 \iffontchar\gmath@ft"##5\char
\gmath@do##2[##5]##3(##6)%
4574 \else
4575 \gmath@restore{##1}{##2}{##3}{##4}{##5}{%
##7}%
4576 \fi}%
4577 {\gmath@restore{##1}{##2}{##3}{##4}{##5}{##7}}%
4578 \fi}%
4579 {\gmath@restore{##1}{##2}{##3}{##4}{##5}{##7}}%
4580 \fi
4581 }% 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.
\gmath@restore 4585 \def\gmath@restore##1##2##3##4##5##6{%
4586 \IfValueT{##6}%
4587 {\ifcsname\char##6\endcsname
4588 \edef\gmu@tempa{%
4589 \unexpanded{\gmath@doif{##1}{##2}{%
##3}[##4][##5]}%
4590 {\@xa\@nx\csname##6\endcsname}\relax
4591 }\gmu@tempa
4592 \@xa\@gobbletwo\char##6% if family ##6 is defined, we gobble the other
branch
4594 \fi
4595 }%
4596 \firstofone
4597 {\if\relax\@nx##2%
4598 \RestoreMacro##2%
4599 \fi
4600 }%
4601 }%
4603 \iffalse\char##6% doesn't work in a non-math font.
\gmath@delc 4604 \DeclareCommand\gmath@delc{mo}{%
% #1 the char or CS to be declared,
% [#2] the Unicode (if not the same as the char).
4610 \gmath@getfamnum
4611 \IfValueTF{##2}{%
4612 \edef\gmu@tempa{%
4613 =\char##2\relax}%
4614 \edef\gmu@tempa{%
4615 \XeTeXdelcode\char##2\gmu@tempa}
4616 }%
4617 {%
4618 \edef\gmu@tempa{%
4619 \XeTeXdelcode\char##2=}
4620 \gmath@famnum\space
4622 `##2\relax}%
4623 }%
4624 \gmu@tempa
4627 }% of \gmath@delc

```

```

\gmath@delcif 4629 \def\gmath@delcif##1##2{%
                % #1 the Unicode enquired,
                % #2 the char to be delcode-declared
4635 \iffontchar\gmath@font"##1\gmath@delc##2[##1]\fi}
4636 \fi}% of iffalse

\gmath@delimif 4638 \def\gmath@delimif##1##2##3{%
                % #1 the Unicode enquired,
                % #2 the CS defined as \XeTeXdelimiter,
                % #3 the math type CS (probably \mathopen or \mathclose).
4645 \iffontchar\gmath@font"##1
4646 \gmath@getfamnum
4647 \protected\edef##2{\@nx\ensuremath{%
4648 \XeTeXdelimiter\mathchar@type##3\space
4649 \gmath@famnum\space"##1\relax}}%
4650 \fi}% of \gmath@delimif.

\rmopname 4652 \pdef\rmopname##1##2##3{%
4653 \mathop{##1\kern\z@}\mathrm{##3}}\csname n##2limits@%
\endcsname

4654 }%

\gmu@dogmathbase 4656 \DeclareCommand\gmu@dogmathbase{oC{\NoValue}}{%
4658 \RestoreMacro\mathchar@type%
4660 \IfValueT{##1}{\mathversion{##1}}%
4662 \edef\gmath@version{\PutIfValue{##1}}%
4665 \@xa\let\@xa\gmath@fam\csname\symgmathroman%
4666 \endcsname
4667 \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.

4671 \typeout{@@@ \gmutils.sty: taking some math chars from
                the font^^J\gmu@fontstring}%

4672 \gmath@do+\mathbin
4673 \gmath@doif{2212}-\mathbin[2013](\gmath@famm)% minus sign if
                present or else en dash

4674 \gmath@do=\mathrel
4675 \gmath@do\mathord
4676 \gmath@do1\mathord
4677 \gmath@do2\mathord
4678 \gmath@do3\mathord
4679 \gmath@do4\mathord
4680 \gmath@do5\mathord
4681 \gmath@do6\mathord
4682 \gmath@do7\mathord
4683 \gmath@do8\mathord
4684 \gmath@dog\mathord
4686 \gmath@doif{2264}\le\mathrel(\gmath@famm)%
4687 \let\leq\le
4688 \let\leeng\le
4689 \gmath@doif{2265}\ge\mathrel(\gmath@famm)%
4690 \let\geq\ge
4691 \let\geeng\ge
4692 \gmath@doif{2A7D}\xleq\mathrel(\gmath@famm)%

```

```

4693 \gmath@doif{2A7E}\xgeq\mathrel(\gmath@famm)%
4694 \@ifpackageloaded{polski}{%
4695   \ifdefined\xleq
4696   \gmu@storeifnotyet\leq
4697   \let\leq=\xleq
4698   \let\le=\leq
4699   \fi
4700   \ifdefined\xgeq
4701   \gmu@storeifnotyet\geq
4702   \let\geq=\xgeq
4703   \let\ge=\geq
4704   \fi}{}%
4706 \gmath@do.\mathpunct
4707 \gmath@do,\mathpunct
4708 \gmath@do;\mathpunct
4709 \gmath@do...\mathpunct
4710 \gmath@do(\mathopen
4713 \gmath@do)\mathclose
4715 \gmath@do[\mathopen
4717 \gmath@do]\mathclose
4720 \gmath@doif{00D7}\times\mathbin(\gmath@famm)%
4721 \gmath@do:\mathrel
4722 \gmath@doif{00B7}\cdot\mathbin(\gmath@famm)%
4723 \gmath@doif{22C6}\*\mathbin(\gmath@famm)% low star
4724 \gmath@doif{2300}\varnothing\mathord(\gmath@famm)%
4725 \gmath@doif{221E}\infty\mathord(\gmath@famm)%
4726 \gmath@doif{2248}\approx\mathrel(\gmath@famm)%
4727 \gmath@doif{2260}\neq\mathrel(\gmath@famm)%
4728 \let\ne\neq
4729 \gmath@doif{00AC}\neg\mathbin(\gmath@famm)%
4730 \gmath@doif{00AC}\nego\mathord(\gmath@famm)%
4731 \gmath@do/\mathop
4732 \gmath@do<\mathrel
4734 \gmath@do>\mathrel
4736 \gmath@doif{2329}\langle\mathopen(\gmath@famm)%
4737 \gmath@doif{232A}\rangle\mathclose(\gmath@famm)%
4738 \gmath@doif{2202}\partial\mathord(\gmath@famm)%
4739 \gmath@doif{00B1}\p\mathbin(\gmath@famm)%
4740 \gmath@doif{007E}\sim\mathrel(\gmath@famm)%
4741 \gmath@doif{2190}\leftarrow\mathrel(\gmath@famm)%
4742 \gmath@doif{2192}\rightarrow\mathrel(\gmath@famm)%
4743 \gmath@doif{2194}\leftrightarrow\mathrel(\gmath@famm)% if not
      present, \gmathfurther will take care of it if left and right arrows are
      present.
4746 \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.
4750 \gmath@doif{2193}\downarrow\mathrel(\gmath@famm)%
4752 \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.

```

4756 \if\iffontchar\gmath@font"25CA□0\else□1\fi
4757 \iffontchar\gmath@font"25FB□0\else\iffontchar%
\gmath@font"25A1□0\else□2\fi\fi
4758 \gmath@do\lozenge[25CA]\mathord
4759 \gmath@doif{25FB}\square\mathord[25A1](\gmath@famm)% 'medium
white square (modal operator)' of just 'white square'.

4761 \fi
4762 \gmath@doif{EB08}\bigcircle\mathbin(\gmath@famm)%
4763 \gmath@doif{2227}\wedge\mathbin(\gmath@famm)%
4764 \gmath@doif{2228}\vee\mathbin(\gmath@famm)%
4766 \gmath@doif{0393}\Gamma\mathalpha(\gmath@famm)%
4767 \gmath@doif{0394}\Delta\mathalpha(\gmath@famm)%
4768 \gmath@doif{0398}\Theta\mathalpha(\gmath@famm)%
4769 \gmath@doif{039B}\Lambda\mathalpha(\gmath@famm)%
4770 \gmath@doif{039E}\Xi\mathalpha(\gmath@famm)%
4772 \gmath@doif{03A3}\Sigma\mathalpha(\gmath@famm)%
4773 \gmath@doif{03A5}\Upsilon\mathalpha(\gmath@famm)%
4774 \gmath@doif{03A6}\Phi\mathalpha(\gmath@famm)%
4775 \gmath@doif{03A8}\Psi\mathalpha(\gmath@famm)%
4776 \gmath@doif{03A9}\Omega\mathalpha(\gmath@famm)%
4778 \@xa\let\@xa\gmath@fam\csname□symletters\gmath@version%
4779 \endcsname
4780 \edef\gmath@famm{symgmathItalic\gmath@version}%
4782 \gmath@doif{03B1}\alpha\mathalpha(\gmath@famm)%
4783 \gmath@doif{03B2}\beta\mathalpha(\gmath@famm)%
4784 \gmath@doif{03B3}\gamma\mathalpha(\gmath@famm)%
4785 \gmath@doif{03B4}\delta\mathalpha(\gmath@famm)%
4786 \gmath@doif{03F5}\epsilon\mathalpha(\gmath@famm)%
4787 \gmath@doif{03B5}\varepsilon\mathalpha(\gmath@famm)%
4788 \gmath@doif{03B6}\zeta\mathalpha(\gmath@famm)%
4789 \gmath@doif{03B7}\eta\mathalpha(\gmath@famm)%
4790 \gmath@doif{03B8}\theta\mathalpha(\gmath@famm)%
4791 \gmath@doif{03D1}\vartheta\mathalpha(\gmath@famm)%
4792 \gmath@doif{03B9}\iota\mathalpha(\gmath@famm)%
4793 \gmath@doif{03BA}\kappa\mathalpha(\gmath@famm)%
4794 \gmath@doif{03BB}\lambda\mathalpha(\gmath@famm)%
4795 \gmath@doif{03BC}\mu\mathalpha(\gmath@famm)%
4796 \gmath@doif{03BD}\nu\mathalpha(\gmath@famm)%
4797 \gmath@doif{03BE}\xi\mathalpha(\gmath@famm)%
4798 \gmath@doif{03C0}\pi\mathalpha(\gmath@famm)%
4799 \gmath@doif{03A0}\Pi\mathalpha(\gmath@famm)%
4800 \gmath@doif{03C1}\rho\mathalpha(\gmath@famm)%
4801 \gmath@doif{03C3}\sigma\mathalpha(\gmath@famm)%
4802 \gmath@doif{03DA}\varsigma\mathalpha(\gmath@famm)% 03C2?
4803 \gmath@doif{03C4}\tau\mathalpha(\gmath@famm)%
4804 \gmath@doif{03C5}\upsilon\mathalpha(\gmath@famm)%
4805 \gmath@doif{03D5}\phi\mathalpha(\gmath@famm)%
4806 \gmath@doif{03C7}\chi\mathalpha(\gmath@famm)%
4807 \gmath@doif{03C8}\psi\mathalpha(\gmath@famm)%
4808 \gmath@doif{03C9}\omega\mathalpha(\gmath@famm)%
4810 \if□1□1%
4811 \iffontchar\gmath@font"221A
4812 \fontdimen61\gmath@font=1pt

```

```

4813     \edef\sqrtsign{%
4814         \XeTeXradical_\@xa\gmu@stripchar\meaning%
            \symgmathroman\space_"221A\relax}%
4815     \fi
4816     \fi% of if 1 1.
\max 4817     \def\max{\rmopname_\relax_m{max}}%
\min 4818     \def\min{\rmopname_\relax_m{min}}%
\lim 4819     \def\lim{\rmopname_\relax_m{lim}}%
\sin 4820     \def\sin{\rmopname_\relax_o{sin}}%
\cos 4821     \def\cos{\rmopname_\relax_o{cos}}%
\tg 4822     \def\tg{\rmopname_\relax_o{tg}}%
\ctg 4823     \def\ctg{\rmopname_\relax_o{ctg}}%
\tan 4824     \def\tan{\rmopname_\relax_o{tan}}%
\ctan 4825     \def\ctan{\rmopname_\relax_o{ctan}}%
4826 }% of \gmu@dogmathbase
4827 \AtBeginDocument{\gmu@dogmathbase[#1](#2)%
4828     \let\gmathbase\gmu@dogmathbase
4829 }% of atbd
4830     \not@onlypreamble\gmathbase
4831 }% of \gmathbase

```

The `\gmatbase` declaration defines a couple of `\gmath` defining commands and then launches them for the default font at begin document and becomes only that launching.

```
4837 \@onlypreamble\gmathbase
```

It's a bit tricky: if `\gmathbase` occurs first time in a document inside document then an error error is raised. But if `\gmathbase` occurs first time in the preamble, then it removes itself from the only-preamble list and redefines itself to be only the inner macro of the former itself.

```

\gmathfurther 4844 \pdef\gmathfurther{%
4851     \def\do##1##2##3{\gmu@storeifnotyet##1%
4852         \def##1{%
4853             \mathop{\mathchoice{\hbox{%
4854                 \rm
4855                 \edef\gma@tempa{\the\fontdimen8\font}%
4856                 \larger[3]%
4857                 \lower\dimexpr(\fontdimen8\font-\gma@tempa)/2_\%
4858                 \hbox{##2}}}{\hbox{%
4859                 \rm
4860                 \edef\gma@tempa{\the\fontdimen8\font}%
4861                 \larger[2]%
4862                 \lower\dimexpr(\fontdimen8\font-%
                    \gma@tempa)/2_\%
4863                 \hbox{##2}}}}%
4864                 {\mathrm{##2}}{\mathrm{##2}}}}##3}}%
4865     \iffontchar\gmath@font"2211_\_\_\_\do\sum{\char"2211}{}%
        \fi%
4866     \do\forall{\gma@quantifierhook_\rotatebox[origin=c]{%
        180}{A}}%
4867     \gmu@forallkerning
4868     }{\nolimits}%
4869     \def\gmu@forallkerning{\setbox0=\hbox{A}\setbox2=\hbox{%
        \scriptsize_x}}%

```



```

4870     \kern\dimexpr\ht2/3*2_-\wdo/2\relax}% to be able to redefine it
         when the big quantifier is Bauhaus-like.
4872     \do\exists{\rotatebox[origin=c]{180}{\gma@quantifierhook_
         E}}\nolimits%
4874     \def\do##1##2##3{\gmu@storeifnotyet##1%
4875         \def##1{##3{%
4876             \mathchoice{\hbox{\rm##2}}{\hbox{\rm##2}}%
4877             {\hbox{\rm\scriptsize##2}}{\hbox{\rm
                 \tiny##2}}}}}%
4879     \unless\iffontchar\gmath@font"2227
4880         \do\vee{\rotatebox[origin=c]{90}{<}}\mathbin%
4881     \fi
4882     \unless\iffontchar\gmath@font"2228
4883         \do\wedge{\rotatebox[origin=c]{-90}{<}}\mathbin
4884     \fi
4886     \unless\iffontchar\gmath@font"2194
4887         \if\iffontchar\gmath@font"2190_0\else1\fi
4888         \iffontchar\gmath@font"2192_0\else2\fi
4889         \do\leftrightarrow{\char"2190\kern-0,1em_
         \char"2192}\mathrel
4891     \fi\fi
4893     \def\do##1##2##3{\gmu@storeifnotyet##1%
4894         \def##1###1{##2{\hbox{%
4895             \rm
4896             \setbox0=\hbox{###1}%
4897             \edef\gma@tempa{\the\hto}%
4898             \edef\gma@tempb{\the\dpo}%
4899             ##3%
4900             \setbox0=\hbox{###1}%
4901             \lower\dimexpr(\hto_+\dpo)/2-\dpo_-(%
                 \gma@tempa+\gma@tempb)/2-\gma@tempb)_
         \box0}}}}%
4902     \do\bigl\mathopen\larger
4903     \do\bigr\mathclose\larger
4904     \do\Bigl\mathopen\largerr
4905     \do\Bigr\mathclose\largerr
4906     \do\biggl\mathopen{\larger[3]}%
4907     \do\biggr\mathclose{\larger[3]}%
4908     \do\Biggl\mathopen{\larger[4]}%
4909     \do\Biggr\mathclose{\larger[4]}%
4910     \addtotoks\everymath{%
4913     \def\do##1##2{\gmu@storeifnotyet##1%
4914         \def##1{\iffmode##2{\mathchoice
4915             {\hbox{\rm\char`##1}}{\hbox{\rm\char`##1}}%
4916             {\hbox{\rm\scriptsize\char`##1}}{\hbox{\rm\tiny%
                 \char`##1}}}}%
4917         \else\char`##1\fi}}%
4918     \do\{\mathopen
4919     \do\}\mathclose
4920     \def\={\mathbin{=}}%
4921     \def\neqb{\mathbin{\neq}}%
4922     \let\neb\neqb
4923     \def\do##1{\gmu@storeifnotyet##1%

```

```

4929     \edef\gma@tempa{%
4930         \def\@xa \@nx\cename_\@xa\gobble\string##1r%
           \endcsname{%
4931             \@nx\mathrel{\@nx##1}}}%
4932     \gma@tempa}%
4933     \do\vee_\do\wedge_\do\neg
4934     \def\fakern{\mkern-3mu}%
4935     \thickmuskip=8mu_\plus_4mu\relax
4937     \gma@gmathhook
4938 }% of \everymath.
4939 \everydisplay\everymath
4940 \ifdefined\Url
4941     \ampulexdef\Url{\let\do}\@makeother
4942     {\everymath}\let\do\@makeother}% 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.
4946 \fi% of \ifdefined\Url.
4947 }% of \def\gmathfurther.

4949 \StoreMacro\mathchar@type

\gmath 4951 \DeclareCommand\gmath{oC{\NoValue}}{%
4952     \gmathbase[#1](#2)%
4953     \gmathfurther
4954     \IfValueT{#1}{\cename_\gmathhook#1\endcsname}% this allows adding
           version-specific stuff (I first used this for Fell fonts rescued with Garamond
           Premier)
4957 }

\gmathscripts 4959 \pdef\gmathscripts{%
4960     \addtotoks\everymath{\catcode`\^=7\relax_\catcode`\_=8%
           \relax_\}%
4961     \everydisplay\everymath}

\gmathcats 4963 \pdef\gmathcats{%
4964     \addtotoks\everymath{\gmu@septify}%
4965     \everydisplay\everymath}

\quantifierhook 4967 \emptify\gma@quantifierhook
\gma@quantifierhook 4968 \def\quantifierhook#1{%
4969     \def\gma@quantifierhook{#1}}

4971 \emptify\gma@gmathhook
\gmathhook 4972 \def\gmathhook#1{\addtomacro\gma@gmathhook{#1}}

\gma@dollar 4975 \def\gma@dollar$#1${{\gmath$#1$}}%
\gma@bare 4976 \def\gma@bare#1{\gma@dollar$#1$}%
\gma@checkbracket 4977 \def\gma@checkbracket{\@ifnextchar\[%
4978     \gma@bracket\gma@bare}
\gma@bracket 4979 \def\gma@bracket\[#1\]{{\gmath\[#1\]}\@ifnextchar\par}{%
           \noindent}}

\gma 4980 \def\gma{\@ifnextchar$%
4981     \gma@dollar\gma@checkbracket}

\garamath 4987 \DeclareCommand\garamath{%
4988     O{\rm}% the font command
4989 }{%

```

Before 2009/10/19 all the stuff was added to `\everymath` which didn't work.

```

4992 \quantifierhook{\addfontfeature{OpticalSize=800}}%
\gma@arrowdash 4994 \def\gma@arrowdash{
4995     \setbox0=\hbox{\char"2192}\copy0\kern-0,6\wdo
4996     \bgcolor\rule[-\dpo]{0,6\wdo}{\dimexpr1,07\ht0+\dpo}%
         \kern-0,6\wdo}}%
\gma@gmathhook 4998 \def\gma@gmathhook{
4999     \def\do####1####2####3{\gmu@storeifnotyet####1%
5000     \def####1{####3{
\mathchoice 5001     \mathchoice{\hbox{#1####2}}{\hbox{#1####2}}%
5002     {\hbox{#1\scriptsize####2}}{\hbox{#1%
         \tiny####2}}}}}%
5003     \do\mapsto{\rule[0,4ex]{0,1ex}{0,4ex}\kern-0,05em%
5004     \gma@arrowdash\kern-0,05em\char"2192}\mathrel
5005     \do\cup{\scshape\sqcup}\mathbin
5006     \do\varnothing{\setbox0=\hbox{\gma@quantifierhook%
         \addfontfeature{Scale=1.272727}0}%
5007     \setbox2=\hbox{\char"2044}%
5008     \copy0\kern-0,5\wdo\kern-0,5\wd2\lower0,125\wdo%
         \copy2
5009     \kern0,5\wdo\kern-0,5\wd2}}% of \varnothing
5010     \do\leftarrow{\char"2190\kern-0,05em\gma@arrowdash}%
         \mathrel
5011     \do\shortleftarrow{\char"2190}\mathrel
5012     \do\rightarrow{\gma@arrowdash\kern-0,05em\char"2192}%
         \mathrel
5013     \do\shortrightarrow{\char"2192\relax}\mathrel
5014     \do\in{\gma@quantifierhook\char"0454}\mathbin
5015     \do\prec{\gma@quantifierhook
5016     \rotatebox[origin=c]{-90}{%
5017     \glyphname{u03A5.a}}}\mathrel\z added 2009/9/11
5018     }% of \gma@gmathhook
5019 }% 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 5028 \def\gmu@tempa{\kern-0,08em\penalty10000\hskiposp\relax
5029     s\penalty10000\hskiposp\relax}
5031 \protected\edef\Vs{V\gmu@tempa}
5033 \protected\edef\Ws{W\gmu@tempa}
\Wz 5035 \pdef\Wz{W\kern-0,05em\penalty10000\hskiposp\relax\z}

```

A left-slanted font

Or rather a left Italic *and* left slanted font. In both cases we sample the skewness of the `itshape` font of the current family, we reverse it and apply to `\itshape` in `\lit|shape` and `\textlit` and to `\sl` in `\lsl`. Note a slight asymmetry: `\litshape` and `\textlit` take the current family while `\lsl` and `\textlsl` the basic Roman family and basic (serif) Italic font. Therefore we introduce the `\lit` declaration for symmetry, that declaration left-slants `\it`.

I introduced them first while typesetting E. Szarzyński's *Letters* to follow his (elaborate) hand-writing and now I copy them here when need left Italic for his *Albert Camus' The Plague* to avoid using bold font.

Of course it's rather esoteric so I wrap all that in a declaration.

```

\leftslanting@ 5057 \pdef\leftslanting@{%
  \litdimen 5058 \def\litdimen{\strip@pt\fontdimen1\font_ex}%
\litcorrection 5059 \def\litcorrection{%
  5060 \ifhmode\null\nobreak\hskip\litdimen\relax\fi}%
  \litkern 5061 \def\litkern{% note it's to be used inside the left slanted font, unlike \lit|
    5064 \leavevmode\null
    5065 \kern-\litdimen\relax}%
\dilitkern 5066 \def\dilitkern{\kern\litdimen\litkern}%
  \litshape 5068 \pdef\litshape{%
    5070 \litcorrection
    5071 \itshape
    5072 \@tempdima=-2\fontdimen1\font
    5073 \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.
    5077 \litcorrection
    5078 \edef\gmu@tempa{%
    5079 \@nx\addfontfeature{FakeSlant=\strip@pt\@tempdima}}%
      when not \edefed, it caused an error, which is perfectly
      understandable.
    5082 \gmu@tempa}%
\textlit 5085 \pdef\textlit##1{%
  5086 {\litshape##1}}%
  \lit 5088 \pdef\lit{\rm\litshape}%
  \lsl 5091 \pdef\lsl{%
    5092 \litcorrection
    5093 \it
    5096 \@tempdima=-\fontdimen1\font
    5097 \litcorrection
    5098 \xdef\gmu@tempa{%
    5099 \@nx\addfontfeature{RawFeature={slant=\strip@pt%
      \@tempdima}}}%
    5100 \rm_% Note in this declaration we left-slant the basic Roman font not the
      itshape of the current family.
    5102 \gmu@tempa}%

```

Now we can redefine `\em` and `\emph` to use left Italic for nested emphasis. In Polish typesetting there is bold in nested emphasis as I have heard but we don't like bold since it perturbs homogeneous greyness of a page. So we introduce a three-cycle instead of two-: Italic, left Italic, upright.

```

\em 5110 \pdef\em{%
  5111 \ifdim\fontdimen1\font=\z@_ \itshape
  5112 \else
  5113 \ifdim\fontdimen1\font>\z@_ \litshape
  5114 \else_ \upshape
  5115 \fi
  5116 \fi}%

```

```

5119 \pdef\emph##1{%
5120   {\em##1}}%
5121 }% of \leftslanting@.
\leftslanting 5123 \pdef\leftslanting{\AtBeginDocument\leftslanting@}
5125 \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 5136 \def\romorzero#1{%
5137   \ifnum#1=0\zero\else\romannumeral#1\fi}

\fakeonum 5139 \DeclareCommand\fakeonum{%
5140   o fake bold for the digit »2« (for which emboldening improves look),
5142   >Pm the text to fake old-style numbers in.
5148 }{% I tried to use this command as a declaration but active digits are very uncom-
        forttable, e.g. you can't define macros with arguments.
5152   \gmu@if@onum{#2}{%
5153     \begingroup
5154     \edef\gmu@tempa{#2}%
5155     \makeatletter%
5156     \IfValueT{#1}{%
5157       \prependtomacro\fake@onum@ii{%
5158         \begingroup\addfontfeature{FakeBold=#1}}%
5159       \addtomacro\fake@onum@ii\endgroup
5160     }%
5161     \endlinechar\m@ne% to suppress the line end added by \scantokens,
        especially in active ^^M's scopes.
5163     \gmu@dofakeonum
5164     \@xa\scantokens\@xa{\gmu@tempa}%
5165     \endgroup
5166   }% of \gmu@if@onum 'else'.
5167 }% of \fakeonum.

\gmu@dofakeonum 5169 \def\gmu@dofakeonum{%
5170   \def\do##1{%
5171     \catcode`##1\active
5172     \scantokens{%
5173       \@xa\let\@xa##1%
5174       \csname fake@onum@\@xa\romorzero\string##1\endcsname%
        \empty}}%
5175   \do0\do1\do2\do3\do4\do5\do6\do7\do8\do9%
5176 }

5178 \def\do#1#2{%
5179   \@namedef{fake@onum@\romorzero#1}{#2}}

\gmu@tempa 5181 \def\gmu@tempa#1{%
5182   \do#1{\leavevmode
5183     \gmu@calculateslant{#1}% uses \gmu@tempa and \gmu@tempb, there-
        fore goes first. And defines \gmu@tempd.

```

```

5185     \gmu@measurewd{#1}% the width of char #1 is in \gmu@tempa without
        kerning and in \gmu@tempb with kerning.
5191     \edef\gmu@tempc{\the\fontcharht\font`#1}%
5192     \hbox□to□\gmu@tempb□{%
5193         \hss\resizebox{\gmu@tempa}%
5194         {\dimexpr\fontdimen5\font+\gmu@tempc-\fontdimen8%
        \font}%
5195         {\gmu@tempd#1}\hss}}
\gmu@measurewd 5197 \def\gmu@measurewd#1{%
5198     \edef\gmu@tempa{\the\fontcharwd\font`#1}%
5199     \settowidth{\@tempdimb}{% to preserve kerning
5200         \char`#1\char`#1\char`#1\char`#1\char`#1\char`#1%
5201         \char`#1\char`#1\char`#1\char`#1\char`#1\char`#1%
5202         \char`#1\char`#1\char`#1\char`#1\char`#1\char`#1%
5203         \char`#1\char`#1\char`#1}%
5204     \edef\gmu@tempb{\the\dimexpr(\@tempdimb-\gmu@tempa)/20}%
5205 }

5207 \gmu@tempa0□% \fake@onum@zero
5208 \gmu@tempa1□% \fake@onum@i
5209 \gmu@tempa2□% \fake@onum@ii

\gmu@tempa 5211 \def\gmu@tempa#1{%
5212     \do#1{\leavevmode
5213         \gmu@measurewd{#1}%
5214         \lower
5215         \dimexpr\fontdimen8\font-\fontdimen5\font\relax
5216         \hbox□to□\gmu@tempb□{\hss#1\hss}}%
5217 }

5219 \gmu@tempa3□% \fake@onum@iii
5220 \gmu@tempa4□% \fake@onum@iv
5221 \gmu@tempa5□% \fake@onum@v
5222 \gmu@tempa7□% \fake@onum@vii
5223 \gmu@tempa9□% \fake@onum@ix

\gmu@tempa 5225 \def\gmu@tempa#1{% to preserve pseudo-kerning in digits sequences.
5226     \do#1{\leavevmode
5227         \gmu@measurewd#1%
5228         \hbox□to□\gmu@tempb□{\hss#1\hss}}

5230 \gmu@tempa6□% \fake@onum@vi
5231 \gmu@tempa8□% \fake@onum@viii

\gmu@if@onum 5234 \protected\def\gmu@if@onum{%
5235     \edef\gmu@tempa{\@xa\meaning\the\font}%
5236     \@xa\@ifinmeaning\detokenize{+onum}\of\gmu@tempa
5237 }

    Thus \gmu@if@onum becomes a two-argument command that executes its #1 if
    there is +onum in current font specification or its #2 if +onum is absent.
    One could easily generalise \gmu@if@onum to \@if@fontfeature, i.e. to a test for
    an arbitrary font feature, probably with employing that very nice feature specification of
    fontspec, so that you could write \IfFontFeature{Numbers=OldStyle}{}{fake□
    old-style□digits}.

\gmu@getslant 5248 \pdef\gmu@getslant{% we define \gmu@tempa to the (fake) slant of current

```

```

font.
5249 \edef\gmu@tempa{\@xa\meaning\the\font\detokenize{%
      slant=0,}}%
5250 \edef\gmu@tempb{%
5251   \def\@nx\gmu@tempb###1%
5252   \detokenize{slant=%
5253   ###2,###3}%
5254 \gmu@tempb\@nil{##2}%
5255 \edef\gmu@tempa{\@xa\gmu@tempb\gmu@tempa\@nil\pt}%
5257 }

```

```

\gmu@calculateslant 5259 \def\gmu@calculateslant#1{%
5260   \gmu@getslant
5262   \edef\gmu@tempa{\the\numexpr\dimexpr\fontdimen1\font\pt+
      \gmu@tempa}% \gmu@tempa bears the number of scaled points of total
      slant(\fontdimen1\font+slant=... if present) per 1pt of #1.
5266   \edef\gmu@tempa{\the\numexpr\gmu@tempa*
5267     \numexpr\fontdimen5\font\relax/\numexpr\fontcharht%
      \font`#1
5268     \relax}% we scale the total slant of #1 by the ratio of original and scaled
      height of #1.
5271   \edef\gmu@tempd{%
5272     \the\dimexpr\gmu@tempa\sp\fontdimen1\font}% and we sub-
      tract slant-fontdimen from the scaled total slant.
5275   \ifdim\gmu@tempd=\z@\emptify\gmu@tempd
5276   \else\edef\gmu@tempd{%
5277     \@nx\addfontfeature{FakeSlant=\strip@pt\dimexpr%
      \gmu@tempd}}%
5278   \fi}

```

```

\gmu@cepstnof 5280 \DeclareCommand\gmu@cepstnof{O{\gmu@tempa}% a cs to be \xdefed the
      font specification,
5282   s% not used really,
5283   m% \fontspec token or name of feature font (Italic,Bold,SmallCaps,
      % BoldItalic),
5285   O{,\Scale=MatchLowercase}% fontspec font features (key=val)
5286 }

```

```

5287 {% \gmu@cepstnof's body
\gmu@reservedc 5288 \def\gmu@reservedc##1:##2:\@nil{%
5289   \ifx:##2:\else\RawFeature={\gmu@maybestripcomma##2,,%
      \@nil}\fi}%

```

```

\gmu@reservedd 5291 \def\gmu@reservedd##1/##2/\@nil{% to check whether font name con-
      tains / (which may not be true!)
5293   \ifx&##2&\@xa\@firstoftwo\else\@xa\@secondoftwo\fi}%

```

```

\gmu@reservede 5295 \def\gmu@reservede##1:##2:\@nil{% to check whether the font name
      contains : when it doesn't contain / .
5297   \ifx&##2&\@xa\@firstoftwo\else\@xa\@secondoftwo\fi}%
5299   \edef\gmu@reserveda{%

```

now, reserved B parses the font name and features. It uses an auxiliary reserved C because after / may be or may not be features specification.

```

5303   \@xa\@xa\@xa\gmu@reservedd\@xa\meaning\the\font//\@nil

```

```

5304     {% font name doesn't contain a slash
5305     \@xa \@xa \@xa \gmu@reservede \@xa \meaning \the \font::%
           \@@nil
5306     {% nor does it contain a colon
5307     \def \@nx \gmu@reservedb \detokenize {select \font
5308     " }###1 \detokenize { " }###2 \@nx \@@nil {%
5309     \ifx \fontspec#3%
5310     \@nx \@nx \@nx \fontspec [ \@gobble#4 \@empty ] {###1}% gobble
           a comma
5311     \else
5312     #3Font={###1} , #3Features={\@gobble #4%
           \@empty}%
5313     \fi
5314     }%
5315     }%
5316     {% no slash but there is a colon
5317     \def \@nx \gmu@reservedb \detokenize {select \font
5318     " }###1 :###2 \detokenize { " }###3 \@nx \@@nil {%
5319     \ifx \fontspec#3%
5320     \@nx \@nx \@nx \fontspec [ \@nx \gmu@reservedc###2::%
           \@nx \@@nil#4 ] {###1}%
5321     \else
5322     #3Font={###1} , #3Features={\@nx%
           \gmu@reservedc###2:: \@nx \@@nil#4}%
5323     \fi
5324     }%
5325     }%
5326     }% of 'no slash' case
5327     {% font name contains a slash
5328     \def \@nx \gmu@reservedb \detokenize {select \font
5329     " }###1 /###2 \detokenize { " }###3 \@nx \@@nil {%
5330     \ifx \fontspec#3%
5331     \@nx \@nx \@nx \fontspec [ \@nx \gmu@reservedc###2::%
           \@nx \@@nil#4 ] {###1}%
5332     \else
5333     #3Font={###1} , #3Features={\@nx%
           \gmu@reservedc###2:: \@nx \@@nil#4}%
5334     \fi
5335     }% of \gmu@reservedb
5336     }% of 'slash present' case
5337     } \gmu@reserveda
5339     \xdef#1 {\@xa \@xa \@xa \gmu@reservedb \@xa \meaning \the \font%
           \@@nil}%
5341 }%

\gmu@maybestripcomma 5343 \def \gmu@maybestripcomma#1 , , #2 \@@nil {#1}
\gmu@setbasefont      5345 \pdf \gmu@setbasefont {\@xa \let \@xa \gmu@basefont \the \font}
                    5347 \let \setbasefont \gmu@setbasefont
\gmu@calc@scale       5349 \DeclareCommand \gmu@calc@scale {%
                    5350 O {1}% a factor,
                    5351 m% number of the fontdimen
                    5352 }

```



```

5353 {\begingroup
      We ‘descale’ the current font:
5358   \gmu@cepstnof\fontspec[, \Scale=1]\gmu@tempa
5359   \@xa\let\@xa\gmu@currfont@descaled\the\font
5360   \gmu@basefont
5362   \gmu@cepstnof\fontspec[, \Scale=1]\gmu@tempa
      now also the base font is descaled.

5364   \xdef\gmu@fontscale{%
5365     \strip@pt
5367     \dimexpr\_\_1pt\_*
5368     \numexpr\dimexpr#1\fontdimen#2\font\relax\relax\_/
5369     \numexpr\fontdimen#2\gmu@currfont@descaled\relax
5370     \relax}%
5372   \endgroup}

```

Varia

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

```

\gmu@tilde 5380 \def\gmu@tilde{%
5381   \leavevmode\lower.8ex\hbox{\$, \widetilde{\mbox{\_}}\$, $}}

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

5387 \AtBeginDocument{% to bypass redefinition of \~ as a text command with vari-
      ous encodings
\texttilde 5389   \pdef\texttilde{%
5396     \@ifnextchar/{\gmu@tilde\kern-0,1667em\relax}\gmu@tilde}}

```

We prepare the proper kerning for “~/”.

The standard `\obeyspaces` declaration just changes the space’s `\catcode` to ¹³ (‘active’). Usually it is fairly enough because no one ‘normal’ redefines the active space. But we are *not* normal and we do *not* do usual things and therefore we want a declaration that not only will `\activeate` the space but also will (re)define it as the `_` primitive. So define `\gmobeyspaces` that obeys this requirement.

(This definition is repeated in `gmverb`.)

```

5408 \foone{\catcode`\_\active}%
\gmobeyspaces 5409 {\def\gmobeyspaces{\let\_\_ \catcode`\_\active}}

```

While typesetting poetry, I was surprised that sth. didn’t work. The reason was that original `\obeylines` does `\let` not `\def`, so I give the latter possibility.

```

5416 \foone{\catcode`\^^M\active}% the comment signs here are crucial.
\defobeylines 5417 {\def\defobeylines{\catcode`\^^M=13\def^^M{\par}}}
```

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

```

\dekssmallskip 5426 \def\dekssmallskip{\vskip\smallskipamount}
\undeeksmallskip 5427 \def\undeeksmallskip{\vskip-\smallskipamount}
\dekmedbigskip 5428 \def\dekmedbigskip{\vskip\glueexpr\medskipamount+
      \smallskipamount}
\dekmedskip 5429 \def\dekmedskip{\vskip\medskipamount}

```

```

\dekbigskip 5430 \def\dekbigskip{\vskip\bigskipamount}
\hfillneg 5433 \def\hfillneg{\hskip\opt\plus-1fill\relax}

```

A mark for the **TO-DO!**s:

```

\TODO 5438 \newcommand*\TODO[1][ ]{{%
5439 \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^AT_EX macro instead.

```

\twocoltoc 5474 \newcommand\twocoltoc{%
5475 \RequirePackage{multicol}%
\@starttoc 5476 \def\@starttoc##1{%
5477 \begin{multicols}{2}\makeatletter\@input{\jobname_
.##1}%
5478 \if@files\@xa\newwrite\csname_ttf##1\endcsname
5479 \immediate\openout\csname_ttf##1\endcsname%
\jobname_##1\relax
5480 \fi
5481 \@nobreakfalse\end{multicols}}
5483 \@onlypreamble\twocoltoc

```

The macro given below is taken from the multicol package (where its name is `\enough@room`). I put it in this package since I needed it in two totally different works.

```

\enoughpage 5488 \DeclareCommand\enoughpage{%
5489 Q{+-0123456789}% (optional short version (number of \baselineskips))
5490 B{2\baselineskip}% (2) optional (formerly mandatory) long version of re-
quired room on a page
5492 >is
5493 B{} \ % (3) what if the room is enough
5494 >isB{\newpage} \ % (4) what if there's too little room on a page
5495 }{%
5502 \if
5503 \ifdim\dimexpr\pagegoal-\pagetotal<\dimexpr
5504 \IfValueTF{#1}{#1\baselineskip}{#2}\relax
5505 1\else\fi
5506 \unless\ifdim\dimexpr\pagegoal-\pagetotal<\z@
5507 1\else2\fi

```

We check both whether space left on page is larger than we check *and* whether it's nonnegative (the latter happens when we are already on the next page). Therefore this test will not work properly for large values of #1 or values close to `\textwidth`.

```

5518 \@xa\@firstoftwo
5519 \else
5520 \@xa\@secondoftwo
5521 \fi
5522 {#4}{#3}%
5523 }

```

An equality sign properly spaced:

```

\equals 5532 \pdef\equals{\hunskip$ }={}\$ \ignorespaces}

```

And for the L^AT_EX's pseudo-code statements:

```
\eequals 5534 \pdef\eequals{\hunskip$ {}=={} }\ignorespaces}
\cdot 5536 \pdef\cdot{\hunskip$ {}·{} }\ignorespaces}
```

While typesetting a UTF-8 ls-R result I found a difficulty that follows: UTF-8 encoding is handled by the inputenc package. It's O.K. so far. The UTF-8 sequences are managed using active chars. That's O.K. so far. While writing such sequences to a file, the active chars expand. You feel the blues? When the result of expansion is read again, it sometimes is again an active char, but now it doesn't star a correct UTF-8 sequence.

Because of that I wanted to 'freeze' the active chars so that they would be \written to a file unexpanded. A very brutal operation is done: we look at all 256 chars' catcodes and if we find an active one, we \let it \relax. As the macro does lots and lots of assignments, it shouldn't be used in \edefs.

```
\freeze@actives 5556 \def\freeze@actives{%
5557   \count \z@\z@
5559   \@whilenum\count \z@<\@cclvi\do{%
5560     \ifnum\catcode\count \z@=\active
5561       \uccode`~=\count \z@
5562       \uppercase{\let~\relax}%
5563     \fi
5564     \advance \count \z@\@ne}}
```

A macro that typesets all 256 chars of given font. It makes use of \@whilenum.

```
\ShowFont 5570 \newcommand*\ShowFont [1] [6] {%
5571   \begin{multicols} {#1} [The \current \font (the \f@encoding\
encoding):]
5572   \parindent \z@
5573   \count \z@\m@ne
5574   \@whilenum\count \z@<\@cclv\do {
5575     \advance \count \z@\@ne
5576     \_\the\count \z@:\~\char\count \z@\par}
5577   \end{multicols}}
```

A couple of macros for typesetting liturgic texts such as psalms of Liturgia Horarum. I wrap them into a declaration since they'll be needed not every time.

```
\liturgiques 5585 \newcommand*\liturgiques [1] [red] {% Requires the color package.
5586   \gmu@RPfor {xcolor} \color%
\czerwo 5587   \newcommand*\czerwo {\small \color {#1}}% environment
\czer 5588   \newcommand {\czer} [1] {\leavevmode {\czerwo##1}}% we leave vmode
because if we don't, then verse's \everypar would be executed in
a group and thus its effect lost.
5591   \StoreMacro \*%
\* 5592   \def \* {\czer {\storedcsname {*}}}%
\+ 5593   \def \+ {\czer {+}}%
\nieczer 5594   \newcommand*\niezcer [1] {\textcolor {black} {##1}}%
5595 }
```

After the next definition you can write \gmu@RP [<options>] { <package> } { <CS> } to get the package #2 loaded with options #1 if the CS#3 is undefined.

```
\gmu@RPfor 5600 \newcommand*\gmu@RPfor [3] [] {%
5601   \ifx \relax #1 \relax \emptify \gmu@resa
\gmu@resa 5603   \else \def \gmu@resa { [#1] }%
```

```

5604 \fi
5605 \@xa\RequirePackage\gmu@resa{#2}

```

Since inside document we cannot load a package, we'll redefine `\gmu@RPfor` to issue a request before the error issued by undefined CS.

```

\gmu@RPfor 5611 \AtBeginDocument{%
5612 \renewcommand*\gmu@RPfor[3][]{%
5613 \unless\ifdefined#3%
5614 \@ifpackageloaded{#2}{}{%
5615 \typeout{^^J!_Package_`#2'_not_loaded!!!_(%
\on@line)^^J}}%
5616 \fi}}

```

It's very strange to me but it seems that `c` is not defined in the basic math packages. It is missing at least in the *Symbols* book.

```

\continuum 5622 \pprovide\continuum{%
5623 \gmu@RPfor{eufrak}\mathfrak\ensuremath{\mathfrak{c}}}

```

And this macro I saw in the *ltugproc* document class and I liked it.

```

\iteracro 5627 \def\iteracro{%
\acro 5628 \pdef\acro##1{%
5629 \begingroup
5630 \acropresetting
5631 \gmu@acrospace##1_\gmu@acrospace
5632 \endgroup
5633 }%
5634 }

```

```

5636 \emptify\acropresetting
5638 \iteracro

```

```

\gmu@acrospace 5640 \def\gmu@acrospace#1_#2\gmu@acrospace{%
5641 \gmu@acroinner#1\gmu@acroinner
5642 \ifx\relax#2\relax\else
5643 \space
5644 \afterfi{\gmu@acrospace#2\gmu@acrospace}% when #2 is nonempty,
it is ended with a space. Adding one more space in this line resulted in
an infinite loop, of course.
5648 \fi}

```

```

\gmu@acroinner 5651 \def\gmu@acroinner#1{%
5652 \ifx\gmu@acroinner#1\relax\else
5653 \ifcat_a\@nx#1\relax%
5654 \ifnum`#1=\ucode`#1%
5655 {\acrocore{#1}}%
5656 \else{#1}% tu byto \smallerr
5657 \fi
5658 \else#1%
5659 \fi
5660 \afterfi\gmu@acroinner
5661 \fi}

```

We extract the very thing done to the letters to a macro because we need to redefine it in fonts that don't have small caps.

```
\acrocore 5665 \pdef\acrocore{\smaller\% was: \scshape\lowercase
5666 }
```

Since the fonts I am currently using do not support required font feature, I skip the following definition.

```
\IMHO 5671 \def\IMHO{\acro{IMHO}}
```

```
\AKA 5672 \def\AKA{\acro{AKA}}
```

```
\usc 5674 \pdef\usc#1{{\addfontfeature{Letters=UppercaseSmallCaps}#1}}
```

```
\uscacro 5676 \def\uscacro{\let\acro\usc}
```

Probably the only use of it is loading `gmdocc.cls` ‘as second class’. This command takes first argument optional, options of the class, and second mandatory, the class name. I use it in an article about `gmdoc`.

```
\secondclass 5694 \def\secondclass{%
```

```
\ifSecondClass 5695 \newif\ifSecondClass
```

```
5696 \SecondClasstrue
```

```
5697 \@fileswithoptions\@clsextension}% [outeroff,gmeometric]{gm|
docc} it’s loading gmdocc.cls with all the bells and whistles except the error
message.
```

Cf. *The T_EXbook* ex. 11.6.

A line from L^AT_EX:

```
%\check@mathfonts\fontsize\sf@size\z@\math@fontsfalse\selectfont
```

didn’t work as I would wish: in a `\footnotesize`’s scope it still was `\scriptsize`, so too large.

```
\gmu@dekfraccsimple 5709 \def\gmu@dekfraccsimple#1/#2{\leavevmode\kern.1em
```

```
5710 \raise.7ex\hbox{%
```

```
5711 \gmu@fracfontsetup#1}\gmu@numerator\kern
```

```
5712 \dekfracslash\gmu@denominator\kern
```

```
5714 {%
```

```
5715 \gmu@fracfontsetup#2}%
```

```
5716 \if@gmu@mmhbox\egroup\fi}
```

```
\gmu@fracfontsetup 5718 \def\gmu@fracfontsetup{%
```

```
5719 \smaller[3]\addfontfeature{FakeBold=1}}
```

```
\dekfraccsimple 5722 \def\dekfraccsimple{%
```

```
5723 \let\dekfracc@args\gmu@dekfraccsimple
```

```
5724 }
```

```
\dekfracslash 5725 \@ifXeTeX{\def\dekfracslash{\char"2044\}}
```

```
\dekfracslash 5726 {\def\dekfracslash{/}}\% You can define it as the fraction
slash, \char"2044\
```

```
5728 \dekfraccsimple
```

A macro that acts like `\,` (thin and unbreakable space) except it allows hyphenation afterwards:

```
\ikern 5736 \newcommand*\ikern{\, \penalty\@M\hskip\z@skip\relax}
```

And a macro to forbid hyphenation of the next word:

```
\nohy 5740 \pdef\nohy{\leavevmode\kernosp\relax}
```

```
\yeshy 5741 \pdef\yeshy{\leavevmode\penalty\@M\hskip\z@skip}
```

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

`\include not only .tex's`

`\include` modified by me below lets you to include files of any extension provided that extension in the argument.

If you want to `\include` a non-.tex file and deal with it with `\includeonly`, give the latter command full file name, with the extension that is.

```
\gmu@gettext 5756 \def\gmu@gettext#1.#2\@nil{%
5757   \def\gmu@filename{#1}%
5758   \def\gmu@fileext{#2}}

5760 \def\include#1{\relax
5761   \ifnum\@auxout=\@partaux
5762   \@latex@error{\string\include\space cannot be nested}\@eha
5763   \else\@include#1\fi}

\@include 5765 \def\@include#1{%
5766   \gmu@gettext#1.\@nil
5768   \ifx\gmu@fileext\empty\def\gmu@fileext{tex}\fi
5769   \clearpage
5770   \if@filesw
5771     \immediate\write\@mainaux{\string\@input{%
5772       \gmu@filename.aux}}%
5773   \fi
5774   \@tempwattrue
5775   \if@partsw
5776     \@tempwafalse
5777     \edef\reserved@a{#1}%
5778     \@for\reserved@a:=\@partlist\do{%
5779       \ifx\reserved@a\reserved@b\@tempwattrue\fi}%
5780   \fi
5781   \if@tempswa
5782     \let\@auxout\@partaux
5783     \if@filesw
5784       \immediate\openout\@partaux\gmu@filename.aux
5785       \immediate\write\@partaux{\relax}%
5786     \fi
5787     \@input{\gmu@filename.\gmu@fileext}%
5788     \inclasthook
5789     \clearpage
5790     \@writeckpt{\gmu@filename}%
5791     \if@filesw
5792       \immediate\closeout\@partaux
5793     \fi
5794   \else

If the file is not included, reset \@include \deadcycles, so that a long list of non-
included files does not generate an 'Output loop' error.

5797   \deadcycles\z@
5798   \@nameuse{cp@\gmu@filename}%
5799   \fi
5800   \let\@auxout\@mainaux}

\whenonly 5803 \newcommand\whenonly[3]{%
\gmu@whonly 5804   \def\gmu@whonly{#1,}%
```

```
5805 \ifx\gmu@whonly\@partlist\afterfi{#2}\else\afterfi{#3}\fi}
```

I assume one usually includes chapters or so so the last page style should be closing.

```
\inclasthook 5809 \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.

```
\filepart 5818 \def\filepart#1{\relax
5819 \ifnum\@auxout=\@partaux
5820 \@latex@error{\string\filepart\space cannot be nested}%
\@eha
5821 \else\afterfi{\@filepart#1}\fi}
```

```
\@filepart 5823 \def\@filepart#1{%
5824 \clearpage
5825 \edef\gmu@filepartname{#1}% we'll use it later
5826 \if@filesw
5827 \immediate\write\@mainaux{\string\@input{#1.aux}}%
5828 \fi
5829 \@tempswatruer
5830 \if@partsw
5831 \@tempswafalse
5832 \@for\reserved@a:=\@partlist\do{%
5833 \ifx\reserved@a\gmu@filepartname\@tempswatrue\fi}%
5834 \fi
5835 \if@tempswa
5836 \let\@auxout\@partaux
5837 \if@filesw
5838 \immediate\openout\@partaux\#1.aux
5839 \immediate\write\@partaux{\relax}%
5840 \fi
5841 \@xa\@firstoftwo
5843 \else
```

If the file is not included, reset `\@include\deadcycles`, so that a long list of non-included files does not generate an 'Output loop' error.

```
5847 \deadcycles\z@
5848 \@nameuse{cp@\gmu@filepartname}%
5849 \let\@auxout\@mainaux
5850 \@xa\@secondoftwo
5851 \fi
5852 {\iftrue}%
5853 {\let\endfilepart\fi
5854 \csname gm@skipped@#1\endcsname
5855 \def\next{\RestoreMacro\endfilepart
5856 \@ifnextchar\bgroup{\show\NextBgroup\@gobble}{}}%
5857 \@xa\next\iffalse}%
5858 }
```

```
\endfilepart 5861 \DeclareCommand\endfilepart{b}{% Note the argument is not used really.
Maybe later we'll use it for checking of proper matching. Or maybe not.
```

```

5863 \inclasthook
5864 \clearpage
5865 \@writeckpt {\gmu@filepartname}%
5866 \if@filesw
5867 \immediate\closeout \@partaux
5868 \fi
5869 \fi% this \fi closes \Iftrue put by line 5841.
5870 \let \@auxout \@mainaux
5871 }
5873 \StoreMacro \endfilepart
\nofileparts 5875 \def \nofileparts {%
5876 \let \filepart \@gobble
\endfilepart 5877 \DeclareCommand \endfilepart {b} {}%
5878 }

```

Fix of including when fontspec is used

The fontspec package creates counters for font families. If a fontspec command is used in a part of a document and then such a part is skipped, an error occurs ‘No counter zf@fam@... defined’. Now we fix that by ensuring all the counters are defined before they are set.

Note it’s a draft version which doesn’t support resetting of one counter within another.

```

\includecountfix 5890 \def \includecountfix {%
  \@wckptelt 5891 \def \@wckptelt##1 {%
5892 \immediate\write \@partaux {%
5893 \providecounter {##1}% to provide the font counters defined in parts
of the document.
5896 \string\setcounter {##1} {\the \@nameuse {c@##1}}}%
5897 }
\providecounter 5899 \pdef \providecounter#1 {%
#1 5900 \unless \ifcsname _c@#1 \endcsname \newcounter {#1} \fi}

```

Faked small caps

```

\gmu@scapLetters 5905 \def \gmu@scapLetters#1 {%
5906 \ifx#1\relax\relax\else% two \relaxes to cover the case of empty #1.
5907 \ifcat _a \@nx#1\relax
5908 \ifnum\the\lccode`#1=`#1\relax
5909 {\fakescapscore\MakeUppercase {#1}}% not Plain \uppercase
because that works bad with inputenc.
5911 \else#1%
5912 \fi
5913 \else#1%
5914 \fi%
5915 \@xa\gmu@scapLetters
5916 \fi}%
\gmu@scapSpaces 5918 \def \gmu@scapSpaces#1_#2\@@nil {%
5919 \ifx#1\relax\relax
5920 \else\gmu@scapLetters#1\relax
5921 \fi

```



```

5922 \ifx#2\relax\relax
5923 \else\afterfi{\_ \gmu@scapSpaces#2\@@nil}%
5924 \fi}
\gmu@scapss 5926 \def\gmu@scapss#1\@@nil{\def~{\nobreakspace}}%
\nobreakspace 5927 \gmu@scapSpaces#1\_ \@@nil}}% %\_ \def\\{\newline}}\relax adding
redefinition of \\ caused stack overflow. Note it disallows hyphenation
except at \-.

\fakescaps 5931 \pdef\fakescaps#1{\gmu@scapss#1\@@nil}}
5933 \let\fakescapscore\gmu@scalematchX
Experimente z akcentami patrz no3.tex.

\tinycae 5936 \def\tinycae{\tiny\AE}}% to use in \fakescaps[\tiny]{...}
5938 \RequirePackage{calc}
wg \zf@calc@scale pakietu fontspec.

5942 \@ifpackageloaded{fontspec}{%
\gmu@scalar 5943 \def\gmu@scalar{1.0}%
\zf@scale 5944 \def\zf@scale{}%
\gmu@scalematchX 5945 \def\gmu@scalematchX{%
\gmu@scalar 5946 \begingroup
5947 \ifx\zf@scale\empty\def\gmu@scalar{1.0}%
5948 \else\let\gmu@scalar\zf@scale\fi
5949 \setlength\@tempdima{\fontdimen5\font}% 5—ex height
5950 \setlength\@tempdimb{\fontdimen8\font}% 8—X1TeX synthesised
uppercase height.
5952 \divide\@tempdimb\_by1000\relax
5953 \divide\@tempdima\_by\_ \@tempdimb
5954 \setlength{\@tempdima}{\@tempdima*\real{\gmu@scalar}}%
5955 \gm@ifundefined{fakesc@extrascale}{\}%
5956 \setlength{\@tempdima}{\@tempdima*\real{\%
\fakesc@extrascale}}}%
5957 \@tempcnta=\@tempdima
5958 \divide\@tempcnta\_by\_ 1000\relax
5959 \@tempcntb=-1000\relax
5960 \multiply\@tempcntb\_by\_ \@tempcnta
5961 \advance\@tempcntb\_by\_ \@tempdima
5962 \xdef\gmu@scscale{\the\@tempcnta.%
5963 \ifnum\@tempcntb<100\_0\fi
5964 \ifnum\@tempcntb<10\_0\fi
5965 \the\@tempcntb}%
5967 \endgroup
5968 \addfontfeature{Scale=\gmu@scscale}%
5969 }}{\let\gmu@scalematchX\smallerr}

\fakescextrascale 5971 \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 5977 \newcommand*\wyzejnizej[1]{%
5978 \edef\gmu@tempa{\gm@ifundefined{r@#1}{\arabic{page}}}{%

```

```

5979     \@xa \@xa \@xa \@secondoftwo \csname _r@#1 \endcsname } }%
5980 \ifnum \gmu@tempa < \arabic {page} \relax _wy \.zej \fi
5981 \ifnum \gmu@tempa > \arabic {page} \relax _ni \.zej \fi
5982 \ifnum \gmu@tempa = \arabic {page} \relax _ \@xa \ignorespaces \fi
5983 }

```

luzniej and napapierki—environments used in page breaking for money

The name of first of them comes from Polish typesetters’ phrase “rozbijać [skład] na papierki”—‘to broaden [leading] with paper scratches’.

```

\napapierkistretch 5993 \def \napapierkistretch {0,3pt}% It’s quite much for 11/13pt leading.
\napapierkicore    5995 \def \napapierkicore {\advance \baselineskip%
                    5996   by _optplus \napapierkistretch \relax}
                    6003 \DeclareEnvironment {napapierki} {s} {%
                    6004   \par \IfValueT {#1} {\global
                    6005     }%
                    6006   }%
                    6007   \napapierkicore}
                    6008 {%
                    6009   \par
                    6010   \IfValueT {#1} {\global \baselineskip=1 \baselineskip \relax
                    6011     }%
                    6012 }% so that you can use \napapierki* >... \endnapapierki* in interlacing envi-
                        ronments.

```

```

\gmu@luzniej 6017 \newcount \gmu@luzniej

```

```

\luzniejcore 6019 \newcommand* \luzniejcore [1] [1] {%
6020   \advance \gmu@luzniej \@ne% We use this count to check whether we open
                        the environment or just set \looseness inside it again.
6021   \ifnum \gmu@luzniej = \@ne _ \multiply \tolerance _by _2 _ \fi
6022   \looseness=#1 \relax}

```

After `\begin{luzniej}` we may put the optional argument of `\luzniejcore`

```

luzniej 6027 \newenvironment* {luzniej} {\par \luzniejcore} {\par}

```

The starred version sets `\looseness` in `\everypar`, which has its advantages and disadvantages.

```

luzniej* 6032 \newenvironment* {luzniej*} [1] [1] {%
6033   \multiply \tolerance _by _2 \relax
6034   \everypar {\looseness=#1 \relax}} {\par}

```

```

\nawj 6036 \newcommand* \nawj {\kern 0.1em \relax}% a kern to be put between paren-
                        theses and letters with descendants such as j or y in certain fonts.

```

The original `\pauza` of polski has the skips rigid (one is even a kern). We make the skips flexible. Moreover, our `\pauza` begins with `\ifhmode` to be usable also at the beginning of a line where it marks a part of a dialogue.

```

\pauza@skipcore 6045 \def \pauza@skipcore {\hskip 0.2em _plus 0.1em \relax
6046   \pauzacore
6047   \@ifnextchar , {% 2009/11/22 added a special case of a comma following
                        pauza
6048   } {\hskip 0.2em _plus 0.1em \relax \ignorespaces}}%
\ppauza@skipcore 6050 \def \ppauza@skipcore {\unskip \penalty 10000 \hskip 0.2em _
                        plus 0.1em \relax}

```

```

6051         \ppauza@dash\hskip.2em\pluso.1em\ignorespaces}
6054 \AtBeginDocument{%
\pauza 6055   \pdef \pauza{%
6056     \ifhmode
6057         \unskip\penalty10000
6058         \hskipo.2em\pluso.1em\relax
6059         \pauzacore\hskip.2em\pluso.1em\relax\ignorespaces%
6060     \else
6061         \pauzadial
6062     \fi}%

```

According to *Instrukcja technologiczna. Skład ręczny i maszynowy* the dialogue dash (in Polish) should be followed by a rigid hskip of ½em.

```

\pauzadial 6067   \pdef \pauzadial{%
6068     \leavevmode \pauzacore \penalty10000 \hskipo,5em%
        \ignorespaces}

```

And a version with no space at the left, to begin a \noindent ed paragraph explaining e.g. a quotation:

```

\lpauza 6072   \pdef \lpauza{%
6073     \leavevmode
6074     \pauzacore\hskip.2em\pluso.1em\ignorespaces}%

```

We define \ppauza as an en dash surrounded with thin stretchable spaces and sticking to the upper line or bare but discretionary depending on the next token being space₁₀. Of course you'll never get such a space after a literal CS so an explicit \ppauza will always result with a bare discretionary en dash, but if we \let-\ppauza...

```

\ppauza 6083   \pdef \ppauza{%
6084     \ifvmode\PackageError{gmutils}{%
6085     command\backslashppauza(en dash) not intended for
        vmode.}%
6086     Use\backslashppauza(en dash) only in number and
        numeral ranges.}%
6087     \else
6088         \unskip\discretionary
6089         {\ppauza@dash}{\ppauza@dash}{\ppauza@dash}%
6090     \fi}%
6091 }% of at begin document

6093 \ifdefined\XeTeXversion
6095 \AtBeginDocument{% to be independent of moment of loading of polski.
\l- 6096   \pdef \-{%
6097     \ifhmode
6098         \unskip\penalty10000
6099         \afterfi{%
6100             \@ifnextspace{\pauza@skipcore}%
6101             {\@ifnextchar,{\pauza@skipcore}% a special case of comma added
                2009/11/22
6102             {\@ifnextMac{\pauza@skipcore}%
6103             {\pauzacore\penalty\hyphenpenalty\hskip%
                \z@skip}}}%
6104         }% of \afterfi's argument
6105     \else

```

According to *Instrukcja technologiczna. Skład ręczny i maszynowy* the dialogue dash should be followed by a rigid hskip of ½ em.

```
6109     \leavevmode \pauzacore \penalty10000 \hskip0,5em%
        \ignorespaces
6110     \fi}%
```

The next command's name consists of letters and therefore it eats any spaces following it, so \@ifnextspace would always be false, therefore we don't use it.

```
\- 6114     \pdef \- {%
6115         \ifvmode \PackageError {gmutils} {%
6116             command \backslash ppauza (en dash) not intended for
                vmode. } {%
6117             Use \backslash ppauza (en dash) only in number and
                numeral ranges. }%
6118     \else
6119         \afterfi {%
6120             \@ifnextspace {\ppauza@skipcore} {%
6121                 \@ifnextMac \ppauza@skipcore
6122                 {\unskip \discretionary
6123                     {\ppauza@dash} {\ppauza@dash} {\ppauza@dash}}}%
6124             }%
6125     \fi
6126     }%
\emdash 6128     \def \emdash {\char` \-}
6129     }% of at begin document
```

```
\longpauza 6131 \def \longpauza {\def \pauzacore {-} }
\pauzacore 6132 \longpauza
\shortpauza 6133 \def \shortpauza {%
\pauzacore 6134     \def \pauzacore {\hbox{-\kern,23em\relax\llap{-}}}%
\ppauza@dash 6135 \def \ppauza@dash {-}%

6138 \else % not XeTeX
\longpauza 6139 \def \longpauza {\def \pauzacore {---} }
\pauzacore 6140 \longpauza
\shortpauza 6141 \def \shortpauza {%
\pauzacore 6142     \def \pauzacore {--\kern,23em\relax\llap{--}}}%
\ppauza@dash 6143 \def \ppauza@dash {--}%

6146 \fi% of if XeTeX.
```

If you have all the three dashes on your keyboard (as I do), you may want to use them for short instead of \pauza, \ppauza and \dywiz. The shortest dash is defined to be smart in math mode and result with –.

```
6154 \ifdefined \XeTeXversion
6155 \foone {\catcode`-\active \catcode`-\active \catcode`-\active} {%
        %
\adashes 6156     \def \adashes {\AtBeginDocument \adashes}% because \pauza is defined
                at begin document.
\adashes 6158     \AtBeginDocument {\def \adashes {%
\- 6159         \catcode`-\active \def{-\}%
\- 6160         \catcode`-\active \def{-\}%
6162         \addtomacro \dospecials {\do \- \do \-}%
6163         \addtomacro \@sanitize {\@makeother \- \@makeother \-}%
6164         \addtomacro \gmu@septify {\do \-13 \do \-13 \relax}%
```

```

6165 }}}
6166 \else
6167 \relaxen\adashes
6168 \fi

```

The hyphen shouldn't be active IMHO because it's used in T_EX control such as `\hskip-2pt`. Therefore we provide the `\ahyphen` declaration reluctantly, because sometimes we need it and always use it with caution. Note that my active hyphen in vertical and math modes expands to `-12`.

```

\gmu@dywiz 6177 \def\gmu@dywiz{\ifmmode-\else
6178   \ifvmode-\else\afterfifi\dywiz\fi\fi}%
6180 \foone{\catcode`-\active}{%  aktywnej diefis aktywny dywiz active hyphen
\ahyphen 6181   \def\ahyphen{\let-\gmu@dywiz\catcode`-\active}}

```

To get current time. Works in ϵ -T_EXs, including X_YT_EX. `\czas` typesets 8.17 and `\czas[:]` typesets 8:17.

```

\czas 6186 \newcommand*\czas[1][.]{%
6187   \the\numexpr(\time-30)/60\relax#1%
6188   \@tempcnta=\numexpr\time-(\time-30)/60*60\relax
6189   \ifnum\@tempcnta<10\fi\the\@tempcnta}
6192 \@ifXeTeX{%
\textbullet 6193   \pdef\textbullet{%
6194     \iffontchar\font"2022\char"2022\else\ensuremath{%
        \bullet}\fi}%
\glyphname 6198   \pprovide\glyphname#1{%
6200     \XeTeXglyph\numexpr\XeTeXglyphindex"#1"\relax\relax}% since
        XeTeX ... \numexpr is redundant.
6202 }
\textbullet 6203 {\def\textbullet{\ensuremath{\bullet}}}
tytulowa 6205 \newenvironment*{tytulowa}{\newpage}{\par\thispagestyle{%
        empty}\newpage}

```

To typeset peoples' names on page 4 (the editorial page):

```

\nazwired 6208 \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`).

```

\polskadata 6222 \pdef\polskadata{%
\gmu@datefsl 6223   \DeclareCommand\gmu@datefsl{%
6224     Q{0123456789\bggroup}>iT{/-\} \quad (1) year
6225     Q{0123456789\bggroup}>iT{/-\} \quad (2) month
6226     Q{0123456789\bggroup} \quad (3) day
6227     T{,} \quad K{##1\gmu@datefsl} \quad (4, 5) additional stuff after comma
6228   }{%
6229     \IfValueF{##2}{\PutIfValue{##3}}%
6230     \IfValueT{##2}{%

```

```

6231      \@tempcnta=0##3\relax\the\@tempcnta
6232      \ifcase##2\relax\or\_\_stycznia\or\_\_lutego%
6233      \or\_\_marca\or\_\_kwietnia\or\_\_maja\or\_\_czerwca\or\_\_
        lipca\or\_\_sierpnia%
6234      \or\_\_wrzeźnia\or\_\_października\or\_\_listopada\or\_\_
        grudnia\else
6235      {}%
6236      \fi}%
6237      \IfValueT{##1}{\space\_\_##1}%
6238      \PutIfValue{##4}\IfValueT{##5}{\_\_##5}%
6239      }% of \gmu@datefsl.
6240      }% of \polskadata

```

```
6242 \polskadata
```

For documentation in English:

```

\englishdate 6245 \pdef\englishdate{%
\gmu@datefsl 6246 \DeclareCommand\gmu@datefsl{%
6247     Q{0123456789\bgroup}>iT{/-\_\_}% (1) year
6248     Q{0123456789\bgroup}>iT{/-\_\_}% (2) month
6249     Q{0123456789\bgroup}\_\_}% (3) day
6250     T{,}\_\_K{##1\gmu@datefsl}\_\_}% (4,5) additional stuff after comma
6251 }{%
6252     \IfValueF{##2}{\PutIfValue{##3}}%
6253     \IfValueT{##2}{%
6254         \ifcase##2\relax\or\_\_January\or\_\_February%
6255         \or\_\_March\or\_\_April\or\_\_May\or\_\_June\or\_\_July\or\_\_
            August%
6256         \or\_\_September\or\_\_October\or\_\_November\or\_\_December%
            \else
6257         {}%
6258         \fi}%
6259     \space
6260     \@tempcnta=##3\relax\the\@tempcnta,
6261     \IfValueT{##1}{\_\_##1}%
6262     \PutIfValue{##4}\IfValueT{##5}{\_\_##5}%
6263     }% of \gmu@datefsl.
6264 }%

```

Dates for memoirs to be able to typeset them also as diaries.

```

\ifdate 6269 \newif\ifdate
\bidate 6271 \pdef\bidate#1{%
6272     \gmu@datefsl#1\gmu@datefsl
6273 }

\linedate 6275 \pdef\linedate{\gm@ifstar\linedate@@\linedate@}
\linedate@@ 6276 \pdef\linedate@@#1{\linedate@{--{}{}}#1}}
\linedate@ 6277 \pdef\linedate@#1{\par
6278     \linedate@hook{#1}%
6279     \ifdate\addvspace{\dateskipamount}%
6280     \possvfil% if we put it before \addvspace, the v-space is always added.
6281     \date@line{\footnotesize\itshape\_\_bidate{#1}}%
6282     \nopagebreak
6283     \else% %\ifnum\arabic{dateinsection}>0\dekbigskip\fi

```

```

6284     \addvspace{\bigskipamount}\possvfil
6285     \fi}% end of \linedate.

6287 \let \linedate@hook \@gobble

6289 \let \dateskipamount \medskipamount

\rdate 6291 \pdef \rdate {\let \date@line \rightline \linedate}

\date@left 6294 \def \date@left#1 {\par {%
6295     \raggedright#1%
6296     \leftskip \z@skip
6301     \@@par}}%

\ldate 6303 \pdef \ldate {%
6305     \let \date@line \date@left
6306     \linedate}

\runindate 6308 \newcommand* \runindate [1] {%
6309     \paragraph {\footnotesize \itshape \gmu@datef#1 \gmu@datef}%
6310     \stepcounter {dateinsection}}

        I'm not quite positive which side I want the date to be put to so let's let for now and
        we'll be able to change it in the very documents.

6313 \let \thedata \ldate

\zwrobcy 6316 \pdef \zwrobcy#1 {\emph {#1}} \ostinato, allegro con moto, garden party etc.,
        także komplement

\tytul 6319 \pdef \tytul#1 {\emph {#1}}

        Maszynopsis w świecie justowanym zrobi delikatną chorągiewkę. (The maszynopsis
        environment will make a delicate ragged right if called in a justified world.)

maszynopsis 6325 \newenvironment {maszynopsis} [1] [] {#1 \ttfamily
6326     \hyphenchar \font=45 \relax% this assignment is global for the font.
6327     \@tempkipa=\glueexpr \rightskip+\leftskip \relax
6328     \ifdim \gluestretch \@tempkipa=\z@
6329     \tolerance900
        it worked well with tolerance = 900.
6331     \advance \rightskip by \z@ plus 0,5em \relax \fi
6332     \fontdimen3 \font=\z@% we forbid stretching spaces...
        % \fontdimen4 \font=\z@ but allow shrinking them.
6334     \hyphenpenalty 0% not to make TEX nervous: in a typewriting this marvel-
        lous algorithm of hyphenation should be turned off and every line broken
        at the last allowable point.

6337     \StoreMacro \pauzacore
\pauzacore 6338 \def \pauzacore {-\rlap {\kern -0,3em -}}%
6339 } {\par}

\justified 6343 \pdef \justified {%
6344     \leftskip=1 \leftskip% to preserve the natural length and discard stretch
        and shrink.
6346     \rightskip=1 \rightskip
6347     \parfillskip=1 \parfillskip
6348     \advance \parfillskip by \osp plus 1fil \relax
6350     \let \\ \@normalcr}

```

To conform Polish recommendation for typesetting saying that a paragraph's last line leaving less than `\parindent` should be stretched to fill the text width:

```
\fullpar 6355 \DeclareCommand\fullpar{%
6356   T{+-}%
6357   Q{+-0123456789}□% optional looseness (most probably negative)
6358 }{%
6359   \begingroup
6360   \IfValueT{#1}{\looseness=#1\IfValueTF{#2}{#2}{1}\relax
6361     \multiply\tolerance□by□\tw@
6362   }%
6363   \fullparcore
6364   \par
6365   \endgroup}
```

```
\fullparcore 6367 \pdef\fullparcore{%
6368   \hunskip
6369   \parfillskip\z@skip}
```

To conform Polish recommendation for typesetting that says that the last line of a paragraph has to be `2\parindent` long at least. The idea is to set `\parfillskip` naturally rigid and long as `\textwidth-2\parindent`, but that causes non-negligible shrinking of the inter-word spaces so we provide a declaration to catch the proper glue where the `parindent` is set (e.g. in footnotes `parindent` is 0pt)

```
\twoparinit 6379 \newcommand*\twoparinit{% the name stands for 'last paragraph line's length
              minimum two \parindent.
\twopar@defts 6381   \def\twopar@defts{%
6382     \hsize-\leftskip-\rightskip-\fontcharwd\font`...}%
\twopar@atleast 6383   \def\twopar@atleast{2\@parindent}%
\twopar 6384   \DeclareCommand\twopar{%
6385     T{+-}% (1) you can specify loosening the paragraph by one only by typing
              single + and tightening by one by typing single -.
6387     Q{+-0123456789}% (2)
6388     A{\twopar@atleast}% (3)
6389     >iT{\cipolagwa}}{%
6390     \begingroup
6391     \IfValueT{##1}{%
6392       \looseness=##1\IfValueTF{##2}{##2}{1}\relax
6393       \multiply\tolerance□by2
6394     }%
6395     \twoparcore<##3>%
6396     \par
6397     \endgroup
6398   }% of \twopar.
6400   \ifdefined\XeTeXversion
\twoparcore 6401     \DeclareCommand\twoparcore{%
6402       A{\twopar@atleast}>iT{\cipolagwa}}{%
6403         \hunskip□% it's O.K. it's in a group, it'll work anyway.
6404         \edef\gmu@tempa{\the\dimexpr\twopar@defts-##1\relax}%
6405         \parfillskip=\glueexpr\gmu@tempa□minus□\gmu@tempa
6406         \relax% to delimit \glueexpr.
6407         \relax% to delimit the assignment.
6408       }%
6409     \else□% not XeTeX—doesn't use \fontcharwd.
```



```

\twoparc core 6410 \DeclareCommand\twoparc core{%
6411 A{\twoparc@default}>iT{\cipolagwa}}{%
6412 \hunskip\parindent it's O.K. it's in a group, it'll work anyway.
6413 {\setboxo=\hbox{\dots}}%
6414 \xdef\gmu@tempa{\the\wdo}}%
6415 \edef\gmu@tempa{%
6416 \the\dimexpr\hsize-\leftskip-\rightskip
6417 -\gmu@tempa-2\@parindent\relax}%
6418 \parfillskip=\glueexpr\gmu@tempa\minus\gmu@tempa
6419 \relax% to delimit\glueexpr.
6420 \relax% to delimit the assignment.
6421 }%
6422 \fi

6424 \AtBeginDocument{%
6425 \unless\ifdefined\@parindent
\twoparc core 6426 \newskip\@parindent
6427 \@parindent=\parindent
6428 \fi
\restoreparindent 6429 \def\restoreparindent{\parindent\@parindent}%
6430 }% of\AtBeginDocument.
6431 }% of\twoparcinit.

```

For dati under poems

Or explanations under results of time.

```

\wherncore 6439 \DeclareCommand\wherncore{om}{%
% [#1] optional value of\hskip of (left) indent of the parbox. If absent,
% parbox is aligned right;
% [#2] optional text for the datum parbox.
6445 \IfValueTF{#1}{\leftline{%
6446 \whernfont
6447 \hskip#1\relax\parbox
6448 {\dimexpr\textwidth-\leftskip-\rightskip-#1}%
6449 {#2}}% of\parbox,
6450 }% of\leftline,
6451 }% ofValueT{#1}.
6452 {% ValueF{#1}:
6453 \rightline
6454 {\whernfont
6455 \whern@parbox{#2}}%
6456 }% of\rightline,
6457 \setprevdepth
6458 }% ofValueF{#1},
6459 }% of\wherncore.

\whern@parbox 6462 \DeclareCommand\whern@parbox{%
6463 S{\leftskip\rightskip}% horizontal alignment of resulting box (the side to
be ragged)
6465 O{t}\parindent% vertical alignment of parbox
6466 >is\parindent% separator
6467 O{0,7666\textwidth}\parindent% (3) width of parbox
6468 m\parindent% (4) parbox contents
6469 }%

```

```

% #1 S the skip of the ragged side,
% #2 S the \parbox's contents.
6474 \parbox[#2]{#3}{%
6475 \IfValueTF{#1}{#1}{\leftskip}=osp\plus\textwidth
6476 \parfillskiposp\relax
6477 \let\\\linebreak
6478 \disobeylines
6479 \whernfont#4\unskip\strut\endgraf
6480 \getprevdepth
6481 }% of \parbox,
6482 }% of \whern@parbox.

```

```

\whern 6484 \def\whern{%
6485 \endgraf\nopagebreak
6486 \gm@ifstar{\wherncore}%
6487 {\vskip\whernskip\wherncore}}
6489 \let\whernfont\footnotesize

```

```

\whernskip 6491 \newskip\whernskip
6492 \whernskip2\baselineskip\minus2\baselineskip\relax

```

```

\whernup 6494 \DeclareCommand\whernup{%
6495 o% a vskip before
6496 >is% separating star (ignored)
6497 o% (2) custom width of parbox
6498 >Pm}{\par
6499 \IfValueT{#1}{\vskip#1\relax}%
6500 \leftline{%
6501 \IfValueTF{#2}{\whern@parbox\rightskip[b][#2]}%
6502 {\whern@parbox\rightskip[b]}%
6503 {#3}%
6504 }%
6505 \setprevdepth
6506 \nopagebreak\relax
6507 \@ifenvir{quote}{\noindent\ignorespaces}{}}

```

Thousand separator

```

\thousep 6513 \pdef\thousep#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.

```

6517 \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.
6520 \gmu@thou@fiver{#1}{% if more than five digits:
6521 \emptify\gmu@thou@put
6522 \relaxen\gmu@thou@o\relaxen\gmu@thou@i\relaxen%
\gmu@thou@ii
6523 \@tempcnta\z@
6524 \gmu@thou@putter#1\gmu@thou@putter
6525 \gmu@thou@put
6526 }}

```

```

\gmu@thou@fiver 6528 \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
6530 \ifx\relax#5\relax\@xa\@firstoftwo
6531 \else\@xa\@secondoftwo
6532 \fi{#6}{#7}}

```

```

\gmu@thou@putter 6534 \def\gmu@thou@putter#1#2{% we are sure to have at least five tokens before the
        sentinel \gmu@thou@putter.
6536 \advance\@tempcnta\@ne
6537 \@tempcntb\@tempcnta
6538 \divide\@tempcntb3\relax
6539 \@tempcnta=\numexpr\@tempcnta-\@tempcntb*3
6540 \edef\gmu@thou@put{\@xa{\gmu@thou@put}\unexpanded{#1}%
6541 \ifx\gmu@thou@putter#2\else
6542 \ifcase\@tempcnta
6543 \gmu@thou@o\or\gmu@thou@i\or\gmu@thou@ii% all three CSes
        are yet \relax so we may put them in an \edef safely.
6546 \fi
6547 \fi}% of \edef
6548 \ifx\gmu@thou@putter#2% if we are at end of the digits...
6549 \edef\gmu@tempa{%
6550 \ifcase\@tempcnta
6551 \gmu@thou@o\or\gmu@thou@i\or\gmu@thou@ii
6552 \fi}%
6553 \@xa\let\gmu@tempa\gmu@thousep% ... we set the proper CS...
6554 \else% or ...
6555 \afterfi{% iterate.
6556 \gmu@thou@putter#2}% of \afterfi
6557 \fi% of if end of digits.
6558 }% of \gmu@thou@putter.

```

```

\gmu@thousep 6560 \def\gmu@thousep{\,}% in Polish the recommended thousand separator is a thin
        space.

```

So you can type `\thousep{7123123123123}` to get 7 123 123 123 123. But what if you want to apply `\thousep` to a count register or a `\numexpr`? You should write one or two `\expandafters` and `\the`. Let's do it only once for all:

```

\xathousep 6568 \pdef\xathousep#1{\@xa\thousep\@xa{\the#1}}

```

Now write `\xathousep{\numexpr 10*9*8*7*6*120}` to get 3 628 800.

```

\shortthousep 6572 \def\shortthousep{%
\thous 6573 \DeclareCommand\thous{Q{+-0123456789}}{%

```

we declare it as a command with Q-type argument to allow spaces between digits.

```

6576 \ifmmode\hbox\bgroup\@gmu@mmhboxtrue\fi
6577 \IfValueTF{##1}{% we are given a sequence of digits
6578 \@tempcnta=##1\relax
6579 \ifnum\@tempcnta<0\@xa\relax
6580 \@tempcnta=-\@tempcnta
6581 \fi
6582 \xathousep\@tempcnta
6583 \if@gmu@mmhbox\egroup
6584 \else\@xa\spifletter
6585 \fi
6586 }%

```

```

6587     {% no bare digits given, then we assume the argument is braced.
6588     \thousep
6589     }%
6590   }% of \thous.
6591 }% of \shortthousep.

```

And now write `\thous 3 628800` to get 3 628 800 even with a blank space (beware of the range of T_EX's counts).

hyperref's `\nolinkurl` into `\url`*

```

\urladdstar 6599 \def\urladdstar{%
6600   \AtBeginDocument{%
6601     \@ifpackageloaded{hyperref}{%
6602       \StoreMacro\url
\url 6603       \pdef\url{\gm@ifstar{\nolinkurl}{\storedcsname{url}}}%
6604       }{}}
6606 \@onlypreamble\urladdstar

```

Footnotes suggested by Andrzej Tomaszewski

```

\ATfootnotes 6611 \DeclareCommand\ATfootnotes{s}{%
We make the footnote mark in the footnote \scriptsize not \scriptscriptsize.
6617   \IfValueT{#1}% the following setting is suitable for old style numbers in foot-
note marks, therefore I place it in the starred version of the command.
6620   {\prependtomacro\gmu@ATfootnotes{%
6621     \pdef\@makefnmark{%
6622       \mbox{\normalfont\textsuperscript{\smaller[3]
\@thefnmark}}}%
6623     }% of prepend,
6624     }% of \IfValueT.
6626   \gmu@ATfootnotes
6627   \gmu@AT@ampulex\maketitle% without hyperref
6628   \ifdefined\HyOrg@maketitle
6629     \afterfi{\gmu@AT@ampulex\HyOrg@maketitle}% with hyperref
6630   \fi
6631 }
\gmu@AT@ampulex 6633 \pdef\gmu@AT@ampulex#1{%
6634   \ampulexdef#1{\def\@makefnmark}%
6635   \if@twocolumn
6636     {\gmu@ATfootnotes\if@twocolumn}% Ampulex redefinition of \maketi|
title for the standard classes.
\@makefnntext 6638   \ampulexdef#1{\long\def\@makefnntext}%
6639   \if@twocolumn{\gmu@ATfootnotes\if@twocolumn}% Ampulex redefini-
tion of \maketitle for mwcls.
6641 }
\gmu@ATfootnotes 6643 \pdef\gmu@ATfootnotes{%
And we make the footnote number not be in superscript but on the base line, accord-
ing to Andrzej Tomaszewski's suggestion on BachoTEX 2008, and the same size as in the
footnote mark.
\@makefnntext 6647   \long\pdef\@makefnntext##1{%

```

```

6648     \ifdefined\@parindent\parindent\@parindent
6649     \else\parindent\relax
6650     \fi
6651     \indent{\ATf@font\scriptsize%
6652       {\@thefnmark}}%
6653     \gmu@fnhook
6654     \enspace\ignorespaces##1}%
6655 }

6657 \let\ATf@font\normalfont
6659 \emptify\gmu@fnhook

\rrthis 6661 \pdef\rrthis{% 'rag right this': make only the current paragraph ragged right
        (e.g. if the paragraph consists of a long URL).
6664     \begingroup\rightskip=osp\plus\hsize\endgraf\endgroup}

\centerthis 6666 \pdef\centerthis{% 2009/12/15
6667     \begingroup
6668     \rightskip=1\rightskip\plus\hsize
6669     \leftskip=1\leftskip\plus\hsize
6670     \parfillskip=\z@skip
6671     \endgraf\endgroup}

\balsmiley 6674 \def\balsmiley#1{}% to balance parentheses and brackets in smileys. ;- )
        % \balsmiley(\;-) .

\scantnoline 6681 \long\pdef\scantnoline#1{% 'rescan tokens without adding line end'
6682     {\endlinechar\m@ne\scantokens{#1}}

```

A fix to the url package

It happened that a URLs typeset with the `\url` command of the `url` package came out sort of spaced because kerning was off because of the math mode. So I provide a redefinition of the internal macros of the `url` package which in my version uses not math mode but `\scantokens` and not `\relpenalty` and `\binoppenalty` but `\hyphenpenalty` (as it is in the paragraph) and `\discretionary`. I tried putting explicit penalties after the symbols but that spoiled kerning.

The rules of line breaking are somewhat different, too: in the original `url` package line breaks are forbidden between any two symbols listed in `\UrlBigBreaks`. In my version line breaks are forbidden between any two *identical* 'URL Breaks' and 'URL Big Breaks'.

There are some more differences in formatting some chars, i.a. `~`, `%` and angle brackets which I don't treat specially and just take from font assuming the font provides ASCII chars and checking whether it provides the angle brackets.

```

6707 \@ifXeTeX{%
\urlFix 6708     \pdef\urlFix{\AtBeginDocument{%
6709         \@ifpackageloaded{url}{\gm@urlFix}}}%
6710     \relaxen\urlFix}%
6712     \AtBeginDocument{%
\urlFix 6713     \pdef\urlFix{%
6714         \@ifpackageloaded{url}{\gm@urlFix}}}%
6715     \relaxen\urlFix}}%
6716 }
6717 {%
\urlFix 6718     \pdef\urlFix{\PackageWarning{gmutils}{!!!\The\string%

```

```

        \UrlFix\space
6719     declaration\works\only\with\XeTeX}}%
6720 }

6723 \@ifXeTeX{}{%
6724     \edef\gmu@restoreUpUpUp{\catcode`\@nx\^^^=\the\catcode`%
        \^^^}%
6725     \AtEndOfPackage\gmu@restoreUpUpUp
6726     \catcode`\^^^=9\}

\gm@UrlFix 6728 \def\gm@UrlFix{%
        default style assignments

\UrlBreaks 6731 \def\UrlBreaks{\do\.\do@|\do\\|\do\/|\do!|\do\_|\do|||\do\;|
        \do\}%
6732     \do\)\do\,\do\?\do\'\do\\"\do\+\do\=\do\#\do\%\do\~\do%
        \_\do\|}%
6733     \do\{\do\}\do\$}%
\UrlBigBreaks 6734 \def\UrlBigBreaks{\do\:}%
\UrlNoBreaks 6735 \def\UrlNoBreaks{\do\(\do\[\do\{\}%
\UrlSpecials 6736 \def\UrlSpecials{%
6737     \do\_{\hbox{\visiblespace}}\do\^M{\hbox{
        \visiblespace}}}%

\Url@Format 6741 \def\Url@Format##1{%
6742     \UrlFont
6743     \ifdefined\verbatim@specials
6744         \catcode`\>\active
6745         \verbatim@specials
6746         \verbatim@mathhack
6747         \fi\% setting of the escape char, begin and end group and optionally math
        shift, defined in gmverb.
6750     \gm@UrlSetup
6751     \UrlLeft
6752     \edef\gmu@theendlinechar{\the\endlinechar}%
6753     \endlinechar\m@ne
6754     \kern\z@% to forbid hyphenating the first word if the URL begins with
        a word
6756     \hyphenchar\font=\UrlHyphenchar\relax
6757     \let\-\gmu@discretionaryhyphen
6758     \scantokens{##1}%
6759     \endlinechar\gmu@theendlinechar\relax
6760     \UrlRight
6761 }% of \Url@Format.

6763 \edef\UrlHyphenchar{%
6764     \ifdefined\gmv@hyphenchar\gmv@hyphenchar
6765     \else"A6\fi}% broken bar, | or the same as provided in gmverb for verba-
        tims. You can redefine it as you please. This char is used as the hyphen-
        ation 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.

\verbatim@mathhack 6773 \def\verbatim@mathhack{%
6774     \ifdefined\verbatim@specials@list

```

```

6775     \@xa\verbatim@mathhack@\verbatim@specials@list
6776     \fi
6777     }%

\verbatim@mathhack@ 6779     \def\verbatim@mathhack@##1##2##3##4##5##6{%
6780     \IfValueT{##4}{%
6781     \edef\gmu@thinmuskip{\the\thinmuskip}%
6782     \edef\gmu@medmuskip{\the\medmuskip}%
6783     \edef\gmu@thickmuskip{\the\thickmuskip}%
6784     \begingroup
6785     \lccode`~=#4\lowercase{%
6786     \endgroup\def~###1~}%
\thinmuskip 6787     {$\thinmuskip\gmu@thinmuskip\relax
6788     \medmuskip\gmu@medmuskip\relax
6789     \thickmuskip\gmu@thickmuskip\relax
6790     ###1%
6791     $}%
6792     \catcode`##4\active
6793     }%
6794     }%

\gm@UrlSetup 6796     \def\gm@UrlSetup{%
6797     \medmuskip\Urlmuskip_\thickmuskip\medmuskip_\%
        \thinmuskipomu%
6798     \relpenalty\UrlBigBreakPenalty_\binoppenalty%
        \UrlBreakPenalty
6801     \def\do{\gmu@doUrlMath\UrlBreakPenalty}\UrlBreaks_\% bin(\hy!
        phenpenalty anyway)
6803     \def\do{\gmu@doUrlMath\UrlBigBreakPenalty}\UrlBigBreaks_\% rel
        (\hyphenpenalty anyway)
6805     \def\do{\gmu@doUrlMath\@M}\UrlNoBreaks_\% open(no break)
6806     \def\do{\gmu@doUrlMathAc\UrlBreakPenalty}\% (\hyphenpenalty)
6807     \UrlSpecials
6808     \if_\iffontchar\font"2329_\1\elseo\fi\iffontchar%
        \font"232A_\1\else2\fi
        we check whether the font provides both left and right angle brackets.
6811     \gmu@measurewd{^^^2329}%
6812     \edef\gmu@tempa{%
6813     \@nx\gmu@doUrlMathAc\@M\@nx\<{%
6814     \hbox_\to\gmu@tempb{\unexpanded{\hss\char"2329_\%
        \hss}}}%
6815     }\gmu@tempa
6816     \gmu@measurewd{^^^232a}%
6817     \edef\gmu@tempa{%
6818     \@nx\do\@nx\>{%
6819     \hbox_\to\gmu@tempb{\unexpanded{\hss\char"232A_\%
        \hss}}}%
6820     }\gmu@tempa
6821     \else
6822     \gmu@doUrlMathAc\@M\<{\langle}\do\>{\rangle}%
6823     \fi
6824     \iffontchar\font"22C6_\% low star
6825     \do\*{\hbox{\char"22C6_\}}%
6826     \else_\do\**%

```

```

6827 \fi
6828 \ifx\do@url@hyp\@empty
6829 \gmu@measurewd{-}% this macro is defined in line 5197.
6830 \edef\gmu@tempa{%
6831 \unexpanded{\gmu@doUrlMathAc\@M\-%}
6832 {\hbox\to\gmu@tempb{\unexpanded{\hss-\hss}}%
6833 \@nx\-%} hyphen is a good point for hyphenation, but the hyphen-
ation char should be sth. else, and it is indeed: | (broken bar,
\char"A6). See also line 6765
6837 }\gmu@tempa
6838 \fi
6839 \addfontfeature{Ligatures=NoCommon, \Mapping=none}% instead of
'doing' \verbatim@nolig@list.
6842 }% of \gm@UrlSetup.
\gmu@doUrlMath 6849 \def\gmu@doUrlMath##1##2{%
% #1 value of the penalty (used as a Boolean: if < 10 000,
% \hyphenpenalty will be used anyway, if ≥ 10 000, there will be no
% \discretionary),
% #2 the char, given as \<char>.
6861 \begingroup
6862 \lccode`~`##2\lowercase{%
6863 \endgroup\def~{\@ifnextchar~}%
6864 \@xa\addtomacro\@xa~}% of \lowercase.
6865 \ifnum##1<\@M
6866 {%
6867 {\char`##2\csname\gmu@dbl\string##2kern\endcsname}% if next
is the same char
6868 {\ifmmode\char`##2% else
6869 \else\gmu@urlbreakable{##1}{##2}%
6870 \fi}%
6871 }% of \addtomacro's argument \ifnum true.
6872 \else
6873 {%
6874 {\char`##2\csname\gmu@dbl\string##2kern\endcsname}{%
\char`##2}%
6875 }% of \addtomacro's argument \ifnum false.
6876 \fi
6877 \catcode`##2=\active
6878 }% of \gmu@doUrlMath.
\gmu@doUrlMathAc 6880 \def\gmu@doUrlMathAc##1##2##3{%
% #1 (value of) a penalty (see the remark to ##1 of the previous macro),
% #2 the char (as \<char>),
% #3 the definition.
6887 \begingroup
6888 \lccode`~`##2\lowercase{%
6889 \endgroup\def~{\@ifnextchar~}%
6890 \@xa\addtomacro\@xa~}% of \lowercase.
6891 \ifnum\##1<\@M
6892 {%
6893 {\ifmmode\char`##2\else$##3\m@th$\fi}%
6894 {\ifmmode\char`##2%

```



```

6895         \else\discretionary{\hbox{$##3\m@th$}}{\hbox{$##3%
        \m@th$}}%
6896         \fi}%
6897     }% of \addtomacro's argument if num true.
6898     \else
6899     {%
6900         {\ifmmode\char`##2\else$##3\m@th$\fi}{\ifmmode%
        \char`##2\else$##3\m@th$\fi}%
6901     }% of \addtomacro's argument if num false.
6902     \fi
6903     \catcode`##2=\active
6904 }% of \gmu@doUrlMathAc.
\gmu@url@rigidbreak 6906 \pdef\gmu@url@rigidbreak##1##2{\discretionary{\char`##2}{%
        }\char`##2}}%
\gmu@url@flexbreak 6908 \pdef\gmu@url@flexbreak##1##2{\penalty\@M\hskip\z@_
        pluso,03em
6909     \char`##2\penalty##1\hskip\z@_pluso,03em\relax}%
6911 \let\gmu@urlbreakable\gmu@url@flexbreak
\Url@z 6913 \def\Url@z##1{%
        Do any hyper referencing due to hyperref (or perform a url-def)
6915     \Url@HyperHook
        Now do the formatting in a group (can also have \Url@HyperHook take this as an
        argument).
6918     {\Url@Format{##1}}%
6919     \endgroup}%
6921 \DeclareUrlCommand\file{\urlstyle{sf}}%
6923 \emptify\Url@moving% with our settings \url is pretty allowed in moving
        arguments, I hope.
6925 }% of \gmu@UrlFix.
\UrlSlashKern 6927 \DeclareCommand\UrlSlashKern{O{tt}m}%
6928 {\AtBeginDocument{%
6929     \nameedef{url@#1style}{\def\@nx\UrlFont{%
6930         \@xa\@nx\csname#1family\endcsname
6931         \def\@xa\@nx\csname_\gmu@dbl\string\kern\endcsname
6932         {\kern#2\relax}%
6933     }% of \UrlFont
6934     }% of \url#1style
6935     \urlstyle{#1}%
6936     }% of \AtBeginDocument
6937 }% of \UrlSlashKern

```

Conditional tilde

Polish typesetting standards say that for 12 dd and 10 dd *<zcionki>* if leading is narrower than $3\frac{1}{2}$ *<kwadratu>*, $3\frac{1}{2} \times 48$ dd, then hanging letters are allowed, which also applies to 8 and 6 dd *<zcionki>* in less than 3 *<kwadrat>* leading. I treat this recommendation not strictly but as an inspiration, that is I translate »dd« to »pt«.

```

\TrzaskaTilde 6952 \def\TrzaskaTilde{%
6953   \@xa\DeclareCommand\@xa\gm@smarttilde
6954   \@xa{\@xa_S\@xa{\all@stars~}}{%
6955     \IfValueTF{##1}{\nobreakspace}}{%
6956     {\ifdim\dimexpr\hsize-\leftskip-\rightskip
6957       -\ifdim\hangindent<\z@-\fi\hangindent_}% the last parameter is
           used with respect to the floatflt package.
6959     >%
6960     \ifdim\fontsize_pt>\dimexpr10pt-1sp\relax
6961       168dd
6962     \else
6963       144dd
6964     \fi
6965     \nobreakspace_}}%
6966     \else
6967     _}%
6968     \fi
6969     }% of \gm@ifstar's else,
6970     }% of \gm@smarttilde,
6971     \let~\gm@smarttilde
6972 }% of \TrzaskaTilde.

\getprevdepth 6975 \pdef\getprevdepth{%
6976   \endgraf
6977   \xdef\setprevdepth{\prevdepth=\the\prevdepth\relax}%
6978 }

```

Storing the catcode of line end

```

\StoreCatM 6981 \def\StoreCatM{%
6982   \protected\edef\RestoreCatM{%
6983     \catcode` \@nx^^M=\the\catcode` ^^M\relax}%
6984 }

\RestoreCatM 6986 \pdef\RestoreCatM{\PackageE{gmutils}{first_store_the_catcode_
           of
6987   ^\empty^\empty_M_with_\string\StoreCatM.}%
6988 }

```

A really empty page

Copied from Marcin Woliński's macros.

```

\clearempydoublepage 6994 \newcommand{\clearempydoublepage}{%
6995   \newpage{\pagestyle{empty}\cleardoublepage}}

\setspaceskip 6998 \DeclareCommand\setspaceskip{%
6999   A}% optional factor for all three components
7000   O{\fontdimen2\font}%
7001   >is
7002   O{\fontdimen3\font}%
7003   >is
7004   O{\fontdimen4\font}}

```

```

7005 {\spaceskip=#1#2plus#1#3minus#1#4\relax}
7007 \foone\obeylines{%
\disobeylines 7008   \def\disobeylines{% for arguments in which line end is active to simulate
normal behaviour
7010   \ifnum\catcode`\^^M=\active%
\@ifnextgroup 7011   \pdef^^M{\@ifnextgroup{\ifhmode\unskip\space\fi}{%
\gmu@disMinner}}%
\gmu@disMinner 7012   \def\gmu@disMinner##1{%
7013   \ifx^^M##1\endgraf%
7014   \else\afterfi{\ifhmode\unskip\space\fi}\fi##1}%
7015   \fi}%
7016 }

\makestarlow 7022 \def\makestarlow{%
7023   \begingroup\lccode`\~=\*\lowercase{%
* 7024   \endgroup\def~{\gmu@lowstar}}% 2009/10/19 \let changed to \def
\gmu@lowstar 7025   to allow redefinitions of \gmu@lowstar.
7026   \catcode`\*=\active
7027   \defLowStarFake
7028 }

\defLowStarFake 7030 \DeclareCommand\defLowStarFake{%
7031   Q{+-0123456789,.}{0,5}% fraction of fontchar depth of the star glyph
7032 }%
7033 {%
\gmu@lowstarfake 7034   \def\gmu@lowstarfake{%
7035     \leavevmode\vbox{\hbox{*}\kern#1\fontchardp\font`*}%
7036   }%
7037 }

\gmu@lowstarfake 7040 \def\gmu@lowstarfake{*}_% useful for next command where normal star is
low.

The \* CS and active * should be defined different to make them distinguishable by
tests, especially with \gm@ifstar in mind.

\* 7048 \DeclareCommand\*{Q{0123456789}{1}}
7050 {\gmu@star@loopo{#1}}

\gmu@star@loop 7052 \def\gmu@star@loop#1#2{% this is an expandable loop as in The  $\epsilon$ -TeX Manual
p. 9.
7054   \ifnum#1<\numexpr#2\relax%
7055   \gmu@lowstar
7056   \@xa\gmu@star@loop
7057   \@xa{\number\numexpr#1+1\@xa}%
7058   \@xa{\number#2\@xa}%
7059   \fi}
7063 \endinput

```

Change History

- 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`, 2683
- General:
 Added macros to make sectioning commands of `mwcls` and standard classes compatible. Now my sectionings allow two optionals in both worlds and with `mwcls` if there's only one optional, it's the title to toc and running head not just to the latter, 7063
- vo.75
`\@ifnextcat`:
`\let` for #1 changed to `\def` to allow things like `\noexpand ~`, 794
`\@ifnextif`:
`\let` for #1 changed to `\def` to allow things like `\noexpand ~`, 836
`\@ifnif`:
 added, 876
- vo.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, 7063
`\freeze@actives`:
 added, 5536
- vo.77
 General:
`\afterfi` & `pals` made two-argument as the Marcin Woliński's analogoi are. At this occasion some redundant macros of that family are deleted, 7063
- vo.78
 General:
`\@namelet` renamed to `\n@melet` to solve a conflict with the beamer class. The package contents regrouped, 7063
- vo.79
`\not@onlypreamble`:
 All the actions are done in a group and therefore `\xdef` used instead of `\edef` because this command has to use `\do` (which is contained in the `\@preamblecmds` list) and `\not@onlypreamble` itself should be able to be let to `\do`, 3312
- vo.80
 General:
 CheckSum 1689 , 0
`\hfillneg`:
 added, 5433
- vo.81
`\dekfracslash`:
 moved here from `pmlectionis.cls`, 5728
`\ifSecondClass`:
 moved here from `pmlectionis.cls`, 5697
- vo.82
`\ikern`:
 added, 5736
- vo.83
`\texttilde`:
 postponed to `\begin{document}` to avoid overwriting by a text command and made sensible to a subsequent `/`, 5389
- vo.84
 General:
 CheckSum 2684 , 0
- vo.85
 General:
 CheckSum 2795 , 0
 fixed behaviour of too clever headings with `gmdoc` by adding an `\ifdim` test, 7063
- vo.86
`\texttilde`:
 renamed from `\~` since the latter is one of L^AT_EX's accents, 5389
- vo.87
 General:
 CheckSum 4027 , 0
 the package goes ϵ -T_EX even more, making use of `\ifdefined` and the code using UTF-8 chars is wrapped in a X_YT_EX-condition, 7063
- vo.88
 General:
 CheckSum 4040 , 0
`\RestoreEnvironment`:
 added, 1181
`\storedcsname`:
 added, 1172
`\StoreEnvironment`:
 added, 1177
- vo.89
 General:
 removed obsolete adjustment of `pgf` for X_YT_EX, 7063
- vo.90
 General:
 CheckSum 4035 , 0
`\XeTeXthree`:
 adjusted to the redefinition of `\verb` in `xlxtra 2008/07/29`, 4158
- vo.91
 General:
 CheckSum 4055 , 0

removed `\jobname\woe` since `\jobname` is always without extension. `\xiispace` forked to `\visible\space` `\let` to `\xxt@visible\space` of `xltxtra` if available. The documentation driver integrated with the `.sty` file, 7063

vo.92
`\@checkend`:
shortened thanks to `\@ifenvir`, 2771
`\@gif`:
added redefinition so that now switches defined with it are `\protected` so they won't expand to a further expanding or unbalanced `\iftrue/false` in an `\edef`, 489
`\@ifenvir`:
added, 2707
`\@ifprevenvir`:
added, 2748
General:
Checksum 4133 , 0

vo.93
`\@nameedef`:
added, 282
General:
A couple of
`\DeclareRobustCommand*`
changed to `\pdef`, 7063
Checksum 4140 , 0
Checksum 4501 , 0
The numerical macros commented out as obsolete and never really used, 7063
`\ampulexdef`:
added, 2523
`\em`:
added, 5110, 5119
`\gmu@RPfor`:
renamed from `\gmu@RPif` and #3 changed from a csname to CS, 5600
`\litshape`:
copied here from E. Szarzyński's *The Letters*, 5068
`\lsl`:
copied here from E. Szarzyński's *The Letters*, 5093
`\nocite`:
a bug fixed: with `natbib` an 'extra' error. Now it fixes only the standard version of `\nocite`, 3349
`\pdef`:
added, 240
`\pprovide`:
added, 269
`\provide`:
added, 254
`\textlit`:
added, 5085

`\thousep`:
added, 6513

vo.94
`\@xau`:
added, 225
General:
`\bgroup` and `\egroup` in the macro storing commands and in `\foone` changed to `\begin\group` and `\end\group` since the former produce an empty `\mathord` in math mode while the latter don't, 7063
Checksum 4880 , 0
removed `\unex@namedef` and `\unex@nameuse`, probably never really used since they were incomplete: `\edef@other` undefined, 7063
The code from ancient `xparse` (1999) of `TeXLive` 2007 rewritten here, 7063
`\afterfifi`:
`\if` removed from parameters' string, 357
`\ampulexdef`:
made `xparse`-ish and `\ampulexset` removed, 2523
`\dekfrac`:
made to work also in math mode, even with math-active digits, 4265
`\gm@ifundefined`:
added. All `\@ifundefined`s used by me changed to this, 623
made robust to unbalanced `\ifs` and `\fis` the same way as \LaTeX 's `\@ifundefined` (after a heavy debug :-), 623
`\gmathfurther`:
removed definition of `\langle letter \rangle`s and `\langle digit \rangle`s, 4844
`\ldate`:
`\leftline` replaced with `\par ... \par` to work well with `floatflt`, 6303
`\prependtomacro`:
order of arguments reversed, 589
`\resizegraphics`:
`\includegraphics` works well in \XeTeX so I remove the complicated version with `\XeTeXpicfile`, 4294
`\textbullet`:
the \XeTeX version enriched with `\iffontchar` due to lack of bullets with the default settings reported by Morten Høgholm and Edd Barrett, 6193

vo.95
General:
Checksum 4908 , 0

`\gm@testdefined:`
 added, 641
`\gm@testundefined:`
 added, 656

vo.96

General:

- Checksum 5363 , 0
- put to CTAN on 2008/11/21, 0, 7063

`\glyphname:`
 moved here from my private document class, 6198

`\gmathfurther:`
 Greek letters completed. Wrapped with `\addtotoks` to allow using in any order with `\garamath` and others, 4844
 the `\everymath`'s left brace moved here: earlier all the stuff was put into `\everymath`, 4913

`\gmathscripts:`
 added, 4959

`\LuaTeX:`
 added, 4109

`\pk:`
`\textup` removed to allow slanting the name in titles (that are usually typeset in Italic font), 3053

vo.97

`*`:
 removed (it was a text tilde, available as `\texttilde`), 5381

`\@allbutfirstof:`
 added, 322

General:

- Checksum 5375 , 0
- put to CTAN on 2008/11/22, 0

`\pdfLaTeX:`
 added, 4086

vo.98

`\@allbutfirstof:`
 renamed from `\@secondofmany`, 322

`\@ifedetokens:`
 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`, 2722

`\@ifinmeaning:`
 moved here from `gmdoc` and rewritten, including split to `\IfAmong` because the latter is needed in `\DeclareCommand`, 442

`\@ifnextsingle:`
 added test for `\egroup`, 956
 extracted from `\@ifnextac`, 951

General:

- Checksum 5429 , 0
- Checksum 5577 because of `\DeclareCommand`, 0
- Checksum 5627 because of `\fakeonum`, 0
- Checksum 6035 because of `\DeclareCommand`, `\arg`, 0
- Checksum 6129 because of `\DeclareEnvironment`, 0
- Checksum 6147 because of `\arg` in `verbatim`s, 0
- Checksum 6535 because of `\UrlFix`, 0
- Checksum 6656 because of type settings for `gmdoc` (`\narrativett`) and for `\verbatimspecials`, 0

`\addto@macro:`
`\toks@` replaced with some `\expandafters` because use of `\toks@` here caused a disaster in `\DeclareCommand`, 576

`\ampulexdef:`
 a bug fixed: added `\unexpanded`, 2539
 first argument (the prefix) made of the `Q` type, 2523

`\arg:`
 made iterating thanks to `\DeclareCommand\arg@dc`, 3299

`\ATfootnotes:`
 moved here from my own private macro package to allow the beauty of these footnotes for the general audience, 6611

`\balsmiley:`
 moved here from my personal macro package since I needed it in the documentation of `gmutils`, 6674

`\cs:`
 added `\verbatim@specials`, 3067
 made `\-` switch to `\normalfont` because IMHO this underlines the fact that a CS belongs to the narrative.
`\hyphenchar` set to 45 as in usual texts, 3067

`\DeclareCommand:`
 added the `\@bsphack—\@esphack` option, 2262
 added the `Q` {*tokens*} argument type (a word over the alphabet *tokens*), 2262
 added the `S/T` spec parsing, the `s` spec parsing rewritten to be a shorthand for `S` {***}, unused code removed, 1605

`\enoughpage:`
`\par` removed since to let it be used for `\pagebreak` inside a paragraph, 5495

made ϵ -TeX-ish, 5495
 made ifthenelse-like with ‘then’ and
 ‘else’ optional default *<nothing>* and
 \newpagemresp., 5495

\fakeonum:
 added, 5148

\gmu@tempb:
 renamed from \@ifstar since
 something redefines \@ifstar, 935

\IfAmong:
 split from \IfAmong, 399

\IfNoValueTF:
 the xparse tests for the presence of
 value redefined and much simplified
 (43 CS'es less). Moreover, they are
 now fully expandable:
 \IfNoValueTF, \IfNoValueT,
 \IfNoValueF, \IfValueTF,
 \IfValueT, \IfValueF, 2162

\napapierkicore:
 added optional star controlling
 globalness, 6003

\rrthis:
 added, 6661

\scantnoline:
 added, 6682

vo.99
 \@ifedetokens:
 moved to a separate macro from
 \@ifenvir and made symmetric to
 both arguments, 2722

vo.991
 \@ifnextif:
 inner macro fixed to handle active
 chars more properly (more as \if
 would do it), 836

\@xau:
 made 1-parameter, 225

General:
 CheckSum 8257 because of some
 shorthands and major development
 of \DeclareCommand, including
 ‘Knuthian’ and general optional
 arguments and
 ‘\afterassignment’
 pseudo-argument, 0
 put to CTAN on 2010/03/04, 0

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.

#1, <u>5900</u>	\@M, 3067, 3669, 5736,	\@currenvline, 2666
*, 7024	5741, 6805, 6813,	\@currsiz, 2835, 2836,
*, 691, 693, 925, 985, <u>986</u> ,	6822, 6831, 6865,	2837, 2838, 2839,
987, 989, 993, <u>994</u> ,	6891, 6908	2840, 2841, 2842,
995, 999, 5591, <u>5592</u> ,	\@aarg, 3210, 3211, 3212,	2843, 2844
6825, 6826, 7023,	3214, 3217	\@dc, 1629, 1657, 1708,
7026, <u>7048</u>	\@aarga, 3210, 3212	1732, 1733, 1768,
\+, <u>5593</u> , 6732	\@afterheading, 3662	2316, 2386
\-, 2976, 3008, 3010, 3082,	\@allbutfirstof, <u>322</u>	\@dc@, 1631, <u>1643</u> , 2327
6181, 6757, 6831, 6833	\@badend, 2771	\@dc@Gname, 2118, 2119,
\<...>, <u>2945</u>	\@begnamedgroup, <u>2652</u> ,	2120, 2125, 2136,
\@@nil, 319, 322, 2168,	2676, 2681	2137, 2138, 2142
2614, 2619, 3023,	\@begnamed	\@dc@Gparams, 2121,
3025, 3027, 3136,	group@ifcs, 2677,	2139, <u>2147</u>
4383, 4391, 5254,	<u>2680</u>	\@dc@K@defaultrepl,
5255, 5288, 5289,	\@car, 4019	<u>2359</u> , 2375
5291, 5295, 5303,	\@cclv, 5574	\@dc@add@ms, 1666, 1684,
5305, 5308, 5318,	\@cclvi, 5559	1720
5320, 5322, 5329,	\@checkend, <u>2771</u>	\@dc@addargum, <u>1750</u> ,
5331, 5333, 5339,	\@clsextension, 5697	1753, <u>1756</u> , 1757,
5343, 5756, 5766,	\@clubpenalty, 3674	1778, 1781, 1786,
5918, 5923, 5926,	\@currenvir, 2659, 2661,	1789, 1800, 1812,
5927, 5931	2719	1821, 1828, 1840,

1851, 1862, 1863,
 1876, 1929, 1940,
 1946, 1947, 1953,
 1954, 1959, 1962,
 1970, 1976, 1982,
 1986, 2024, 2092,
 2097, 2121, 2126,
 2139, 2143, 2206, 2374
 \@dc@alllong@false, 2287
 \@dc@alllong@true, 2292
 \@dc@argnum, 1599, 1617,
 1693, 1695, 2311
 \@dc@arguments, 1596,
 1654, 1751, 1771,
 1778, 1781, 1783,
 1786, 1789, 1791,
 1800, 1803, 1812,
 1815, 1821, 1824,
 1827, 1828, 1831,
 1840, 1842, 1850,
 1851, 1862, 1863,
 1872, 1876, 1899,
 1931, 1934, 1940,
 1947, 1954, 1959,
 1962, 1967, 1970,
 1973, 1976, 1979,
 1982, 1985, 1986,
 2015, 2026, 2087,
 2093, 2096, 2097,
 2105, 2116, 2121,
 2126, 2134, 2139,
 2143, 2204, 2207,
 2378, 2413
 \@dc@bsphack@, 2296,
 2299, 2302, 2320
 \@dc@catchernum, 1598,
 1616, 1691, 2310,
 2368, 2372
 \@dc@commasep, 1946,
 1949, 1949, 1953
 \@dc@define@K, 1705,
 1712, 2363
 \@dc@env@argstoend,
 2304, 2344, 2409
 \@dc@esphack@, 2297,
 2299, 2302, 2350
 \@dc@global@, 2305,
 2307, 2318, 2339,
 2380, 2438, 2450, 2453
 \@dc@ignorearg, 1685, 1755
 \@dc@ignorefalse,
 1628, 1697
 \@dc@ignoretrue, 1672,
 1676, 1767
 \@dc@long@, 1627, 1671,
 1678, 1687, 1713,
 1765, 2381, 2385
 \@dc@long@yes, 1594, 1671
 \@dc@outer@, 2305, 2306,
 2318, 2329
 \@dc@parse@next, 1773,
 1781, 1798, 1817,
 1838, 1901, 1931,
 1936, 1947, 1954,
 1959, 1962, 1969,
 1975, 1981, 2015,
 2026, 2106, 2205,
 2208, 2378
 \@dc@quiet@false, 2288
 \@dc@quiet@true, 2293
 \@dc@sequence, 1902,
 1921, 1928, 1928, 1929
 \@dc@sequence@finish,
 1912, 1923, 1926
 \@dischyph, 3011
 \@emptyfy, 599, 600, 604
 \@enumctr, 3928, 3929, 3935
 \@enumdepth, 3924, 3927,
 3928
 \@filepart, 5821, 5823
 \@fileswithoptions, 5697
 \@firstofmany, 319,
 2168, 3136
 \@firstofone, 1650
 \@firstofthree, 309
 \@gif, 475, 476, 483
 \@gmu@mmhboxtrue,
 4267, 6576
 \@if, 498, 499, 502
 \@ifQueerEOL, 3288
 \@ifXeTeX, 4150, 4156,
 4162, 4333, 5725,
 6192, 6707, 6723
 \@ifdetokens, 2719,
 2722, 2746, 2760
 \@ifempty, 386, 397, 409,
 1721, 2541, 2984,
 2988, 2994, 3001,
 3604, 3624
 \@ifendswithpdf, 4300,
 4305
 \@ifenvir, 2707, 2771, 6507
 \@ifinmeaning, 442, 5236
 \@ifjobname, 2746
 \@ifl@aded, 3335
 \@ifncat, 801, 804, 819
 \@ifnextMac, 1009, 6102,
 6121
 \@ifnextac, 887
 \@ifnextcat, 794, 833,
 955, 956, 1816, 1826,
 1837, 1849, 3312
 \@ifnextgroup, 960, 968,
 7011
 \@ifnextif, 836, 938,
 3260, 3262
 \@ifnextsingle, 888,
 951, 962, 1873, 1910
 \@ifnextspace, 992,
 6100, 6120
 \@ifnif, 846, 849, 876
 \@ifnonempty, 397
 \@ifnotmw, 3483, 3508,
 3657, 3762, 3851, 3907
 \@ifprevenir, 2748
 \@include, 5763, 5765
 \@itemdepth, 3942, 3945,
 3946
 \@itemitem, 3946, 3947
 \@makefntext, 6638, 6647
 \@minus, 3773, 3779,
 3785, 3787, 3792,
 3794, 3801, 3806
 \@nameedef, 282, 2441, 6929
 \@nobreakfalse, 3668, 5481
 \@nobreaktrue, 3663
 \@normalcr, 6350
 \@nx, 223
 \@oarg, 3198, 3200, 3201, 3250
 \@oargsq, 3198, 3201
 \@onlypreamble, 3375,
 4837, 5483, 6606
 \@parg, 3205, 3207, 3208,
 3251
 \@pargp, 3205, 3208
 \@parindent, 3931, 3932,
 3934, 3949, 3950,
 3952, 6383, 6417,
 6425, 6426, 6427,
 6429, 6648
 \@pkgextension, 3336
 \@preamblecmds, 3330,
 3332, 3345, 3349
 \@prevenir, 2659, 2760
 \@prevgrouplevel, 2653
 \@relaxen, 608, 609, 613
 \@secondoffive, 312
 \@secondofthree, 310
 \@starttoc, 5476
 \@tempdima, 5072, 5079,
 5096, 5099, 5949,
 5953, 5954, 5956,
 5957, 5961
 \@tempdimb, 5199, 5204,
 5950, 5952, 5953
 \@temptokena, 1622,
 1637, 1694, 1695,
 2313, 2342, 2456
 \@temptokenb, 1582,
 1583, 1623, 1681,
 1689, 1721, 1725, 2314

<code>\@textsuperscript,</code>	3285, 4153, 4410,	<code>\ArgumentCatcher@Pa,</code>
4326, 4327	5644, 5660, 5805,	2157
<code>\@thirdofthree,</code> <u>311</u>	5821, 5923, 6099,	<code>\ArgumentCatcher@PB,</code>
<code>\@toodeep,</code> 3925, 3943	6119, 6555, 6629, 7014	1842
<code>\@topnewpage,</code> 3564	<code>\afterfifi,</code> 256, 257,	<code>\ArgumentCatcher@Pb,</code>
<code>\@typeset@protect,</code> 1649	<u>357</u> , 632, 634, 1097,	1824
<code>\@undefined,</code> 181	1102, 4152, 4404,	<code>\ArgumentCatcher@PC,</code>
<code>\@verbaarga,</code> 3214, 3265	4405, 6178	<u>1979</u>
<code>\@verbaargact,</code> 3217, 3264	<code>\afterfififi,</code> <u>365</u>	<code>\ArgumentCatcher@Pc,</code>
<code>\@verbmargm,</code> 3223, 3260	<code>\afteriffifi,</code> <u>359</u>	<u>1973</u>
<code>\@wckptelt,</code> <u>5891</u>	<code>\afteriffififi,</code> <u>364</u>	<code>\Argument!</code>
<code>\@whilenum,</code> 2006, 2037,	<code>\afteriffiffififi,</code> <u>363</u>	<code>Catcher@Pc@,</code> <u>1951</u> ,
5559, 5574	<code>\ahyphen,</code> <u>6181</u>	1961, 1975, 1981
<code>\@xa,</code> 222, 225	<code>\AKA,</code> <u>5672</u>	<code>\ArgumentCatcher@PG,</code>
<code>\@xau,</code> <u>225</u> , 2375, 6540	<code>\all@other,</code> 2548, <u>4138</u>	2130, 2154, 2157
<code>\@xifncat,</code> 806, 819	<code>\all@stars,</code> <u>690</u> , 692,	<code>\ArgumentCatcher@PK,</code>
<code>\@xifnif,</code> 851, 876	694, 1888, 6954	2108
<code>\@xiispaces,</code> <u>2614</u> , 2616,	<code>\all@unders,</code> 680, 682,	<code>\ArgumentCatcher@Pm,</code>
2619	684, 686	1827, 1850, 1985, 2096
<code>\aarg,</code> 3210, 3253	<code>\alpha,</code> 4782	<code>\ArgumentCatcher@PO,</code>
<code>\acro,</code> <u>5628</u> , 5671, 5672, 5676	<code>\among,</code> 399, 424, 458,	<u>1803</u>
<code>\acrocore,</code> 5655, <u>5665</u>	1699, 1701, 1861,	<code>\ArgumentCatcher@Po,</code>
<code>\acropresetting,</code> 5630,	1920, 1945, 1952,	<u>1783</u>
5636	2195, 2300, 2306, 2307	<code>\Argument!</code>
<code>\activeM,</code> <u>774</u>	<code>\ampulexdef,</code> 2510, 2561,	<code>Catcher@Po@,</code>
<code>\adashes,</code> <u>6156</u> , 6156,	2577, 3372, 4941,	1785, 1788, 1810
<u>6158</u> , <u>6167</u>	6634, 6638	<code>\ArgumentCatcher@PQ,</code>
<code>\adddefaultfontfea </code>	<code>\ampulexhash,</code> <u>2581</u>	1906
<code>tures,</code>	<code>\AmSTeX,</code> <u>4056</u>	<code>\ArgumentCatcher@PS,</code>
<u>4179</u>	<code>\approx,</code> <u>4726</u>	1882
<code>\addfontfeature,</code> 4192,	<code>\arg,</code> <i>p.</i> 46, 3243, 3282, <u>3283</u>	<code>\ArgumentCatcher@Ps,</code>
4223, 4225, 4256,	<code>\arg@dc,</code> 3214, 3217, 3230,	1891
4338, 4992, 5006,	<u>3235</u> , 3255, 3296	<code>\ArgumentCatcher@Q,</code>
5079, 5099, 5158,	<code>\ArgumentCatcher@A,</code> <u>2153</u>	<u>1893</u> , 1906
5277, 5674, 5719,	<code>\ArgumentCatcher@a,</code> <u>2156</u>	<code>\ArgumentCatcher@Q@,</code>
5968, 6839	<code>\ArgumentCatcher@B,</code> <u>1831</u>	1903, <u>1908</u> , 1922
<code>\addto@macro,</code> <u>576</u> , 586	<code>\ArgumentCatcher@b,</code> <u>1815</u>	<code>\Argument!</code>
<code>\addtoheading,</code> <u>3638</u>	<code>\ArgumentCatcher@C,</code> <u>1967</u>	<code>Catcher@Q@@,</code> 1911,
<code>\addtomacro,</code> <u>586</u> , <u>682</u> ,	<code>\ArgumentCatcher@c,</code> <u>1934</u>	<u>1915</u>
684, 686, 692, 694,	<code>\ArgumentCatcher@c@,</code>	<code>\ArgumentCatcher@S,</code>
1921, 2429, 2450,	1936, <u>1944</u> , <u>1958</u> , 1969	<u>1866</u> , 1882, 1883,
4180, 4972, 5159,	<code>\ArgumentCatcher@G,</code>	1888, 2224
6162, 6163, 6164,	<u>2112</u> , 2153, 2156	<code>\ArgumentCatcher@s,</code>
6864, 6890	<code>\ArgumentCatcher@K,</code>	1887, 1891
<code>\addtotoks,</code> <u>595</u> , 1685,	<u>2100</u> , 2108	<code>\ArgumentCatcher@S@,</code>
1686, 1751, 4913,	<code>\ArgumentCatcher@m@,</code>	<u>1854</u> , 1875
4960, 4964	1817, 1838	<code>\ArgumentCatcher@T,</code> 1883
<code>\AE,</code> 5936	<code>\Argument!</code>	<code>\AtBeginDocument,</code> 762,
<code>\afterassignment,</code>	<code>Catcher@mmmmmmmmmm,</code>	3108, 3281, 3343,
1672, 1706, 2201,	2087	3379, 3454, 3969,
2208, 2214, 2215	<code>\ArgumentCatcher@O,</code> 1791	4173, 4189, 4827,
<code>\afterfi,</code> 259, <u>354</u> , 637,	<code>\ArgumentCatcher@o,</code> <u>1771</u>	5123, 5125, 5387,
996, 997, <u>1041</u> , 1119,	<code>\ArgumentCatcher@o@,</code>	5611, 6054, 6095,
2168, 2169, 2608,	1773, 1780, 1798	6156, 6158, 6424,
2619, 2681, 2682,	<code>\ArgumentCatcher@PA,</code>	6600, 6708, 6712, 6928
2969, 2970, 3046,	<u>2154</u>	<code>\AtEndOfPackage,</code> 6725
		<code>\ATf@font,</code> 6651, 6657

<code>\ATfootnotes</code> , 6611	<code>\cos</code> , 4821	<code>\DeclareFontShape</code> ,
<code>\backquote</code> , 703	<code>\count</code> , 4007 , 4008 , 4009 ,	4463 , 4476 , 4488 , 4500
<code>\balsmiley</code> , 6674	4010 , 4011 , 4012 ,	<code>\DeclareLogo</code> , 3971 ,
<code>\begin</code> , 2676	4013 , 4014 , 5557 ,	3991 , 4018 , 4029 ,
<code>\begin*</code> , 2676	5559 , 5560 , 5561 ,	4059 , 4065 , 4068 ,
<code>\beta</code> , 4783	5564 , 5573 , 5574 ,	4070 , 4073 , 4076 ,
<code>\bgcolor</code> , 4996	5575 , 5576	4083 , 4085 , 4086 ,
<code>\BibTeX</code> , 4059	<code>\cs</code> , 3067 , 3101 , 3106 , 3109 ,	4090 , 4097 , 4109
<code>\bidate</code> , 6271 , 6281	3182	<code>\DeclareMathVersion</code> ,
<code>\bigcircle</code> , 4762	<code>\cs@inner</code> , 3091 , 3093	4443
<code>\Biggl</code> , 4909	<code>\ctan</code> , 4825	<code>\DeclareOption</code> , 215
<code>\biggl</code> , 4907	<code>\ctg</code> , 4823	<code>\DeclareRobustCom </code>
<code>\Biggr</code> , 4910	<code>\cup</code> , 5005	<code>mand</code> , 2867 , 2868 ,
<code>\biggr</code> , 4908	<code>\currentgrouplevel</code> , 2653	2869 , 2870
<code>\Bigl</code> , 4905	<code>\czas</code> , 6186	<code>\DeclareSymbolFont</code> ,
<code>\bigl</code> , 4903	<code>\czer</code> , 5588 , 5592 , 5593	4455 , 4458 , 4472 ,
<code>\Bigr</code> , 4906	<code>\czerwo</code> , 5587 , 5588	4484 , 4496
<code>\bigr</code> , 4904	<code>\date@left</code> , 6294 , 6305	<code>\DeclareTextCommand</code> ,
<code>\bigskipamount</code> , 5430 , 6284	<code>\date@line</code> , 6281 , 6291 , 6305	3984
<code>\bihyphen</code> , 2980	<code>\dateskipamount</code> , 6279 ,	<code>\DeclareTextCommand </code>
<code>\binoppenalty</code> , 6798	6289	<code>Default</code> ,
<code>\boldmath</code> , 4019	<code>\dc@bsname</code> , 2270 , 2275 ,	3986
<code>\box</code> , 4040 , 4902	2280 , 2324 , 2328 ,	<code>\DeclareUrlCommand</code> , 6921
<code>\bslash</code> , 738 , 1049 , 1131 ,	2368 , 2372	<code>\defLowStarFake</code> , 7027 ,
1142 , 1144 , 1149 ,	<code>\dc@gobblespace@dummy</code> ,	7030
1173 , 2039 , 2041 ,	2224 , 2229 , 2229 , 2231	<code>\defobeylines</code> , 5417
2411 , 2442 , 2443 ,	<code>\dc@grab@afterass</code> ,	<code>\dekbigskip</code> , 5430
2455 , 3631 , 3632 ,	1707 , 1743	<code>\dekfrac</code> , 4265
3633 , 6085 , 6086 ,	<code>\dc@parse@next</code> , 2124 , 2125	<code>\dekfracc@args</code> , 4244 ,
6116 , 6117	<code>\DCcoordinate</code> , <i>p.</i> 19 , 1943	4254 , 4268 , 5723
<code>\bullet</code> , 6196 , 6203	<code>\DCMessagesfalse</code> , <i>p.</i> 21	<code>\dekfraccssimple</code> , 5722 ,
<code>\c@#1</code> , 5900	<code>\DCMessagestrue</code> , <i>p.</i> 21	5728
<code>\c@NoNumSecs</code> , 3456	<code>\DCnocoordinate</code> , <i>p.</i> 19 ,	<code>\dekfraccsslash</code> , 5712 ,
<code>\c@secnumdepth</code> , 3513	1957 , 1965	5725 , 5726
<code>\cat</code> , 3311	<code>\deadcycles</code> , 5797 , 5847	<code>\dekmedbigskip</code> , 5428
<code>\centerthis</code> , 6666	<code>\DeclareCommand</code> , 1605 ,	<code>\dekmedskip</code> , 5429
<code>\chaptermark</code> , 3855	2237 , 2237 , 2276 ,	<code>\deksmallskip</code> , 5426
<code>\chardef</code> , 699	2280 , 2363 , 2390 ,	<code>\Delta</code> , 4767
<code>\chi</code> , 4806	2432 , 2510 , 2787 ,	<code>\delta</code> , 4785
<code>\cipolagwa</code> , 6389 , 6411	2980 , 2982 , 3067 ,	<code>\detoken@xa</code> , 2536 , 2537 ,
<code>\cipolaḡwa</code> , 6402	3235 , 3311 , 3763 ,	2551 , 2552 , 4128
<code>\ClassError</code> , 3630	4158 , 4334 , 4430 ,	<code>\detokenize</code> , 1687 , 2193 ,
<code>\cleardoublepage</code> ,	4436 , 4514 , 4516 ,	2201 , 2214 , 2215 ,
3470 , 6995	4552 , 4604 , 4656 ,	2223 , 2226 , 2227 ,
<code>\clearemptydou </code>	4951 , 4987 , 5139 ,	2737 , 2741 , 4128 ,
<code>blepage</code> ,	5280 , 5349 , 5488 ,	4245 , 4247 , 4309 ,
6994	5861 , 5877 , 6223 ,	4312 , 4320 , 4321 ,
<code>\clubpenalty</code> , 3116 ,	6246 , 6355 , 6384 ,	5236 , 5249 , 5252 ,
3669 , 3674	6401 , 6410 , 6439 ,	5307 , 5308 , 5317 ,
<code>\cmd</code> , 3182	6462 , 6494 , 6573 ,	5318 , 5328 , 5329
<code>\cmd@to@cs</code> , 3182 , 3186	6611 , 6927 , 6953 ,	<code>\dilitkern</code> , 5066
<code>\color</code> , 2931 , 5586 , 5587	6998 , 7030 , 7048	<code>\dimexpr</code> , 4857 , 4862 ,
<code>\continuum</code> , 5622	<code>\DeclareEnvironment</code> ,	4870 , 4901 , 4996 ,
<code>\copy</code> , 4001 , 4032 , 4046 ,	<i>p.</i> 21 , 2390 , 6003	5194 , 5204 , 5215 ,
4995 , 5008	<code>\DeclareFontFamily</code> ,	5262 , 5272 , 5277 ,
	4461 , 4474 , 4486 , 4498	5367 , 5368 , 5503 ,

5506, 6404, 6416,
 6448, 6956, 6960
 \discre, 2961, 2964, 2970,
 2991, 3017, 3034
 \discret, 2964
 \disobeylines, 6478, 7008
 \divide, 4009, 4012, 4013,
 5952, 5953, 5958, 6538
 \do@url@hyp, 6828
 \document, 3375
 \dots, 6413
 \downarrow, 4750
 \dp, 4898, 4901, 4996
 \dywiz, 6178

 \eequals, 5534
 \egfirstofone, 372, 374
 \egRestore@Macro, 1111,
 1113
 \egRestore@MacroSt,
 1111, 1114
 \egStore@Macro, 1031, 1036
 \egStore@MacroSt, 1031,
 1037
 \em, 5110, 5120
 \emdash, 6128
 \emptify, 600, 1627, 1713,
 1902, 2299, 2302,
 2304, 2385, 4194,
 4416, 4512, 4967,
 4971, 5275, 5602,
 5636, 6521, 6659, 6923
 \encodingdefault,
 4456, 4459, 4461,
 4463, 4466, 4473,
 4474, 4476, 4485,
 4486, 4488, 4497,
 4498, 4500, 4503
 \endfilepart, 5853,
 5855, 5861, 5873, 5877
 \endgraf, 6479, 6485,
 6664, 6671, 6976, 7013
 \endlinechar, 305, 5161,
 6682, 6752, 6753, 6759
 \endlist, 3938, 3955
 \englishdate, 6245
 \enoughpage, 5488
 \enspace, 6654
 \ensuremath, 2902, 2918,
 4078, 4328, 4647,
 5623, 6196, 6203
 enumerate*, 3923
 \env, 3101
 \envhack, 2244, 2300,
 2394, 2426
 \envirlet, 1244
 \epsilon, 4786

 \equals, 5532
 \eta, 4789
 \eTeX, 4076, 4081, 4083, 4177
 \everydisplay, 4939,
 4961, 4965
 \everymath, 4913, 4939,
 4942, 4960, 4961,
 4964, 4965
 \everypar, 3665, 3675, 6034
 \exists, 4872

 \f@encoding, 5571
 \f@family, 4421
 \f@series, 4019
 \f@size, 6960
 \fake@onum@ii, 5157, 5159
 \fakeonum, 5139
 \fakern, 4934
 \fakesc@extrascap, 5956, 5971
 \fakesc@extrascap, 5931
 \fakesc@score, 5909, 5933
 \fakesc@extrascap, 5971
 \file, 3039, 6921
 \filepart, 5818, 5820, 5876
 \fontcharhp, 7035
 \fontcharht, 5191, 5267
 \fontcharwd, 5198, 6382
 \fontspec, 766, 5309,
 5310, 5319, 5320,
 5330, 5331, 5358, 5362
 \fooatletter, 376
 \foone, 372, 376, 670, 674,
 677, 681, 683, 685,
 691, 693, 702, 706,
 722, 742, 746, 749,
 752, 772, 774, 925,
 970, 1008, 3105, 3216,
 3220, 3280, 5408,
 5416, 6155, 6180, 7007
 \forall, 4866
 \FormatHangHeading,
 3772, 3780, 3788, 3795
 \FormatRunInHeading,
 3802, 3807
 \freeze@actives, 5556
 \fullpar, 6355
 \fullparcore, 6363, 6367

 \g@emptify, 604, 605
 \g@relaxen, 613, 614
 \gaddtomacro, 571
 \Gamma, 4766
 \gamma, 4784
 \garamath, p. 64, 4987
 \ge, 4689, 4690, 4691, 4703
 \geeng, 4691

 \gemptify, 605
 \geq, 4690, 4701, 4702, 4703
 \getprevdepth, 6480, 6975
 \glet, 563, 3689
 \glueexpr, 5428, 6327,
 6405, 6418
 \gluestretch, 6328
 \glyphname, 5017, 6198
 \gm@clearpagesduetoopenright,
 3469, 3532
 \gm@dontnumbersectionsoutofmain
 3467, 3516
 \gm@gobmacro, 4138, 4144
 \gm@hyperrefstepcounter,
 3459, 3462
 \gm@ifnac, 889, 892
 \gm@ifstar, 935, 1031,
 1111, 2676, 3901,
 4199, 4250, 6275,
 6486, 6603
 \gm@ifundefined, 623,
 1130, 1142, 3483,
 3516, 3532, 3608,
 3629, 3684, 3697,
 3768, 3897, 4055,
 4089, 4091, 4096,
 4098, 4245, 5955, 5978
 \gm@letspace, 988, 996
 \gm@notprerr, 3341, 3348
 \gm@pswords, 3023, 3025,
 3027
 \gm@sec, 3890, 3901, 3902
 \gm@secini, 3854, 3874,
 3880, 3886, 3899
 \gm@secmarkh, 3882
 \gm@secstar, 3858, 3868,
 3875, 3887, 3901, 3902
 \gm@secx, 3890, 3891
 \gm@secxx, 3857, 3885, 3892
 \gm@smarttilde, 6953, 6971
 \gm@straightensec,
 3896, 3905
 \gm@targetheading,
 3460, 3463
 \gm@testdefined, 641,
 659, 2273
 \gm@testundefined, 656
 \gm@UrlFix, 6709, 6714, 6728
 \gm@UrlSetup, 6750, 6796
 \gma, 4980
 \gma@arrowdash, 4994,
 5004, 5010, 5012
 \gma@bare, 4976, 4978
 \gma@bracket, 4978, 4979
 \gma@checkbracket,
 4977, 4981

\backslash gma@dollar, [4975](#), 4976, 4981
 \backslash gma@gmathhook, 4937, 4971, 4972, [4998](#)
 \backslash gma@quantifierhook, 4866, 4872, 4967, 4969, 5006, 5014, 5015
 \backslash gma@tempa, 4855, 4857, 4860, 4862, 4897, 4901, 4929, 4932
 \backslash gma@tempb, 4898, 4901
 \backslash gmath, *p.* 64, [4951](#), 4975, 4979
 \backslash gmath@delc, [4604](#), 4635
 \backslash gmath@delcif, [4629](#)
 \backslash gmath@delimif, [4638](#)
 \backslash gmath@do, [4516](#), 4567, 4570, 4573, 4672, 4674, 4675, 4676, 4677, 4678, 4679, 4680, 4681, 4682, 4683, 4684, 4706, 4707, 4708, 4709, 4710, 4713, 4715, 4717, 4721, 4731, 4732, 4734, 4758
 \backslash gmath@doif, [4552](#), 4589, 4673, 4686, 4689, 4692, 4693, 4720, 4722, 4723, 4724, 4725, 4726, 4727, 4729, 4730, 4736, 4737, 4738, 4739, 4740, 4741, 4742, 4743, 4746, 4750, 4752, 4759, 4762, 4763, 4764, 4766, 4767, 4768, 4769, 4770, 4772, 4773, 4774, 4775, 4776, 4782, 4783, 4784, 4785, 4786, 4787, 4788, 4789, 4790, 4791, 4792, 4793, 4794, 4795, 4796, 4797, 4798, 4799, 4800, 4801, 4802, 4803, 4804, 4805, 4806, 4807, 4808
 \backslash gmath@fam, 4430, 4520, 4558, 4564, 4665, 4778
 \backslash gmath@famm, 4667, 4673, 4686, 4689, 4692, 4693, 4720, 4722, 4723, 4724, 4725, 4726, 4727, 4729, 4730, 4736, 4737, 4738, 4739, 4740, 4741, 4742, 4743, 4746, 4750, 4752, 4759, 4762, 4763, 4764, 4766, 4767, 4768, 4769, 4770, 4772, 4773, 4774, 4775, 4776, 4782, 4783, 4784, 4785, 4786, 4787, 4788, 4789, 4790, 4791, 4792, 4793, 4794, 4795, 4796, 4797, 4798, 4799, 4800, 4801, 4802, 4803, 4804, 4805, 4806, 4807, 4808
4741, 4742, 4743, 4746, 4750, 4752, 4759, 4762, 4763, 4764, 4766, 4767, 4768, 4769, 4770, 4772, 4773, 4774, 4775, 4776, 4780, 4782, 4783, 4784, 4785, 4786, 4787, 4788, 4789, 4790, 4791, 4792, 4793, 4794, 4795, 4796, 4797, 4798, 4799, 4800, 4801, 4802, 4803, 4804, 4805, 4806, 4807, 4808
 \backslash gmath@famnum, 4431, 4527, 4543, 4613, 4620, 4649
 \backslash gmath@font, 4635, 4645, 4756, 4757, 4811, 4812, 4865, 4879, 4882, 4886, 4887, 4888
 \backslash gmath@ft, 4562, 4567, 4570, 4573
 \backslash gmath@getfamnum, 4430, 4523, 4610, 4646
 \backslash gmath@restore, 4575, 4577, 4579, [4585](#)
 \backslash gmath@version, 4565, 4662, 4667, 4778, 4780
 \backslash gmathbase, 4514, 4828, 4830, [4837](#), 4952
 \backslash gmathcats, [4963](#)
 \backslash gmathFams, [4436](#), 4515
 \backslash gmathfamshook, 4509, 4512
 \backslash gmathfurther, [4844](#), 4953
 \backslash gmathhook, 4972
 \backslash gmathscripts, [4959](#)
 \backslash gml@storeCS, 1076, 1099, 1159
 \backslash gml@storemacros, 1077, [1088](#), 1097, 1102, 1162
 \backslash gmobeyspaces, [5409](#)
 \backslash gmshowlists, [2588](#)
 \backslash GMtextsuperscript, [4332](#)
 \backslash gmu@acroinner, 5641, 5651, 5652, 5660
 \backslash gmu@acrospace, 5631, 5640, 5640, 5644
 \backslash gmu@activespace, [772](#), 2922
 \backslash gmu@activespaceblank, 2906, [2921](#)
 \backslash gmu@among@, [408](#), 408, 410
 \backslash gmu@AT@ampulex, 6627, 6629, 6633
 \backslash gmu@ATfootnotes, 6620, 6626, 6636, 6639, 6643
 \backslash gmu@basefont, 5345, 5360
 \backslash gmu@bihyphen@char, 2985, 2994
 \backslash gmu@bihyphen@corr, 2989, 3001, 3015
 \backslash gmu@calc@scale, 4398, 4403, 5349
 \backslash gmu@calculateslant, 5183, 5259
 \backslash gmu@cepstnof, [5280](#), 5358, 5362
 \backslash gmu@checkaftersec, 3683, 3742
 \backslash gmu@checkM, 968, 972
 \backslash gmu@currfont@descaled, 5359, 5369
 \backslash gmu@datef, 6309
 \backslash gmu@datefsl, [6223](#), 6227, 6246, 6250, 6272
 \backslash gmu@dekfrac, [4222](#), 4246, 4250
 \backslash gmu@dekfraccsimple, 4250, 5709, 5723
 \backslash gmu@denominatorkern, 4224, 4272, 5712
 \backslash gmu@discretionaryhyphen, 2976, 2982, 3010, 3011, 6757
 \backslash gmu@discretionaryslash, 3034, 3045
 \backslash gmu@disMinner, 7011, [7012](#)
 \backslash gmu@dofakeonum, 5163, 5169
 \backslash gmu@dogmathbase, 4656, 4827, 4828
 \backslash gmu@doUrlMath, 6801, 6803, 6805, [6849](#)
 \backslash gmu@doUrlMathAc, 6806, 6813, 6822, 6831, [6880](#)
 \backslash gmu@dywiz, [6177](#), 6181
 \backslash gmu@extremnum, [3130](#), 3148, 3149, 3153, 3154
 \backslash gmu@fileext, 5758, 5768, 5786
 \backslash gmu@filename, 5757, 5771, 5783, 5786, 5789, 5798
 \backslash gmu@filepartname, 5825, 5833, 5848, 5865
 \backslash gmu@fnhook, 6653, 6659
 \backslash gmu@fontscale, 4408, 5364

<code>\gmu@fontscalebr,</code>	<code>\gmu@nl@reserveda,</code>	<code>\gmu@setheading,</code> <u>3739</u> ,
4408, 4409, 4416,	1231, 1234, 1239, 1242	3745, 3746
4464, 4477, 4489, 4501	<code>\gmu@nocite@ampulex,</code>	<code>\gmu@setsetSMglobal,</code>
<code>\gmu@fontstring,</code> <u>4379</u> ,	<u>3366</u> , 3379	1075, <u>1080</u> , 1158
4464, 4477, 4480,	<code>\gmu@numeratorkern,</code>	<code>\gmu@setSMglobal,</code> 1082,
4489, 4501, 4508	4223, <u>4271</u> , 5711	1084, 1102
<code>\gmu@fontstring@,</code>	<code>\gmu@PDFdetector,</code> 4311,	<code>\gmu@SMdo@scope,</code> 1201,
4380, <u>4382</u> , 4671	4321	1203, 1206, 1207, 1221
<code>\gmu@fontstring@@,</code>	<code>\gmu@pdfdetector,</code>	<code>\gmu@SMdo@setscope,</code>
4383, 4391	4308, 4320	1199, <u>1205</u> , 1219
<code>\gmu@forallkerning,</code>	<code>\gmu@prevsec,</code> 3664,	<code>\gmu@SMglobalfalse,</code>
4867, 4869	3666, 3688, 3695, 3725	1043, 1057, 1084,
<code>\gmu@forarg,</code> 424, <u>434</u>	<code>\gmu@printslashes,</code>	1093, 1121, 1152, 1209
<code>\gmu@foreach,</code> 423, <u>431</u> ,	3039, <u>3041</u> , 3041,	<code>\gmu@SMglobaltrue,</code>
431, 433, 438	3043, 3046	1019, 1082
<code>\gmu@forer,</code> <u>432</u> , 436, 438	<code>\gmu@quiettrue,</code> 215	<code>\gmu@smtempa,</code> 1047,
<code>\gmu@fracfontsetup,</code>	<code>\gmu@resa,</code> 5602, 5603, 5605	1056, 1146, 1151
5711, 5715, <u>5718</u>	<code>\gmu@reserveda,</code> <u>392</u> ,	<code>\gmu@star@loop,</code> 7050,
<code>\gmu@getaddvs,</code> <u>3734</u> ,	393, 458, 459, 985,	<u>7052</u> , 7056
3734, 3740	987, 993, 995, <u>1089</u> ,	<code>\gmu@storeifnotyet,</code>
<code>\gmu@gettext,</code> <u>5756</u> , 5766	1092, 1095, 1635,	<u>1135</u> , 3008, 4561,
<code>\gmu@getfontdata,</code>	1640, 2193, 2194,	4696, 4701, 4851,
<u>4415</u> , 4451, 4470,	2195, 2196, 2338,	4874, 4893, 4916,
4482, 4493	2352, 2426, 2429,	4928, 4999
<code>\gmu@getfontscale,</code>	2433, 2452, 2463,	<code>\gmu@storespecials,</code> <u>2787</u>
<u>4393</u> , 4394, 4410,	2737, 2742, 3640,	<code>\gmu@stripchar,</code> <u>4428</u> ,
4420, 4422	3689, 3690, 3693,	4431, 4814
<code>\gmu@getfontstring,</code>	<u>3979</u> , <u>3981</u> , 3983,	<code>\gmu@tempa,</code> 264, 860,
4378, 4423	3984, 3985, 5299, 5337	862, <u>924</u> , 934, 938,
<code>\gmu@getslant,</code> <u>5248</u> , 5260	<code>\gmu@reservedb,</code> 2741,	944, 2011, 2029,
<code>\gmu@gobdef,</code> 257, <u>263</u>	2742, 5307, 5317,	2365, 2369, 2371,
<code>\gmu@hashes,</code> <u>1988</u> , 1992,	5328, 5339	2382, 2530, 2536,
2023	<code>\gmu@reservedc,</code> <u>5288</u> ,	2551, 2559, 2736,
<code>\gmu@hashesbraced,</code>	5320, 5322, 5331, 5333	2737, 2740, 2741,
<u>1997</u> , 2000, 2025	<code>\gmu@reservedd,</code> 5291, 5303	4143, 4145, 4300,
<code>\gmu@if,</code> <u>287</u> , 1704, 1706	<code>\gmu@reservede,</code> <u>5295</u> , 5305	4302, 4304, 4322,
<code>\gmu@if@onum,</code> <u>5152</u> , <u>5234</u>	<code>\gmu@restorespecials,</code>	<u>4395</u> , 4397, <u>4401</u> ,
<code>\gmu@ifstored,</code> 1118,	2790	4402, 4453, 4467,
<u>1129</u> , 1137, 4470,	<code>\gmu@restoreUpUpUp,</code>	4471, 4478, 4483,
4482, 4531	6724, 6725	4490, 4494, 4504,
<code>\gmu@lowstar,</code> <u>910</u> , 930,	<code>\gmu@RPfor,</code> 5586, <u>5600</u> ,	4525, 4532, 4533,
7024, 7055	<u>5612</u> , 5623	4535, 4536, 4540,
<code>\gmu@lowstarfake,</code> 911,	<code>\gmu@scalar,</code> 5943, <u>5947</u> ,	4547, 4588, 4591,
7034, 7040	5948, 5954	4612, 4614, 4615,
<code>\gmu@luzniej,</code> <u>6017</u> ,	<code>\gmu@scalematchX,</code>	4618, 4624, <u>5028</u> ,
6020, 6022	5933, 5945, 5969	5031, 5033, <u>5078</u> ,
<code>\gmu@maxnum,</code> <u>3153</u>	<code>\gmu@scapLetters,</code>	5082, 5098, 5102,
<code>\gmu@maybestripcomma,</code>	<u>5905</u> , 5915, 5920	5154, 5164, <u>5181</u> ,
5289, 5343	<code>\gmu@scapSpaces,</code> <u>5918</u> ,	5193, 5198, <u>5204</u> ,
<code>\gmu@measurewd,</code> 5185,	5923, 5927	5207, 5208, 5209,
5197, 5213, 5227,	<code>\gmu@scapss,</code> <u>5926</u> , 5931	<u>5211</u> , 5219, 5220,
6811, 6816, 6829	<code>\gmu@scscale,</code> 5962, 5968	5221, 5222, 5223,
<code>\gmu@medmuskip,</code> 6782, 6788	<code>\gmu@septify,</code> <u>2793</u> ,	<u>5225</u> , 5230, 5231,
<code>\gmu@minnum,</code> 3116, <u>3154</u>	4964, 6164	5235, 5236, 5249,
<code>\gmu@ms,</code> 2007, 2013, 2017,	<code>\gmu@setbasefont,</code>	5255, 5262, 5266,
2021, 2038, 2039,	<u>5345</u> , 5347	5272, 5280, 5358,
2040, 2041, 2042		5362, 5978, 5980,

5981, 5982, 6404, 6405, 6414, 6415, 6417, 6418, 6549, 6553, 6812, 6815, 6817, 6820, 6830, 6837	<code>\gmu@tempb</code> , 926, 941, 2531, 2537, 2552, 2560, 4396, 4397, 4402, 5192, 5204, 5216, 5228, 5250, 5251, 5254, 5255, 6814, 6819, 6832	<code>\gmu@tempc</code> , 2532, 2566, 5191, 5194	<code>\gmu@tempd</code> , 2534, 2535, 2538, 2543, 2548, 2552, 2557, 2558, 2570, 2577, 5195, 5271, 5275, 5276, 5277	<code>\gmu@tempe</code> , 2547, 2555, 2574, 2578	<code>\gmu@tempf</code> , 2573, 2577	<code>\gmu@theendlinechar</code> , 6752, 6759	<code>\gmu@theskewchar</code> , 4424, 4462, 4475, 4487, 4499	<code>\gmu@thickmuskip</code> , 6783, 6789	<code>\gmu@thinmuskip</code> , 6781, 6787	<code>\gmu@thou@fiver</code> , 6517, 6520, 6528, 6528	<code>\gmu@thou@i</code> , 6522, 6543, 6551	<code>\gmu@thou@ii</code> , 6522, 6543, 6551	<code>\gmu@thou@o</code> , 6522, 6543, 6551	<code>\gmu@thou@put</code> , 6521, 6525, 6540	<code>\gmu@thou@putter</code> , 6524, 6534, 6541, 6548, 6556	<code>\gmu@thousep</code> , 6553, 6560	<code>\gmu@tilde</code> , 5380, 5396	<code>\gmu@twostring</code> , 2110, 2118, 2136	<code>\gmu@url@flexbreak</code> , 6908, 6911	<code>\gmu@url@rigidbreak</code> , 6906	<code>\gmu@url@breakable</code> , 6869, 6911	<code>\gmu@whonly</code> , 5804, 5805	<code>\gmu@xedekfracplain</code> , 4199, 4253	<code>\gmu@xedekfracstar</code> , 4199, 4214	<code>\gmu@xfraccdef</code> , 4215, 4229, 4230, 4231, 4232, 4233, 4234, 4235, 4236, 4237, 4238, 4239, 4240, 4241, 4242, 4243	<code>\gmv@hyphen</code> , 3083	<code>\gmv@hyphenchar</code> , 3089, 6764	<code>\gn@melet</code> , 1238	<code>\gobble</code> , 247, 249, 2330, 4930	<code>\gobblespace</code> , 1676, 2223, 2226, 2227, 2231	<code>\gobbletwo</code> , 250	<code>\grab@default</code> , 1700, 1728, 1739, 1740, 1745, 1746	<code>\grab@defaults</code> , 1702, 1736	<code>\grab@prefix</code> , 1669, 1763	<code>\grefstepcounter</code> , 530, 546	<code>\grelaxen</code> , 614, 3666	<code>\hash</code> , 3109	<code>\hathat</code> , 3106	<code>\HeadingNumber</code> , 3542, 3544	<code>\HeadingNumbered!</code> false, 3468, 3513	<code>\HeadingRHeadText</code> , 3526	<code>\HeadingText</code> , 3528	<code>\HeadingTOCText</code> , 3527	<code>\HeShe</code> , 3433	<code>\heshe</code> , 3428	<code>\hfillneg</code> , 5433	<code>\hgrefstepcounter</code> , 545	<code>\hidden@iffalse</code> , 508, 2541	<code>\hidden@iftrue</code> , 509, 2541	<code>\HimHer</code> , 3435	<code>\himher</code> , 3430	<code>\HisHer</code> , 3434	<code>\hisher</code> , 3429	<code>\HisHers</code> , 3436	<code>\hishers</code> , 3431	<code>\hrule</code> , 3713	<code>\hsize</code> , 6382, 6416, 6664, 6668, 6669, 6956	<code>\hunskip</code> , 553, 5532, 5534, 5536, 6368, 6403, 6412	<code>\HyOrg@maketitle</code> , 6628, 6629	<code>\hyphenpenalty</code> , 3027, 6103, 6334	<code>\if@afterindent</code> , 3670	<code>\if@dc@alllong@</code> , 1586, 1671	<code>\if@dc@ignore</code> , 1601, 1679, 1685, 1692	<code>\if@dc@quiet@</code> , 1587, 2321	<code>\if@filesw</code> , 5478, 5770, 5782, 5790, 5826, 5837, 5866	<code>\if@gmu@mmhbox</code> , 4248, 4258, 4262, 5716, 6583	<code>\if@mainmatter</code> , 3468	<code>\if@nobreak</code> , 3667	<code>\if@openright</code> , 3470	<code>\if@specialpage</code> , 3531	<code>\if@twoside</code> , 3557	<code>\IfAmong</code> , 399, 424, 458, 1699, 1701, 1861, 1920, 1945, 1952, 2195, 2300, 2306, 2307	<code>\ifcsname</code> , 629, 2119, 2137, 2681, 4587, 5900	<code>\ifdate</code> , 6269, 6279	<code>\IfDCMessages</code> , p. 21, 1590, 2274, 2279, 2323	<code>\ifdefined</code> , 190, 255, 647, 698, 763, 1582, 3081, 3083, 3089, 3288, 4151, 4695, 4700, 4940, 5613, 6093, 6154, 6400, 6425, 6628, 6648, 6743, 6764, 6774	<code>\iffontchar</code> , 911, 4077, 4216, 4567, 4570, 4573, 4635, 4645, 4756, 4757, 4811, 4865, 4879, 4882, 4886, 4887, 4888, 6196, 6808, 6824	<code>\ifgmu@postsec</code> , 3685, 3724, 3732	<code>\ifgmu@quiet</code> , 213, 1122, 1143	<code>\ifgmu@SMglobal</code> , 1017, 1041, 1048, 1081, 1119, 1147, 1206	<code>\ifHeadingNumbered</code> , 3512, 3540	<code>\IfIntersect</code> , 412, 1764, 1766, 2290, 2293, 2294, 2449	<code>\IfIntersect@next</code> , 422, 425, 426	<code>\IfLong</code> , p. 20, 2186	<code>\IfNoValueF</code> , 2174, 2181	<code>\IfNoValueT</code> , 2172, 2184	<code>\IfNoValueTF</code> , 2162, 2172, 2174, 2176	<code>\ifSecondClass</code> , 5695
--	--	---	--	---	--------------------------------------	--	--	---	--	--	--	---	--	--	--	--	--------------------------------------	---	---	--	---	---------------------------------------	--	---	---	---------------------------------	--	-------------------------------	---	---	-------------------------------	---	--	--	--	------------------------------------	---------------------------	-----------------------------	--	--	---------------------------------------	----------------------------------	-------------------------------------	----------------------------	----------------------------	-------------------------------	--------------------------------------	--	---	-----------------------------	-----------------------------	-----------------------------	-----------------------------	------------------------------	------------------------------	----------------------------	---	--	---	---	-------------------------------------	---	--	---	--	---	------------------------------------	---------------------------------	-----------------------------------	-------------------------------------	---------------------------------	--	---	-----------------------------------	---	--	--	---	--	---	---	---	---	------------------------------------	---------------------------------------	---------------------------------------	---	------------------------------------

<code>\IfValueF</code> , 2184, 3767, 6229, 6252	<code>\leftrightharrow</code> , 4743, 4889	<code>\mathbin</code> , 4672, 4673, 4720, 4722, 4723, 4729, 4739, 4762, 4763, 4764, 4880, 4883, 4925, 4926, 5005, 5014
<code>\IfValueT</code> , 2178, 2181, 2428, 2575, 2791, 2983, 2987, 3000, 3248, 3250, 3251, 3252, 3253, 4163, 4442, 4457, 4465, 4469, 4502, 4586, 4660, 4954, 5156, 6005, 6010, 6230, 6237, 6238, 6253, 6261, 6262, 6360, 6391, 6499, 6617, 6780	<code>\leftslanting</code> , 5123, 5125	<code>\mathchar@type</code> , 4526, 4542, 4648, 4658, 4949
<code>\IfValueTF</code> , 2176, 2375, 2575, 2991, 2993, 4335, 4454, 4495, 4524, 4569, 4572, 4611, 5504, 6360, 6392, 6445, 6475, 6501, 6577, 6955	<code>\leftslanting@</code> , 5057, 5123, 5125	<code>\mathchoice</code> , 4853, 4876, 4917, 5001
<code>\ignoreactiveM</code> , 967, 974	<code>\leq</code> , 4687, 4696, 4697, 4698	<code>\mathclose</code> , 4713, 4717, 4737, 4904, 4906, 4908, 4910, 4923
iI, p. 20	<code>\lim</code> , 4819	<code>\mathfrak</code> , 5623
<code>\ikern</code> , 5736	<code>\linebreak</code> , 6477	<code>\mathit</code> , 4466
<code>\IMHO</code> , 5671	<code>\linedate</code> , 6275, 6291, 6306	<code>\mathop</code> , 4653, 4731, 4853
<code>\in</code> , 4752, 5014	<code>\linedate@</code> , 6275, 6276, 6277	<code>\mathopen</code> , 4710, 4715, 4736, 4903, 4905, 4907, 4909, 4922
<code>\inclasthook</code> , 5787, 5809, 5863	<code>\linedate@@</code> , 6275, 6276	<code>\mathord</code> , 4675, 4676, 4677, 4678, 4679, 4680, 4681, 4682, 4683, 4684, 4724, 4725, 4730, 4738, 4758, 4759
<code>\includecountfix</code> , 5890	<code>\linedate@hook</code> , 6278, 6287	<code>\mathpunct</code> , 4706, 4707, 4708, 4709
<code>\indent</code> , 6651	<code>\list</code> , 3929, 3947	<code>\mathrel</code> , 4674, 4686, 4689, 4692, 4693, 4721, 4726, 4727, 4732, 4734, 4740, 4741, 4742, 4743, 4746, 4750, 4752, 4889, 4931, 5004, 5010, 5011, 5012, 5013, 5017
<code>\infty</code> , 4725	<code>\listparindent</code> , 3934, 3952	<code>\mathrm</code> , 4503, 4653, 4864
<code>\iota</code> , 4792	<code>\lit</code> , 5088	<code>\mathversion</code> , 4660
<code>\itemindent</code> , 3931, 3949	<code>\litcorrection</code> , 5059, 5070, 5077, 5092, 5097	<code>\max</code> , 4817
<code>itemize*</code> , 3941	<code>\litdimen</code> , 5058, 5060, 5065, 5066	<code>\medmuskip</code> , 2969, 6782, 6788, 6797
<code>\iteracro</code> , 5627, 5638	<code>\litkern</code> , 5061, 5066	<code>\meta</code> , 2894, 2945, 3192, 3200, 3207, 3211
<code>\justified</code> , 6343	<code>\litshape</code> , 5068, 5086, 5088, 5113	<code>\meta@font@select</code> , 2907, 2933
<code>\kappa</code> , 4793	<code>\liturgiques</code> , 5585	<code>\meta@fontsetting</code> , 2902, 2918, 2931, 2933, 2948
<code>\labelsep</code> , 3933, 3951	<code>\longafterfi</code> , 353, 355	<code>\metachar</code> , 2947, 3249
<code>\labelwidth</code> , 3932, 3933, 3950, 3951	<code>\longpauza</code> , 6131, 6132, 6139, 6140	<code>\metacharfont</code> , 2947, 2948
<code>\Lambda</code> , 4769	<code>\looseness</code> , 6023, 6034, 6360, 6392	<code>\min</code> , 4818
<code>\lambda</code> , 4794	<code>\lozenge</code> , 4758	<code>\mkern</code> , 4934
<code>\larger</code> , p. 41, 2867, 4856, 4861, 4903, 4904, 4907, 4908, 4909, 4910	<code>\lpauza</code> , 6072	<code>\mskip</code> , 2969
<code>\largerr</code> , p. 41, 2871, 4905, 4906	<code>\lsl</code> , 5091	<code>\mu</code> , 4795
<code>\LaTeXe</code> , 3965, 4018	<code>\LuaTeX</code> , 4109	<code>\multiply</code> , 4008, 4011, 5960, 6022, 6033, 6361, 6393
<code>\LaTeXpar</code> , 4029	<code>luzniej</code> , 6027	
<code>\ldate</code> , 6303, 6313	<code>luzniej*</code> , 6032	
<code>\le</code> , 4686, 4687, 4688, 4698	<code>\luzniejcore</code> , 6019, 6027	
<code>\leeng</code> , 4688	<code>\macro</code> , 4144	
<code>\leftarrow</code> , 4741, 5010	<code>\makestarlow</code> , 7022	
<code>\leftline</code> , 6445, 6500	<code>\MakeUppercase</code> , 5909	
<code>\leftmargin</code> , 3930, 3948	<code>\mapsto</code> , 5003	
	<code>\marg</code> , 3192, 3229, 3252	
	<code>maszynopis</code> , 6325	
	<code>\math@arg</code> , 3282, 3283	
	<code>\mathalpha</code> , 4766, 4767, 4768, 4769, 4770, 4772, 4773, 4774, 4775, 4776, 4782, 4783, 4784, 4785, 4786, 4787, 4788, 4789, 4790, 4791, 4792, 4793, 4794, 4795, 4796, 4797, 4798, 4799, 4800, 4801, 4802, 4803, 4804, 4805, 4806, 4807, 4808	

<code>\mw@getflags</code> , 3688	<code>\next@dc</code> , 1669, 1700,	<code>\PassOptionsToPack </code>
<code>\mw@HeadingBreakAfter</code> ,	1702, 1705, 1707,	age,
3533, 3553, 3568,	1708, 1712, 1716, 1732	4163
3572, 3603, 3689	<code>\NextBgroup</code> , 5856	<code>\pauza</code> , 6055
<code>\mw@HeadingBreakBefore</code> ,	<code>\nfss@text</code> , 2903	<code>\pauza@skipcore</code> , <u>6045</u> ,
3530, 3602, 3690	<code>\nieczer</code> , <u>5594</u>	6100, 6101, 6102
<code>\mw@HeadingLevel</code> , 3510,	<code>\nobeyspaces</code> , <u>5926</u> ,	<code>\pauzacore</code> , 6046, 6059,
3513	6955, 6965	6068, 6074, 6103,
<code>\mw@HeadingRunIn</code> ,	<code>\nocite</code> , 3372	6109, <u>6131</u> , <u>6134</u> ,
3548, 3602	<code>\nofileparts</code> , <u>5875</u>	<u>6139</u> , <u>6142</u> , <u>6337</u> , <u>6338</u>
<code>\mw@HeadingType</code> , 3529,	<code>\nohy</code> , <u>5740</u>	<code>\pauzadial</code> , 6061, <u>6067</u>
3664, 3696, 3697, 3710	<code>\nolimits</code> , 4868, 4872	<code>\pdef</code> , <u>240</u> , 254, 304, 386,
<code>\mw@HeadingWholeWidth</code> ,	<code>\nolinkurl</code> , 6603	399, 442, 472, 489,
3551, 3603	<code>NoNumSecs</code> , <u>3456</u>	508, 509, 530, 545,
<code>\mw@normalheading</code> ,	<code>\not@onlypreamble</code> ,	553, 794, 836, 887,
3555, 3564, 3567,	<u>3328</u> , 3332, 3333,	910, 935, 951, 960,
3571, 3745	3334, 3335, 3336, 4830	967, 972, 1009, 1019,
<code>\mw@processflags</code> , 3604	<code>\NoValue</code> , 1740, 1778,	1030, 1073, 1110,
<code>\mw@runinheading</code> ,	1786, 1821, 1828,	1135, 1155, 1177, 1181,
3549, 3746	1888, 1940, 1976,	1244, 1605, 1755,
<code>\mw@secdef</code> , 3609, 3610,	2156, 2157, <u>2160</u> ,	2186, 2514, 2722,
3611, <u>3617</u>	2168, 2206, 2224,	2793, 2831, 2871,
<code>\mw@section</code> , 3608	4437, 4514, 4559,	2872, 2894, 2921,
<code>\mw@sectionxx</code> , <u>3509</u>	4656, 4951	2947, 2961, 2964,
<code>\mw@secundef</code> , 3613,	<code>\nu</code> , 4796	3017, 3032, 3039,
3625, <u>3628</u>	<code>\numexpr</code> , 1993, 2001,	3053, 3093, 3101,
<code>\mw@setflags</code> , 3614	2007, 2023, 2025,	3106, 3192, 3198,
	2038, 3115, 3144,	3200, 3201, 3205,
<code>\n@melet</code> , 1230, 1245,	3312, 5262, 5266,	3210, 3283, 3460,
1246, 2214, 2226,	5267, 5368, 5369,	3967, <u>3987</u> , 4179,
3639, 3641, 3899, 4219	6187, 6188, 6200,	4191, 4244, 4254,
<code>\nameshow</code> , 2593	6539, 7054, 7057	4265, 4294, 4325,
<code>\nameshowthe</code> , 2594		4652, 4844, 4959,
<code>\napapierkicore</code> , <u>5995</u> ,	<code>\oarg</code> , 3198	4963, 5035, 5057,
6007	<code>\obeyspaces</code> , 772	5068, 5085, 5088,
<code>\napapierkistretch</code> ,	<code>\oldLaTeX</code> , 3964	5091, 5110, 5119,
5993, 5996	<code>\oldLaTeXe</code> , 3965	5123, 5248, 5345,
<code>\narrativett</code> , 3087,	<code>\Omega</code> , 4776	5389, 5532, 5534,
3098, 3192, 3200,	<code>\omega</code> , 4808	5536, 5628, 5665,
3207, 3211		5674, 5740, 5741,
<code>\narra </code>	<code>\PackageE</code> , 6986	5899, 5931, 6055,
<code>tivett@storedhyphench</code>	<code>\PackageError</code> , 3347,	6067, 6072, 6083,
3088, 3095	6084, 6115	6096, 6114, 6193,
<code>\nawj</code> , 6036	<code>\PackageWarning</code> , 1123,	6222, 6245, 6271,
<code>\nazwired</code> , <u>6208</u>	1144, 2275, 2858,	6275, 6276, 6277,
<code>\ne</code> , 4728	2862, 6718	6291, 6303, 6316,
<code>\neb</code> , 4927	<code>\pagebreak</code> , 3556, 3568, 3572	6319, 6343, 6367,
<code>\neg</code> , 4729, 4933	<code>\pagegoal</code> , 5503, 5506	6513, 6568, 6603,
<code>\nego</code> , 4730	<code>\pagestyle</code> , 6995	6621, 6633, 6643,
<code>\neq</code> , 4727, 4728, 4926	<code>\pagetotal</code> , 5503, 5506	6647, 6661, 6666,
<code>\neqb</code> , 4926, 4927	<code>\paragraph</code> , 6309	6681, 6708, 6713,
<code>\newcount</code> , 1598, 1599, 6017	<code>\paragraphmark</code> , 3855	6718, 6906, 6908,
<code>\newcounter</code> , 3456, 5900	<code>\ParanoidPostsec</code> , <u>3721</u>	6975, 6986, 7011
<code>\newgif</code> , 472	<code>\parg</code> , 3205	<code>\pdfTeX</code> , 4083
<code>\newskip</code> , 6426, 6491	<code>\partial</code> , 4738	<code>\pdfLaTeX</code> , 4086
<code>\newtoks</code> , 1583, 1596	<code>\partmark</code> , 3855	<code>\pdfTeX</code> , <u>4085</u>
<code>\newwrite</code> , 5478	<code>\partopsep</code> , 3930, 3948	<code>\Phi</code> , 4774

\backslash phi, 4805
 \backslash Pi, 4799
 \backslash pi, 4798
 \backslash pk, 3053
 \backslash PlainTeX, 4068
 \backslash pm, 4739
 \backslash polksadata, 6222, 6242
 \backslash possfil, 3112
 \backslash possvfil, 3114, 6280, 6284
Pp!lL\long\par, p. 20
 \backslash ppauza, 6083
 \backslash ppauza@dash, 6051, 6089, 6123, 6135, 6143
 \backslash ppauza@skipcore, 6050, 6120, 6121
 \backslash pprovide, 269, 5622, 6198
 \backslash prec, 5015
 \backslash prependtomacro, 589, 5157, 6620
 \backslash prevdepth, 6977
 \backslash printsaces, 3023, 3032
 \backslash protected, 240, 269, 484, 503, 1630, 2244, 2319, 2394, 2511, 4647, 5031, 5033, 5234, 6982
 \backslash provide, 254, 269
 \backslash providecounter, 5893, 5899
 \backslash Psi, 4775
 \backslash psi, 4807
 \backslash PutIfValue, 2178, 4456, 4459, 4461, 4463, 4472, 4473, 4474, 4476, 4479, 4484, 4485, 4486, 4488, 4497, 4498, 4500, 4507, 4662, 6229, 6238, 6252, 6262
 \backslash quad, 6208
 \backslash quantifierhook, 4968, 4992
quiet, 215
 \backslash rdate, 6291
 \backslash real, 5954, 5956
 \backslash reflectbox, 4093, 4100
 \backslash relaxen, 609, 2305, 3015, 3599, 6167, 6522, 6710, 6715
 \backslash relpenalty, 6798
 \backslash relsize, p. 41, 2831, 2832, 2867, 2868, 2869, 2870, 2871, 2872
 \backslash renewcommand, 5612
 \backslash RequirePackage, 4165, 5475, 5605, 5938
 \backslash resizebox, 4299, 5193
 \backslash resizegraphics, 4294
 \backslash Restore@Macro, 1113, 1116, 1159, 1169
 \backslash Restore@Macros, 1155, 1157
 \backslash Restore@MacroSt, 1114, 1141
 \backslash RestoreCatM, 6982, 6986
 \backslash RestoreEnvironment, 1181
 \backslash RestoreMacro, 1110, 1183, 1757, 3865, 3969, 4172, 4174, 4177, 4598, 4658, 5855
 \backslash RestoreMacros, 1155
 \backslash restoreparindent, 6429
 \backslash RestoringDo, 1218
 \backslash rho, 4800
 \backslash rightarrow, 4742, 5012
 \backslash rightline, 6291, 6453
 \backslash rmopname, 4652, 4817, 4818, 4819, 4820, 4821, 4822, 4823, 4824, 4825
 \backslash romannumeral, 2007, 2038, 3928, 3946, 5137
 \backslash romorzero, 5136, 5174, 5179
 \backslash rotatebox, 4866, 4872, 4880, 4883, 5016
 \backslash rrthis, 6661
 \backslash rs@size@warning, 2850, 2855, 2858
 \backslash rs@unknown@warning, 2845, 2862
 \backslash runindate, 6308
 \backslash scantnoline, 6681
 \backslash scantokens, 306, 5164, 5172, 6682, 6758
 \backslash scantwo, 304
 \backslash scanverb, 3249
 \backslash scshape, 4066, 5005
 \backslash secondclass, 5694
 \backslash SecondClasstrue, 5696
 \backslash sectionmark, 3855
 \backslash setbasefont, 5347
 \backslash SetMathAlphabet, 4466, 4503
 \backslash setprevdepth, 6457, 6505, 6977
 \backslash SetSectionFormat|ting, 3599, 3600, 3764, 3769, 3778, 3786, 3793, 3800, 3805
 \backslash setspaceskip, 6998
 \backslash SetSymbolFont, 4454, 4495
 \backslash SetTwoheadSkip, 3748, 3777, 3785, 3792
 \backslash sfname, 3032, 3042
 \backslash shortleftarrow, 5011
 \backslash shortpauza, 6133, 6141
 \backslash shortrightarrow, 5013
 \backslash shortthousep, 6572
 \backslash showboxbreadth, 2588
 \backslash showboxdepth, 2588
 \backslash ShowFont, 5570
 \backslash showlists, 2588
 \backslash showthe, 2594
 \backslash Sigma, 4772
 \backslash sigma, 4801
 \backslash sim, 4740
 \backslash sin, 4820
 \backslash skewchar, 4424, 4462, 4475, 4487, 4499
 \backslash SliTeX, 4065
 \backslash smaller, p. 41, 2868, 5665, 5719, 6622
 \backslash smallerr, p. 41, 2872, 5969
 \backslash smallskipamount, 5426, 5427, 5428
 \backslash smartunder, 724
 \backslash SMglobal, 1019
 \backslash spaceskip, 7005
 \backslash sphack, 2244, 2294, 2394, 2449
 \backslash spifletter, 833, 6584
 \backslash sqrtsign, 4813
 \backslash square, 4759
 \backslash Store@Macro, 1036, 1039, 1076
 \backslash Store@Macros, 1073, 1074
 \backslash Store@MacroSt, 1037, 1046
 \backslash StoreCatM, 6981, 6987
 \backslash Stored@Macro, 1168, 1169
 \backslash storedcsname, 1172, 3864, 4470, 4482, 5592, 6603
 \backslash StoredMacro, 1168
 \backslash StoreEnvironment, 1177
 \backslash StoreMacro, 1030, 1137, 1179, 1753, 3968, 4022, 4023, 4081, 4164, 4531, 4949, 5591, 5873, 6337, 6602
 \backslash StoreMacros, 1073, 3855
 \backslash StoringAndRelax|ingDo, 1198
 \backslash strip@pt, 5058, 5073, 5079, 5099, 5277, 5365
 \backslash subs, 671, 726
 \backslash subsectionmark, 3855

<code>\subsubsectionmark</code> , 3855	<code>\udigits</code> , 4191, 4194	<code>\verb</code> , 4164, 4172
<code>\sum</code> , 4865	<code>\undeksmallskip</code> , 5427	<code>\verbatim@mathhack</code> ,
<code>\sup</code> , 675	<code>\unexpanded</code> , 225, 580,	6746, 6773
<code>\symgmathroman</code> , 4814	591, 1639, 2322,	<code>\verbatim@mathhack@</code> ,
	2349, 2366, 2373,	6775, 6779
<code>\tan</code> , 4824	2375, 2412, 2413,	<code>\verbatim@specials</code> ,
<code>\tau</code> , 4803	2461, 2539, 2559,	3081, 6743, 6745
<code>\TB</code> , 4074	2560, 2564, 2565,	<code>\verba </code>
<code>\TeXbook</code> , 4073, 4074	2568, 3084, 3640,	<code>tim@specials@list</code> ,
<code>\textbullet</code> , 6193, 6203	4306, 4309, 4311,	6774, 6775
<code>\textcolor</code> , 5594	4312, 4315, 4589,	<code>\vfil</code> , 3114
<code>\textlarger</code> , 2869	6540, 6814, 6819,	<code>\vfilneg</code> , 3128
<code>\textlit</code> , 5085	6831, 6832	<code>\visiblespace</code> , 764, 768,
<code>\textsl</code> , 4073	<code>\unless</code> , 659, 1122, 1143,	3017, 6737
<code>\textsmaller</code> , 2870	1582, 1692, 1712,	<code>\Vs</code> , 5031
<code>\textstyle</code> , 4020	2119, 2137, 2321,	<code>\vs</code> , 3017, 3023, 3027
<code>\textsuperscript</code> ,	4152, 4156, 4879,	
4330, 4334, 4343, 6622	4882, 4886, 5506,	<code>\wd</code> , 3999, 4002, 4033,
<code>\textsuperscript@@</code> ,	5613, 5900, 6425	4038, 4046, 4047,
4325, 4330, 4335, 4343	<code>\uparrow</code> , 4746	4870, 4995, 4996,
<code>\texttilde</code> , 5389	<code>\upshape</code> , 4470, 4493, 5114	5008, 5009, 6414
<code>\textvisiblespace</code> , 766	<code>\Upsilon</code> , 4773	<code>\Web</code> , 4070
<code>\textwidth</code> , 6448, 6467, 6475	<code>\upsilon</code> , 4804	<code>\wedge</code> , 4763, 4883, 4933
<code>\tg</code> , 4822	<code>\Url</code> , 4940, 4941	<code>\whenonly</code> , 5803
<code>\thedata</code> , 6313	<code>\url</code> , 6602, 6603	<code>\whern</code> , 6484
<code>\Theta</code> , 4768	<code>\Url@Format</code> , 6741, 6918	<code>\whern@parbox</code> , 6455,
<code>\theta</code> , 4790	<code>\Url@HyperHook</code> , 6915	6462, 6501, 6502
<code>\thickmuskip</code> , 4935,	<code>\Url@moving</code> , 6923	<code>\wherncore</code> , 6439, 6486,
6783, 6789, 6797	<code>\Url@z</code> , 6913	6487
<code>\thinmuskip</code> , 6781, 6787,	<code>\urladdstar</code> , 6599, 6606	<code>\whernfont</code> , 6446, 6454,
6797	<code>\UrlBigBreakPenalty</code> ,	6479, 6489
<code>\thous</code> , 6573	6798, 6803	<code>\whernskip</code> , 6487, 6491, 6492
<code>\thousep</code> , 6513, 6568, 6588	<code>\UrlBigBreaks</code> , 6734, 6803	<code>\whernup</code> , 6494
<code>\thr@@</code> , 3924, 3942	<code>\UrlBreakPenalty</code> ,	<code>\widowpenalty</code> , 3116
<code>\time</code> , 6187, 6188	6798, 6801, 6806	<code>\WPheadings</code> , 3763
<code>\tinyae</code> , 5936	<code>\UrlBreaks</code> , 6731, 6801	<code>\Ws</code> , 5033
<code>\TODO</code> , 5438	<code>\UrlFix</code> , 6708, 6710, 6713,	<code>\wyzejnizej</code> , 5977
<code>\toks</code> , 3735, 3736, 3742, 3743	6715, 6718, 6718	<code>\Wz</code> , 5035
<code>\tolerance</code> , 6022, 6033,	<code>\UrlFont</code> , 6742, 6929	
6329, 6361, 6393	<code>\UrlHyphenchar</code> , 6756, 6763	<code>\xathousep</code> , 6568, 6582
<code>\truetextsuper </code>	<code>\UrlLeft</code> , 6751	<code>\Xedekfrac</code> , 4199
<code>script</code> , 4340,	<code>\Urlmuskip</code> , 6797	<code>\XeLaTeX</code> , 4097
4342	<code>\UrlNoBreaks</code> , 6735, 6805	<code>\XeTeX</code> , 4090
<code>\TrzaskaTilde</code> , 6952	<code>\UrlRight</code> , 6760	<code>\XeTeXdelcode</code> , 4615, 4619
<code>\twocoltoc</code> , 5474, 5483	<code>\UrlSlashKern</code> , 6927	<code>\XeTeXdelimiter</code> , 4648
<code>\twopar</code> , 6384	<code>\UrlSpecials</code> , 6736, 6807	<code>\XeTeXglyph</code> , 6200
<code>\twopar@atleast</code> , 6383,	<code>\urlstyle</code> , 6921, 6936	<code>\XeTeXglyphindex</code> , 6200
6388, 6402	<code>\usc</code> , 5674, 5676	<code>\XeTeXifprefix</code> , 4155
<code>\twopar@default</code> , 6411	<code>\uscacro</code> , 5676	<code>\XeTeXinputencoding</code> , 191
<code>\twopar@defts</code> , 6381, 6404	<code>\usecounter</code> , 3935	<code>\XeTeXmathchardef</code> , 4533
<code>\twopar@core</code> , 6395, 6401,		<code>\XeTeXmathcode</code> , 4536, 4541
6410	<code>\varepsilon</code> , 4020, 4078,	<code>\XeTeXradical</code> , 4814
<code>\twopar@init</code> , 6379	4787	<code>\XeTeXthree</code> , 4158
<code>\type@bslash</code> , 3067	<code>\varnothing</code> , 4724, 5006	<code>\XeTeXversion</code> , 180, 181,
<code>\type@lbrace</code> , 3086, 3192	<code>\varsigma</code> , 4802	190, 698, 4151, 4152,
<code>\tytul</code> , 6319	<code>\vartheta</code> , 4791	6093, 6154, 6400
<code>\tytulowa</code> , 6205	<code>\vee</code> , 4764, 4880, 4933	<code>\xgeq</code> , 4693, 4700, 4702

<code>\Xi</code> , 4770	<code>\xiiunder</code> , <u>678</u> , 680	<code>\zeta</code> , 4788
<code>\xi</code> , 4797	<code>\xleq</code> , 4692, 4695, 4697	<code>\zf@default@options</code> ,
<code>\xiiand</code> , <u>747</u>	<code>\xxt@visible</code> space,	4180
<code>\xiibackslash</code> , <u>734</u> , 738	763, 764	<code>\zf@scale</code> , <u>5944</u> , 5947, 5948
<code>\xiihash</code> , <u>753</u>	<code>\xxt@visible</code> space@fallback,	<code>zwrobcy</code> , <u>6316</u>
<code>\xiilbrace</code> , <u>709</u> , 3260	<u>765</u>	
<code>\xiipercen</code> t, <u>743</u>		<code>\cdot</code> , <u>5536</u>
<code>\xiirbrace</code> , <u>710</u>	<code>\yeshy</code> , 3086, <u>5741</u>	
<code>\xiispace</code> , <u>750</u> , 768,		<code>\-</code> , <u>6114</u> , <u>6160</u> , 6162, 6163, 6164
2605, 2606, 2618	<code>\z@skip</code> , 3067, 5736, 5741,	<code>\-</code> , <u>6096</u> , 6128, <u>6159</u> , 6162,
<code>\xiistring</code> , <u>2604</u>	6103, 6296, 6369, 6670	6163, 6164